SPRING
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2017

C. Filsfils
S. Previdi
P. Psenak
L. Ginsberg
Cisco Systems, Inc.

October 17, 2016

Segment Routing Recursive Information
draft-filsfils-spring-sr-recursing-info-03

Abstract

Segment Routing (SR) allows for a flexible definition of end-to-end paths within IGP topologies by encoding paths as sequences of topological sub-paths, called "segments". These segments are advertised by the link-state routing protocols (IS-IS and OSPF).

There are use cases where it is desirable to utilize a SID associated with a given node in order to transport traffic destined to different local services supported by such node. This document defines the mechanism to do so and illustrates it.

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 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 April 20, 2017.

Table of Contents

1. Introduction

Segment Routing (SR) as defined in [I-D.ietf-spring-segment-routing] utilizes forwarding instructions called "segments" to direct packets through the network. When an MPLS dataplane is in use Segment Identifiers (SIDs) are assigned to prefixes and are associated with a specific algorithm. SIDs may be Local or Global in scope. When a SID has global scope the SID is mapped via the Segment Routing Global Block (SRGB) to a node specific label which acts as the forwarding instruction.

There are use cases where it is desirable to utilize a SID associated with a node N to transport traffic destined to different local services supported by N. This document defines the mechanism to do so and illustrates it.

2. Use Cases

In some deployments, multiple loopback addresses are configured in a single router. Each of these loopback addresses can serve as the address of the node. Specific addresses within this set of node addresses may be used as the endpoint for a particular service or capability. If the number of labeled entries installed in the forwarding plane is a concern, the use of a single label for the set of node addresses (or for some subset of the set of node addresses) can be used in order to reduce the number of forwarding entries required to reach any of the node addresses. This, in turn, would require sharing of a SID among multiple prefixes.

Some deployments attach different services to an edge router in a network via unique interfaces. Rather than assigning a unique SID for the address associated with each service the desired behavior is to use a Node-SID to reach the edge router and then utilize a service specific Local SID to direct the packet to the correct service.

The first use case is a sub-case of the second one where the local SID is not present (e.g. encoded as implicit-null). Hence in the remainder of this document, we will focus on the more generic use case, i.e., utilizing a single Node-SID in combination with an optional Local SID to transport traffic up to the node and then have the node apply the correct service based on the local SID (if present).

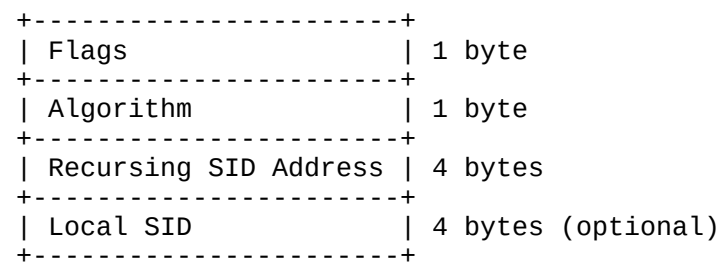
3. Segment Routing Recursing Information (SRRI)

This document introduces and defines the concept of a "Segment Routing Recursing Information (SRRI) that needs to be carried into IGPs (ISIS and OSPF), e.g., as a TLV (or SubTLV) attached to the prefix advertisement.

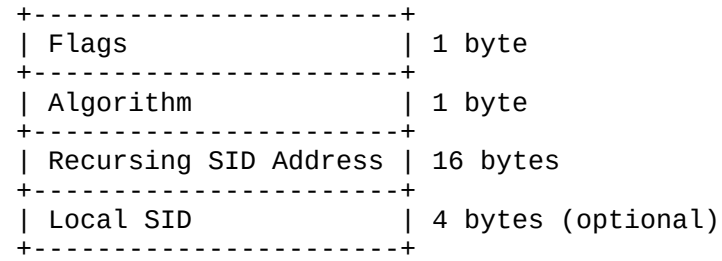
The description in this document is protocol agnostic and can be applied to both IGPs (IS-IS and OSPF). The protocol specific format of this advertisement will be defined in protocol specific specifications.

Advertisement of a prefix P with SRRI (R, Alg, SID-L) indicates that a remote node M MUST use a segment list {SID(R), SID-L) to transport traffic to P; where SID(R) is the prefix SID of R for the specified algorithm. The generic advertisement format is then:

IPv4 SRRI



IPv6 SRRI



where:

- "Flags" is one byte field of flags. Only one flag is currently defined: the V-flag. If set, the receiver of the SRRI MUST verify that the originator of the prefix the SRRI is attached to and the prefix covering the Recursing SID address are originated by the same node ("same origin node").

- "Algorithm" defines the algorithm related to the prefix reachability as defined in [I-D.ietf-spring-segment-routing].

- "Recursing SID Address" contains an IPv4 or IPv6 address whose Node-SID is to be used in order to reach the prefix with which the SRRI information is associated.

- "Local SID" is the local SID allocated to the prefix the SRRI is attached to.

The SRRI is associated with a prefix reachability advertisement. The manner of this association is protocol specific.

The following apply to the SRRI:

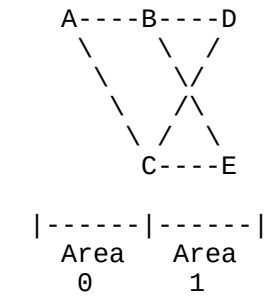
- The prefix reachability advertisement this SRRI is attached to MUST NOT have a Prefix-SID assigned for the algorithm specified in the SRRI.

- Multiple "Recursing SID Address" and "Local SID" MAY be associated with the same parent prefix.

4. Illustration

4.1. Reference Diagram

The following reference topology diagram is used:



A is in Area 0
B and C are ABRs
D and E are in Area 1

D has a loopback 1.0.0.4/32 and Node SID 16004
D has a local service 1.0.0.99/32 with Local SID 30004
E has a loopback 1.0.0.5/32 and Node SID 16005
E has a local service 1.0.0.99/32 with Local SID 30005

For simplicity, we abstracted the algorithm variable in the SRRI and process.

4.2. Description

D advertises prefix 1.0.0.99/32 with SRRI (1.0.0.4, 30004, V=1)

B receives the 1.0.0.99/32 prefix advertisement from D. Since the V-flag is set, B MUST confirm that D also originates the "Recursing SID address" 1.0.0.4/32 (i.e.: "same origin node").

If same origin node is not confirmed, then B does not install any SR RIB entry for prefix 1.0.0.99/32. If same origin node is confirmed, B installs an SR RIB entry for 1.0.0.99/32 which uses the segment list {16004, 30004} and the OIF to 16004.

Furthermore, B leaks the prefix in area 0 and advertises it with the SRRI (1.0.0.4, 30004, V=0). The V-flag unset indicates that area 0 nodes do not need to perform the "same origin node" check.

E advertises prefix 1.0.0.99/32 with the SRRI (1.0.0.5, 30005, V=1)

B does the same for the route from E as he did for the route from D.

Specifically, let us highlight two elements:

o First, B ends up with an ECMP SR-based RIB entry to 1.0.0.99/32:

- {16004, 30004} OIF to 16004
- {16005, 30005} OIF to 16005

o Second, B advertises 1.0.0.99/32 in area 0 with two SRRI's:

- (1.0.0.4, 30004, V=0)
- (1.0.0.5, 30005, V=0)

C does the same for the routes from D and E. Briefly, C advertises 1.0.0.99/32 in area 0 with two SRRI's:

- (1.0.0.4, 30004, V=0)
- (1.0.0.5, 30005, V=0)

A learns 1.0.0.99/32 from B and C and uses normal IGP process to select one, the other or both.

Let us assume A prefers the path via B.

A receives the 1.0.0.99/32 prefix advertisement from B. Since the V-flag is unset, no "same origin node" is verified. A installs an SR RIB entry for 1.0.0.99/32 with an ECMP set of path:

path 1 is {16004, 30004} and the OIF to 16004
path 2 is {16005, 30005} and the OIF to 16005

5. Benefits

The mechanism and SR Recursive Information defined in this document supports:

o single-area and multi-area deployments.

o single-homed and multi-homed prefixes.

o the presence of a Local-SID or not.

- Any combination of the above.

The Segment Routing Recursive Information minimize the number of global prefix SID's.

Finally, the Segment Routing Recursive Information is consistent with the MPLS FIB structure: different FEC's have different entries.

6. IANA Considerations

This document doesn't introduce any new code-point.

7. Security Considerations

TBD

8. Acknowledgements

We would like to thank Les Ginsberg and Ahmed Bashandy for their contributions.

9. Normative References

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-09 (work in progress), July 2016.

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

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Stefano Previdi
Cisco Systems, Inc.
Via Del Serafico, 200
Rome 00142
Italy

Email: sprevidi@cisco.com

Peter Psenak
Cisco Systems, Inc.
Apollo Business Center
Mlynske nivy 43
Bratislava 821 09
Slovakia

Email: ppsenak@cisco.com

Les Ginsberg
Cisco Systems, Inc.
US

Email: ginsberg@cisco.com

SPRING
Internet-Draft
Intended status: Standards Track
Expires: December 22, 2017

C. Filsfils
S. Previdi
P. Psenak
L. Ginsberg
Cisco Systems, Inc.

June 20, 2017

Segment Routing Recursive Information
draft-filsfils-spring-sr-recursing-info-05

Abstract

Segment Routing (SR) allows for a flexible definition of end-to-end paths within IGP topologies by encoding paths as sequences of topological sub-paths, called "segments". These segments are advertised by the link-state routing protocols (IS-IS and OSPF).

There are use cases where it is desirable to utilize a SID associated with a given node in order to transport traffic destined to different local services supported by such node. This document defines the mechanism to do so and illustrates it.

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 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 22, 2017.

Table of Contents

1. Introduction

Segment Routing (SR) as defined in [I-D.ietf-spring-segment-routing] utilizes forwarding instructions called "segments" to direct packets through the network. When an MPLS dataplane is in use Segment Identifiers (SIDs) are assigned to prefixes and are associated with a specific algorithm. SIDs may be Local or Global in scope. When a SID has global scope the SID is mapped via the Segment Routing Global Block (SRGB) to a node specific label which acts as the forwarding instruction.

There are use cases where it is desirable to utilize a SID associated with a node N to transport traffic destined to different local services supported by N. This document defines the mechanism to do so and illustrates it.

2. Use Cases

In some deployments, multiple loopback addresses are configured in a single router. Each of these loopback addresses can serve as the address of the node. Specific addresses within this set of node addresses may be used as the endpoint for a particular service or capability. If the number of labeled entries installed in the forwarding plane is a concern, the use of a single label for the set of node addresses (or for some subset of the set of node addresses) can be used in order to reduce the number of forwarding entries required to reach any of the node addresses. This, in turn, would require sharing of a SID among multiple prefixes.

Some deployments attach different services to an edge router in a network via unique interfaces. Rather than assigning a unique SID for the address associated with each service the desired behavior is to use a Node-SID to reach the edge router and then utilize a service specific Local SID to direct the packet to the correct service.

The first use case is a sub-case of the second one where the local SID is not present (e.g. encoded as implicit-null). Hence in the remainder of this document, we will focus on the more generic use case, i.e., utilizing a single Node-SID in combination with an optional Local SID to transport traffic up to the node and then have the node apply the correct service based on the local SID (if present).

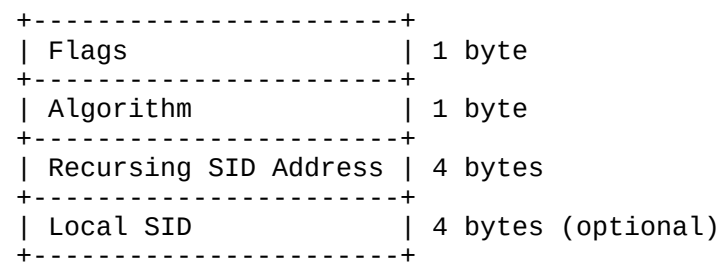
3. Segment Routing Recursing Information (SRRI)

This document introduces and defines the concept of a "Segment Routing Recursing Information (SRRI) that needs to be carried into IGPs (ISIS and OSPF), e.g., as a TLV (or SubTLV) attached to the prefix advertisement.

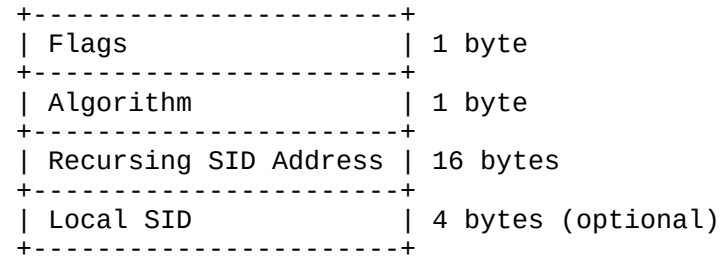
The description in this document is protocol agnostic and can be applied to both IGPs (IS-IS and OSPF). The protocol specific format of this advertisement will be defined in protocol specific specifications.

Advertisement of a prefix P with SRRI (R, Alg, SID-L) indicates that a remote node M MUST use a segment list {SID(R), SID-L) to transport traffic to P; where SID(R) is the prefix SID of R for the specified algorithm. The generic advertisement format is then:

IPv4 SRRI



IPv6 SRRI



where:

- "Flags" is one byte field of flags. Only one flag is currently defined: the V-flag. If set, the receiver of the SRRI MUST verify that the originator of the prefix the SRRI is attached to and the prefix covering the Recursing SID address are originated by the same node ("same origin node").

- "Algorithm" defines the algorithm related to the prefix reachability as defined in [I-D.ietf-spring-segment-routing].

- "Recursing SID Address" contains an IPv4 or IPv6 address whose Node-SID is to be used in order to reach the prefix with which the SRRI information is associated.

- "Local SID" is the local SID allocated to the prefix the SRRI is attached to.

The SRRI is associated with a prefix reachability advertisement. The manner of this association is protocol specific.

The following apply to the SRRI:

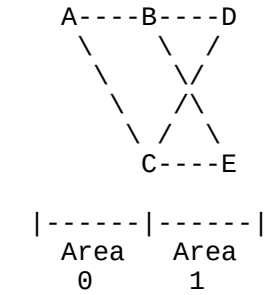
- The prefix reachability advertisement this SRRI is attached to MUST NOT have a Prefix-SID assigned for the algorithm specified in the SRRI.

- Multiple "Recursing SID Address" and "Local SID" MAY be associated with the same parent prefix.

4. Illustration

4.1. Reference Diagram

The following reference topology diagram is used:



A is in Area 0
B and C are ABRs
D and E are in Area 1

D has a loopback 1.0.0.4/32 and Node SID 16004
D has a local service 1.0.0.99/32 with Local SID 30004
E has a loopback 1.0.0.5/32 and Node SID 16005
E has a local service 1.0.0.99/32 with Local SID 30005

For simplicity, we abstracted the algorithm variable in the SRRI and process.

4.2. Description

D advertises prefix 1.0.0.99/32 with SRRI (1.0.0.4, 30004, V=1)

B receives the 1.0.0.99/32 prefix advertisement from D. Since the V-flag is set, B MUST confirm that D also originates the "Recursing SID address" 1.0.0.4/32 (i.e.: "same origin node").

If same origin node is not confirmed, then B does not install any SR RIB entry for prefix 1.0.0.99/32. If same origin node is confirmed, B installs an SR RIB entry for 1.0.0.99/32 which uses the segment list {16004, 30004} and the OIF to 16004.

Furthermore, B leaks the prefix in area 0 and advertises it with the SRRI (1.0.0.4, 30004, V=0). The V-flag unset indicates that area 0 nodes do not need to perform the "same origin node" check.

E advertises prefix 1.0.0.99/32 with the SRRI (1.0.0.5, 30005, V=1)

B does the same for the route from E as he did for the route from D.

Specifically, let us highlight two elements:

o First, B ends up with an ECMP SR-based RIB entry to 1.0.0.99/32:

- {16004, 30004} OIF to 16004
- {16005, 30005} OIF to 16005

o Second, B advertises 1.0.0.99/32 in area 0 with two SRRI's:

- (1.0.0.4, 30004, V=0)
- (1.0.0.5, 30005, V=0)

C does the same for the routes from D and E. Briefly, C advertises 1.0.0.99/32 in area 0 with two SRRI's:

- (1.0.0.4, 30004, V=0)
- (1.0.0.5, 30005, V=0)

A learns 1.0.0.99/32 from B and C and uses normal IGP process to select one, the other or both.

Let us assume A prefers the path via B.

A receives the 1.0.0.99/32 prefix advertisement from B. Since the V-flag is unset, no "same origin node" is verified. A installs an SR RIB entry for 1.0.0.99/32 with an ECMP set of path:

path 1 is {16004, 30004} and the OIF to 16004
path 2 is {16005, 30005} and the OIF to 16005

5. Benefits

The mechanism and SR Recursive Information defined in this document supports:

o single-area and multi-area deployments.

o single-homed and multi-homed prefixes.

o the presence of a Local-SID or not.

o Any combination of the above.

The Segment Routing Recursive Information minimize the number of global prefix SID's.

Finally, the Segment Routing Recursive Information is consistent with the MPLS FIB structure: different FEC's have different entries.

6. IANA Considerations

This document doesn't introduce any new code-point.

7. Security Considerations

TBD

8. Acknowledgements

We would like to thank Les Ginsberg and Ahmed Bashandy for their contributions.

9. Normative References

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-11 (work in progress), February 2017.

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

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Stefano Previdi
Cisco Systems, Inc.
Italy

Email: stefano@previdi.net

Peter Psenak
Cisco Systems, Inc.
Apollo Business Center
Mlynske nivy 43
Bratislava 821 09
Slovakia

Email: ppsenak@cisco.com

Les Ginsberg
Cisco Systems, Inc.
US

Email: ginsberg@cisco.com

Networking Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2017

L. Ginsberg
P. Psenak
S. Previdi
Cisco Systems
M. Pilka

October 26, 2016

Segment Routing Conflict Resolution
draft-ietf-spring-conflict-resolution-02.txt

Abstract

In support of Segment Routing (SR) routing protocols advertise a variety of identifiers used to define the segments which direct forwarding of packets. In cases where the information advertised by a given protocol instance is either internally inconsistent or conflicts with advertisements from another protocol instance a means of achieving consistent forwarding behavior in the network is required. This document defines the policies used to resolve these occurrences.

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 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 April 29, 2017.

Table of Contents

1. Introduction

Segment Routing (SR) as defined in [SR-ARCH] utilizes forwarding instructions called "segments" to direct packets through the network. Depending on the forwarding plane architecture in use, routing protocols advertise various identifiers which define the permissible

values which can be used as segments, which values are assigned to specific prefixes, etc. Where segments have global scope it is necessary to have non-conflicting assignments - but given that the advertisements may originate from multiple nodes the possibility exists that advertisements may be received which are either internally inconsistent or conflicting with advertisements originated by other nodes. In such cases it is necessary to have consistent resolution of conflicts network-wide in order to avoid forwarding loops.

The problem to be addressed is protocol independent i.e., segment related advertisements may be originated by multiple nodes using different protocols and yet the conflict resolution MUST be the same on all nodes regardless of the protocol used to transport the advertisements.

The remainder of this document defines conflict resolution policies which meet these requirements. All protocols which support SR MUST adhere to the policies defined in this document.

2. SR Global Block Inconsistency

In support of an MPLS dataplane routing protocols advertise an SR Global Block (SRGB) which defines a set of label ranges reserved for use by the advertising node in support of SR. The details of how protocols advertise this information can be found in the protocol specific drafts e.g., [SR-OSPF], [SR-OSPFv3], and [SR-IS-IS]. However the protocol independent semantics are illustrated by the following example:

The originating router advertises the following ranges:

```
Range 1: (100, 199)
Range 2: (1000, 1099)
Range 3: (500, 599)
```

The receiving routers concatenate the ranges and build the Segment Routing Global Block (SRGB) as follows:

```
SRGB = (100, 199)
       (1000, 1099)
       (500, 599)
```

The indeces span multiple ranges:

```
index=0 means label 100
...
index 99 means label 199
index 100 means label 1000
index 199 means label 1099
...
index 200 means label 500
...
```

Note that the ranges are an ordered set - what labels are mapped to a given index depends on the placement of a given label range in the set of ranges advertised.

For the set of ranges to be usable the ranges MUST be disjoint. Sender behavior is defined in various SR protocol drafts such as [SR-IS-IS] which specify that senders MUST NOT advertise overlapping ranges.

Receivers of SRGB ranges MUST validate the SRGB ranges advertised by other nodes. If the advertised ranges do not conform to the restrictions defined in the respective protocol specification receivers MUST ignore all advertised SRGB ranges from that node. Operationally the node is treated as though it did not advertise any SRGB ranges. [SR-MPLS] defines the procedures for mapping global SIDs to outgoing labels.

Note that utilization of local SIDs (e.g. adjacency SIDs) advertised by a node is not affected by the state of the advertised SRGB.

3. SR-MPLS Segment Identifier Conflicts

In support of an MPLS dataplane Segment identifiers (SIDs) are advertised and associated with a given prefix. SIDs may be advertised in the prefix reachability advertisements originated by a routing protocol (PFX) . SIDs may also be advertised by a Segment Routing Mapping Server (SRMS).

Information in a SID advertisement is used to construct a mapping entry. A generalized mapping entry can be represented using the following definitions:

Prf - Preference Value (See Section 3.1)
Pi - Initial prefix
Pe - End prefix
L - Prefix length
Lx - Maximum prefix length (32 for IPv4, 128 for IPv6)
Si - Initial SID value
Se - End SID value
R - Range value (See Note 1)
T - Topology
A - Algorithm

A Mapping Entry is then the tuple: (Prf, Src, Pi/L, Si, R, T, A)
Pe = (Pi + ((R-1) << (Lx-L))
Se = Si + (R-1)

NOTE 1: The SID advertised in a prefix reachability advertisement always has an implicit range of 1.

Conflicts in SID advertisements may occur as a result of misconfiguration. Conflicts may occur either in the set of advertisements originated by a single node or between advertisements originated by different nodes. Conflicts which occur within the set of advertisements (P-SID and SRMS) originated by a single node SHOULD be prevented by configuration validation on the originating node.

When conflicts occur, it is not possible for routers to know which of the conflicting advertisements is "correct". In order to avoid forwarding loops and/or blackholes, there is a need for all nodes to resolve the conflicts in a consistent manner. This in turn requires that all routers have identical sets of advertisements and that they all use the same selection algorithm. This document defines procedures to achieve these goals.

3.1. SID Preference

If a node acts as an SRMS, it MAY advertise a preference to be associated with all SRMS SID advertisements sent by that node. The means of advertising the preference is defined in the protocol specific drafts e.g., [SR-OSPF], [SR-OSPFv3], and [SR-IS-IS]. The preference value is an unsigned 8 bit integer with the following properties:

0 - Reserved value indicating advertisements from that node MUST NOT be used.
1 - 255 Preference value

Advertisement of a preference value is optional. Nodes which do not advertise a preference value are assigned a preference value of 128.

All SIDs advertised in prefix reachability advertisements implicitly have a preference value of 192.

3.2. Conflict Types

Two types of conflicts may occur - Prefix Conflicts and SID Conflicts. Examples are provided in this section to illustrate these conflict types.

3.2.1. Prefix Conflict

When different SIDs are assigned to the same prefix we have a "prefix conflict". Prefix conflicts are specific to mapping entries sharing the same topology and algorithm.

Example PC1

(192, 192.0.2.120/32, 200, 1, 0, 0)
(192, 192.0.2.120/32, 30, 1, 0, 0)

The prefix 192.0.2.120/32 has been assigned two different SIDs:
200 by the first advertisement
30 by the second advertisement

Example PC2

(192, 2001:DB8::1/128, 400, 1, 2, 0)
(192, 2001:DB8::1/128, 50, 1, 2, 0)

The prefix 2001:DB8::1/128 has been assigned two different SIDs:
400 by the first advertisement
50 by the second advertisement

Prefix conflicts may also occur as a result of overlapping prefix ranges.

Example PC3

(128, 192.0.2.1/32, 200, 200, 0, 0)
(128, 192.0.2.121/32, 30, 10, 0, 0)

Prefixes 192.0.2.121/32 - 192.0.2.130/32 are assigned two different SIDs:
320 through 329 by the first advertisement
30 through 39 by the second advertisement

Example PC4
(128, 2001:DB8::1/128, 400, 200, 2, 0)
(128, 2001:DB8::121/128, 50, 10, 2, 0)

Prefixes 2001:DB8::121/128 - 2001:DB8::130/128 are assigned two different SIDs:
420 through 429 by the first advertisement
50 through 59 by the second advertisement

Examples PC3 and PC4 illustrate a complication - only part of the range advertised in the first advertisement is in conflict. It is logically possible to isolate the conflicting portion and try to use the non-conflicting portion(s).

A variant of the overlapping prefix range is a case where we have overlapping prefix ranges but no actual SID conflict.

Example PC5

(128, 192.0.2.1/32, 200, 200, 0, 0)
(128, 192.0.2.121/32, 320, 10, 0, 0)

(128, 2001:DB8::1/128, 400, 200, 2, 0)
(128, 2001:DB8::121/128, 520, 10, 2, 0)

Although there is prefix overlap between the two IPv4 entries (and the two IPv6 entries) the same SID is assigned to all of the shared prefixes by the two entries.

Given two mapping entries:

(Prf, P1/L1, S1, R1, T1, A1) and
(Prf, P2/L2, S2, R2, T2, A2)

where P1 <= P2

a prefix conflict exists if all of the following are true:

1)(T1 == T2) && (A1 == A2)
2)P1 <= P2
3)The prefixes are in the same address family.
2)L1 == L2
3)(P1e >= P2) && ((S1 + (P2 - P1)) != S2)

3.2.2. SID Conflict

When the same SID has been assigned to multiple prefixes we have a "SID conflict". SID conflicts are independent of address-family, independent of prefix len, independent of topology, and independent of algorithm. A SID conflict occurs when a mapping entry which has previously been checked to have no prefix conflict assigns one or more SIDs that are assigned by another entry which also has no prefix conflicts.

Example SC1

(192, 192.0.2.1/32, 200, 1, 0, 0)
(192, 192.0.2.222/32, 200, 1, 0, 0)
SID 200 has been assigned to 192.0.2.1/32 by the
first advertisement.
The second advertisement assigns SID 200 to 192.0.2.222/32.

Example SC2

(192, 2001:DB8::1/128, 400, 1, 2, 0)
(192, 2001:DB8::222/128, 400, 1, 2, 0)
SID 400 has been assigned to 2001:DB8::1/128 by the
first advertisement.
The second advertisement assigns SID 400 to 2001:DB8::222/128

SID conflicts may also occur as a result of overlapping SID ranges.

Example SC3

(128, 192.0.2.1/32, 200, 200, 0, 0)
(128, 198.51.100.1/32, 300, 10, 0, 0)

SIDs 300 - 309 have been assigned to two different prefixes.
The first advertisement assigns these SIDs
to 192.0.2.101/32 - 192.0.2.110/32.
The second advertisement assigns these SIDs to
198.51.100.1/32 - 198.51.100.10/32.

Example SC4
(128, 2001:DB8::1/128, 400, 200, 2, 0)
(128, 2001:DB8:1::1/128, 500, 10, 2, 0)

SIDs 500 - 509 have been assigned to two different prefixes.
The first advertisement assigns these SIDs to
2001:DB8::101/128 - 2001:DB8::10A/128.
The second advertisement assigns these SIDs to
2001:DB8:1::1/128 - 2001:DB8:1::A/128.

Examples SC3 and SC4 illustrate a complication - only part of the range advertised in the first advertisement is in conflict.

3.3. Processing conflicting entries

Two general approaches can be used to process conflicting entries.

1. Conflicting entries can be ignored

2. A standard preference algorithm can be used to choose which of the conflicting entries will be used

The following sections discuss these two approaches in more detail.

Note: This document does not discuss any implementation details i.e. what type of data structure is used to store the entries (trie, radix tree, etc.) nor what type of keys may be used to perform lookups in the database.

3.3.1. Policy: Ignore conflicting entries

In cases where entries are in conflict none of the conflicting entries are used i.e., the network operates as if the conflicting advertisements were not present.

Implementations are required to identify the conflicting entries and ensure that they are not used.

3.3.2. Policy: Preference Algorithm/Quarantine

For entries which are in conflict properties of the conflicting advertisements are used to determine which of the conflicting entries are used in forwarding and which are "quarantined" and not used. The entire quarantined entry is not used.

This approach requires that conflicting entries first be identified and then evaluated based on a preference rule. Based on which entry is preferred this in turn may impact what other entries are considered in conflict i.e. if A conflicts with B and B conflicts with C - it is possible that A does NOT conflict with C. Hence if as a result of the evaluation of the conflict between A and B, entry B is not used the conflict between B and C will not be detected.

3.3.3. Policy: Preference algorithm/ignore overlap only

A variation of the preference algorithm approach is to quarantine only the portions of the less preferred entry which actually conflicts. The original entry is split into multiple ranges. The ranges which are in conflict are quarantined. The ranges which are not in conflict are used in forwarding. This approach adds complexity as the relationship between the derived sub-ranges of the original mapping entry have to be associated with the original entry - and every time some change to the advertisement database occurs the derived sub-ranges have to be recalculated.

3.3.4. Preference Algorithm

The following algorithm is used to select the preferred mapping entry when a conflict exists. Evaluation is made in the order specified. Prefix conflicts are evaluated first. SID conflicts are then evaluated on the Active entries remaining after Prefix Conflicts have been resolved.

1. Higher preference value wins
2. Smaller range wins
3. IPv6 entry wins over IPv4 entry
4. Longer prefix length wins
5. Smaller algorithm wins

6. Smaller starting address (considered as an unsigned integer value) wins

7. Smaller starting SID wins

8. If topology IDs are NOT identical both entries MUST be ignored

As SIDs associated with prefix reachability advertisements have a preference of 192 while by default SRMS preference is 128, the default behavior is then to prefer SIDs advertised in prefix reachability advertisements over SIDs advertised by SRMSs, but an operator can choose to override this behavior by setting SRMS preference higher than 192.

Preferring advertisements with smaller range has the nice property that a single misconfiguration of an SRMS entry with a large range will not be preferred over a large number of advertisements with smaller ranges.

Since topology identifiers are locally scoped, it is not possible to make a consistent choice network wide when all elements of a mapping entry are identical except for the topology. This is why both entries MUST be ignored in such cases (Rule #8 above). Note that Rule #8 only applies when considering SID conflicts since Prefix conflicts are restricted to a single topology.

3.3.5. Example Behavior - Single Topology/Algorithm

The following mapping entries exist:in the database. For brevity, Topology/Algorithm is omitted and assumed to be (0,0) in all entries.

1. (192, 192.0.2.1/32, 100, 1)

2. (192, 192.0.2.101/32, 200, 1)

3. (128, 192.0.2.1/32, 400, 255) !Prefix conflict with entries 1 and 2

4. (128, 198.51.100.40/32, 200,1) !SID conflict with entry 2

The table below shows what mapping entries will be used in the forwarding plane (Active) and which ones will not be used (Excluded) under the three candidate policies:

Policy	Active Entries	Excluded Entries
Ignore		(192,192.0.2.1/32,100,1) (192,192.0.2.101/32,200,1) (128,192.0.2.1/32,400,255) (128,198.51.100.40/32,200,1)
Quarantine	(192,192.0.1.1/32,100,1) (192,192.0.2.101/32,200,1)	(128,192.0.2.1/32,400,255) (128,198.51.100.40/32,200,1)
Overlap-Only	(192,192.0.2.1/32,100,1) (192,192.0.2.101/32,200,1) *(128,192.0.2.2/32,401,99) *(128,192.0.2.102/32, 501,153)	(128,198.51.100.40/32,200,1) *(128,192.0.2.1/32,400,1) *(128,192.0.2.101/32,500,1)

* Derived from (128,192.0.2.1/32,400,300)

3.3.6. Example Behavior - Multiple Topologies

When using a preference rule the order in which conflict resolution is applied has an impact on what entries are usable when entries for multiple topologies (or algorithms) are present. The following mapping entries exist in the database:

1. (192, 192.0.2.1/32, 100, 1, 0, 0) !Topology 0

2. (192, 192.0.2.1/32, 200, 1, 0, 0) !Topology 0, Prefix Conflict with entry #1

3. (192, 198.51.100.40/32, 200,1,1,0) ! Topology 1, SID conflict with entry 2

The table below shows what mapping entries will be used in the forwarding plane (Active) and which ones will not be used (Excluded) under the Quarantine Policy based on the order in which conflict resolution is applied.

Order	Active Entries	Excluded Entries
Prefix-Conflict First	(192,192.0.2.1/32,100,1,0,0) (192,198.51.100.40/32,200,1,1,0)	(192,192.0.2.101/32,200,1,0)
SID-Conflict First	(192,192.0.2.1/32,100,1,0,0)	(192,192.0.2.101/32,200,1,0) (192,198.51.100.40/32,200,1,1,0)

This illustrates the advantage of evaluating prefix conflicts within a given topology (or algorithm) before evaluating topology (or algorithm) independent SID conflicts. It insures that entries which will be excluded based on intratopology preference will not prevent a SID assigned in another topology from being considered Active.

3.3.7. Evaluation of Policy Alternatives

The previous sections have defined three alternatives for resolving conflicts - ignore, quarantine, and ignore overlap-only.

The ignore policy impacts the greatest amount of traffic as forwarding to all destinations which have a conflict is affected.

Quarantine allows forwarding for some destinations which have a conflict to be supported.

Ignore overlap-only maximizes the destinations which will be forwarded as all destinations covered by some mapping entry (regardless of range) will be able to use the SID assigned by the winning range. This alternative increases implementation complexity as compared to quarantine. Mapping entries with a range greater than 1 which are in conflict with other mapping entries have to internally be split into 2 or more "derived mapping entries". The derived mapping entries then fall into two categories - those that are in conflict with other mapping entries and those which are NOT in conflict. The former are ignored and the latter are used. Each time the underived mapping database is updated the derived entries have to be recomputed based on the updated database. Internal data structures have to be maintained which maintain the relationship between the advertised mapping entry and the set of derived mapping entries. All nodes in the network have to achieve the same behavior regardless of implementation internals.

There is then a tradeoff between a goal of maximizing traffic delivery and the risks associated with increased implementation complexity.

Consensus of the working group is that maximizing traffic delivery is the most important deployment consideration - therefore ignore-overlap-only is specified as the standard policy which MUST be implemented by all nodes which support SR-MPLS.

3.3.8. Guaranteeing Database Consistency

In order to obtain consistent active entries all nodes in a network MUST have the same mapping entry database. Mapping entries can be obtained from a variety of sources.

- SIDs can be configured locally for prefixes assigned to interfaces on the router itself. Only SIDs which are advertised to protocol peers can be considered as part of the mapping entry database.
- SIDs can be received in prefix reachability advertisements from protocol peers. These advertisements may originate from peers local to the area or be leaked from other areas and/or redistributed from other routing protocols.
- SIDs can be received from SRMS advertisements - these advertisements can originate from routers local to the area or leaked from other areas
- In cases where multiple routing protocols are in use mapping entries advertised by all routing protocols MUST be included.

4. Scope of SR-MPLS SID Conflicts

The previous section defines the types of SID conflicts and procedures to resolve such conflicts when using an MPLS dataplane. The mapping entry database used MUST be populated with entries for destinations for which the associated SID will be used to derive the labels installed in the forwarding plane of routers in the network. This consists of entries associated with intra-domain routes.

There are cases where destinations which are external to the domain are advertised by protocol speakers running within that network - and it is possible that those advertisements have SIDs associated with those destinations. However, if reachability to a destination is topologically outside the forwarding domain of the protocol instance then the SIDs for such destinations will never be installed in the forwarding plane of any router within the domain - so such advertisements cannot create a SID conflict within the domain. Such

entries therefore MUST NOT be installed in the database used for intra-domain conflict resolution.

Consider the case of two sites "A and B" associated with a given [RFC4364] VPN. Connectivity between the sites is via a provider backbone. SIDs associated with destinations in Site A will never be installed in the forwarding plane of routers in Site B. Reachability between the sites (assuming SR is being used across the backbone) only requires using a SID associated with a gateway PE. So a destination in Site A MAY use the same SID as a destination in Site B without introducing any conflict in the forwarding plane of routers in Site A.

Such cases are handled by insuring that the mapping entries in the database used by the procedures defined in the previous section only include entries associated with advertisements within the site.

5. Security Considerations

The ability to introduce SID conflicts into a deployment may compromise traffic forwarding. Protocol specific security mechanisms SHOULD be used to insure that all SID advertisements originate from trusted sources.

6. IANA Consideration

This document has no actions for IANA.

7. Acknowledgements

The authors would like to thank Jeff Tantsura, Wim Henderickx, and Bruno Decraene for their careful review and content suggestions.

8. References

8.1. Normative References

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

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

[SR-IS-IS] "IS-IS Extensions for Segment Routing, draft-ietf-isis-segment-routing-extensions-08(work in progress)", October 2016.

[SR-MPLS] "Segment Routing with MPLS dataplane, draft-ietf-spring-segment-routing-mpls-05(work in progress)", July 2016.

[SR-OSPF] "OSPF Extensions for Segment Routing, draft-ietf-ospf-segment-routing-extensions-09(work in progress)", July 2016.

[SR-OSPFv3] "OSPFv3 Extensions for Segment Routing, draft-ietf-ospf-ospfv3-segment-routing-extensions-06(work in progress)", July 2016.

8.2. Informational References

[SR-ARCH] "Segment Routing Architecture, draft-ietf-spring-segment-routing-09(work in progress)", July 2016.

Authors' Addresses

Les Ginsberg
Cisco Systems
821 Alder Drive
Milpitas, CA 95035
USA

Email: ginsberg@cisco.com

Peter Psenak
Cisco Systems
Apollo Business Center Mlynske nivy 43
Bratislava 821 09
Slovakia

Email: ppsenak@cisco.com

Stefano Previdi
Cisco Systems
Via Del Serafico 200
Rome 0144
Italy

Email: sprevidi@cisco.com

Martin Pilka

Email: martin@infobox.sk

Networking Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2018

L. Ginsberg
P. Psenak
S. Previdi
Cisco Systems
M. Pilka

July 2, 2017

Segment Routing MPLS Conflict Resolution
draft-ietf-spring-conflict-resolution-05.txt

Abstract

In support of Segment Routing (SR) for an MPLS data plane routing protocols advertise a variety of identifiers used to define the segments which direct forwarding of packets. In cases where the information advertised by a given protocol instance is either internally inconsistent or conflicts with advertisements from another protocol instance a means of achieving consistent forwarding behavior in the network is required. This document defines the policies used to resolve these occurrences.

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 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 January 3, 2018.

Table of Contents

1. Introduction

Segment Routing (SR) as defined in [SR-ARCH] utilizes forwarding instructions called "segments" to direct packets through the network. Depending on the forwarding plane architecture in use, routing protocols advertise various identifiers which define the permissible values which can be used as segments, which values are assigned to specific prefixes, etc. Where segments have global scope it is necessary to have non-conflicting assignments - but given that the advertisements may originate from multiple nodes the possibility exists that advertisements may be received which are either internally inconsistent or conflicting with advertisements originated by other nodes. In such cases it is necessary to have consistent resolution of conflicts network-wide in order to avoid forwarding loops.

This document is limited to discussion of conflict resolution for identifiers used in an MPLS data plane.

The problem to be addressed is protocol independent i.e., segment related advertisements may be originated by multiple nodes using different protocols and yet the conflict resolution MUST be the same on all nodes regardless of the protocol used to transport the advertisements.

The remainder of this document defines conflict resolution policies which meet these requirements. All protocols which support SR MUST adhere to the policies defined in this document.

2. SR Global Block Inconsistency

In support of an MPLS dataplane [SR-MPLS] routing protocols advertise an SR Global Block (SRGB) which defines a set of label ranges reserved for use by the advertising node in support of SR. The details of how protocols advertise this information can be found in the protocol specific drafts e.g., [SR-OSPF], [SR-OSPFv3], [SR-IS-IS], and [SR-BGP]. However the protocol independent semantics are illustrated by the following example:

The originating router advertises the following ranges:

```
Range 1: (100, 199)
Range 2: (1000, 1099)
Range 3: (500, 599)
```

The receiving routers concatenate the ranges and build the Segment Routing Global Block (SRGB) as follows:

```
SRGB = (100, 199)
       (1000, 1099)
       (500, 599)
```

The indeces span multiple ranges:

```
index=0 means label 100
...
index 99 means label 199
index 100 means label 1000
index 199 means label 1099
...
index 200 means label 500
...
```

Note that the ranges are an ordered set - what labels are mapped to a given index depends on the placement of a given label range in the set of ranges advertised.

For the set of ranges to be usable the ranges MUST be disjoint. Sender behavior is defined in various SR protocol drafts such as [SR-IS-IS] which specify that senders MUST NOT advertise overlapping ranges.

Receivers of SRGB ranges MUST validate the SRGB ranges advertised by other nodes. If the advertised ranges do not conform to the restrictions defined in the respective protocol specification receivers MUST ignore all advertised SRGB ranges from that node. Operationally the node is treated as though it did not advertise any SRGB ranges. [SR-MPLS] defines the procedures for mapping global SIDs to outgoing labels.

Note that utilization of local SIDs (e.g. adjacency SIDs) advertised by a node is not affected by the state of the advertised SRGB.

3. SR-MPLS Segment Identifier Conflicts

In support of an MPLS dataplane Segment Identifiers (SIDs) are advertised and associated with a given prefix. SIDs may be advertised in the prefix reachability advertisements originated by a routing protocol (PFX) . SIDs may also be advertised by a Segment Routing Mapping Server (SRMS). How this is done is defined in the protocol specific drafts e.g., [SR-OSPF], [SR-OSPFv3], [SR-IS-IS], and [SR-BGP]

Information in a SID advertisement is used to construct a mapping entry. A generalized mapping entry can be represented using the following definitions:

Prf - Preference Value (See Section 3.1)
Pi - Initial prefix
Pe - End prefix
L - Prefix length
Lx - Maximum prefix length (32 for IPv4, 128 for IPv6)
Si - Initial SID value
Se - End SID value
R - Range value (See Note 1)
T - Topology
A - Algorithm (see [SR-ARCH])

A Mapping Entry is then the tuple: (Prf, Pi/L, Si, R, T, A)
Pe = (Pi + ((R-1) << (Lx-L))
Se = Si + (R-1)

NOTE 1: The SID advertised in a prefix reachability advertisement always has an implicit range of 1.

NOTE 2: IPv4/IPv6 addresses can be viewed as 32/128 bit integers. Where operations such as addition, subtraction, and/or bit shifting are specified for prefixes this should be interpreted as operations on the integer representation of a prefix.

Note: Topology is a locally scoped identifier assigned by each router. Although it may have an association with Multitopology Identifiers (MTID) advertised by routing protocols it is NOT equivalent to these identifiers. MTIDs are scoped by a given routing protocol. MTID ranges are protocol specific and there may be standardized protocol specific MTID assignments for topologies of a specific type (e.g., an AFI specific topology). As mapping entries can be sourced from multiple protocols it is not possible to use a

network scoped identifier for a topology when storing mapping entries in the local datbase.

Conflicts in SID advertisements may occur as a result of misconfiguration. When conflicts occur, it is not possible for routers to know which of the conflicting advertisements is "correct". In order to avoid forwarding loops and/or blackholes, there is a need for all nodes to resolve the conflicts in a consistent manner. This in turn requires that all routers have identical sets of advertisements and that they all use the same selection algorithm. This document defines procedures to achieve these goals.

3.1. SID Preference

If a node acts as an SRMS, it MAY advertise a preference to be associated with all SRMS SID advertisements sent by that node. The means of advertising the preference is defined in the protocol specific drafts e.g., [SR-OSPF], [SR-OSPFv3], and [SR-IS-IS]. The preference value is an unsigned 8 bit integer with the following properties:

0 - Reserved value indicating advertisements from that node MUST NOT be used.
1 - 255 Preference value

Advertisement of a preference value is optional. Nodes which do not advertise a preference value are assigned a preference value of 128.

All SIDs advertised in prefix reachability advertisements originated by an IGP implicitly have a preference value of 192.

All SIDs advertised in prefix reachability advertisements originated by BGP implicitly have a preference value of 64.

These preference values are deliberately chosen to favor SID advertisements originated within a domain (IGP and SRMS) over SID advertisements which may have been imported from other domains (BGP). In addition, as BGP originated advertisements may not be known on all nodes within a domain (because not every node will be a BGP speaker), the presence of a BGP originated mapping entry MUST NOT cause a mapping entry originated within the domain to become unusable as this would introduce inconsistency in the set of SIDs considered usable by a node which has the BGP originated mapping entries and the set considered usable by nodes without the BGP originated mapping entries.

3.2. Conflict Types

Two types of conflicts may occur - Prefix Conflicts and SID Conflicts. Examples are provided in this section to illustrate these conflict types and generic definitions of algorithms to determine when there is a conflict are presented.

3.2.1. Prefix Conflict

When different SIDs are assigned to the same prefix we have a "prefix conflict". Prefix conflicts are limited to mapping entries sharing the same topology, algorithm, address-family, and prefix length.

3.2.1.1. Prefix Conflict Examples

The simplest example is when two advertisements with a range of 1 assign different SIDs to the same prefix.

Example PC1

(192, 192.0.2.120/32, 200, 1, 0, 0)
(192, 192.0.2.120/32, 30, 1, 0, 0)

The prefix 192.0.2.120/32 has been assigned two different SIDs:
  200 by the first advertisement
  30 by the second advertisement

Example PC2

(192, 2001:DB8::1/128, 400, 1, 2, 0)
(192, 2001:DB8::1/128, 50, 1, 2, 0)

The prefix 2001:DB8::1/128 has been assigned two different SIDs:
 400 by the first advertisement
 50 by the second advertisement

Prefix conflicts may also occur as a result of overlapping prefix ranges.

Example PC3

(128, 192.0.2.1/32, 200, 200, 0, 0)
(128, 192.0.2.121/32, 30, 10, 0, 0)

Prefixes 192.0.2.121/32 - 192.0.2.130/32 are assigned two different SIDs:
320 through 329 by the first advertisement
30 through 39 by the second advertisement

Example PC4
(128, 2001:DB8::1/128, 400, 200, 2, 0)
(128, 2001:DB8::121/128, 50, 10, 2, 0)

Prefixes 2001:DB8::121/128 - 2001:DB8::130/128 are assigned two different SIDs:
420 through 429 by the first advertisement
50 through 59 by the second advertisement

Examples PC3 and PC4 illustrate a complication - only part of the range advertised in the first advertisement is in conflict. It is logically possible to consider the sub-range(s) which are in conflict as unusable while considering the sub-range(s) not in conflict as usable.

A variant of the overlapping prefix range is a case where we have overlapping prefix ranges but no actual prefix conflict.

Example PC5

(128, 192.0.2.1/32, 200, 200, 0, 0)
(128, 192.0.2.121/32, 320, 10, 0, 0)

(128, 2001:DB8::1/128, 400, 200, 2, 0)
(128, 2001:DB8::121/128, 520, 10, 2, 0)

Although there is prefix overlap between the two IPv4 entries (and the two IPv6 entries) the same SID is assigned to all of the shared prefixes by the two entries.

3.2.1.2. Prefix Conflict Generic Algorithm

The following generic algorithm can be used to determine when any two mapping entries have Prefix Conflicts and what the set of prefixes in conflict are.

Given two mapping entries:

(Prf, P1/L1, S1, R1, T1, A1) and
(Prf, P2/L2, S2, R2, T2, A2)

where P1 <= P2

a prefix conflict exists if all of the following are true:

1)Topologies, algorithms, and prefix lengths are identical

(T1 == T2) && (A1 == A2) && (L1 == L2)

2)The prefixes are in the same address-family.

3)If there are overlapping prefixes in the two ranges and
if there are different SIDs assigned to any of the prefixes
in the overlapping range

(P1e >= P2) && ((S1 + ((P2 - P1) >> (Lx-L1)) != S2)

Prefixes in the following range are in conflict:

P2 through MIN(P1e,P2e)

3.2.2. SID Conflict

When the same SID has been assigned to multiple prefixes we have a "SID conflict". SID conflicts are independent of address-family, independent of prefix len, independent of topology, and independent of algorithm.

3.2.2.1. SID Conflict Examples

The simplest example is when two mapping entries with a range of 1 assigns different SIDs to the same prefix.

Example SC1

(192, 192.0.2.1/32, 200, 1, 0, 0)
(192, 192.0.2.222/32, 200, 1, 0, 0)
SID 200 has been assigned to 192.0.2.1/32 by the
first advertisement.
The second advertisement assigns SID 200 to 192.0.2.222/32.

Example SC2

(192, 2001:DB8::1/128, 400, 1, 2, 0)
(192, 2001:DB8::222/128, 400, 1, 2, 0)
SID 400 has been assigned to 2001:DB8::1/128 by the
first advertisement.
The second advertisement assigns SID 400 to 2001:DB8::222/128

SID conflicts may also occur as a result of overlapping SID ranges.

Example SC3

(128, 192.0.2.1/32, 200, 200, 0, 0)
(128, 198.51.100.1/32, 300, 10, 0, 0)

SIDs 300 - 309 have been assigned to two different prefixes.
The first advertisement assigns these SIDs
to 192.0.2.101/32 - 192.0.2.110/32.
The second advertisement assigns these SIDs to
198.51.100.1/32 - 198.51.100.10/32.

Example SC4
(128, 2001:DB8::1/128, 400, 200, 2, 0)
(128, 2001:DB8:1::1/128, 500, 10, 2, 0)

SIDs 500 - 509 have been assigned to two different prefixes.
The first advertisement assigns these SIDs to
2001:DB8::101/128 - 2001:DB8::10A/128.
The second advertisement assigns these SIDs to
2001:DB8:1::1/128 - 2001:DB8:1::A/128.

Examples SC3 and SC4 illustrate a complication - only part of the range advertised in the first advertisement is in conflict.

SID conflicts may also occur because the same SID has been used in two different algorithms, two different topologies, two different address families, or prefixes with two different lengths.

Example SC5

(128, 192.0.2.1/32, 200, 1, 0, 0)
(128, 192.0.2.1/32, 200, 1, 0, 1)

SID 200 has been assigned to the same prefix with two different algorithms.

Example SC6
(128, 192.0.2.1/32, 200, 1, 0, 0)
(128, 2001:DB8::1/128, 200, 1, 0, 0)

SID 200 has been assigned to prefixes in two different address-families.

3.2.2.2. SID Conflict Generic Algorithm

The following generic algorithm can be used to determine when any two mapping entries have SID Conflicts and what the set of SIDs in conflict are.

Given two mapping entries:

(Prf, P1/L1, S1, R1, T1, A1) and
(Prf, P2/L2, S2, R2, T2, A2)

a SID conflict exists if all of the following are true:

1)If the SID ranges overlap

(S1 <= S2) && (S1e >= S2)

2)If the same SID is assigned to prefixes with different address-families, prefix lengths, topologies, or algorithms or the same SID is assigned to two different prefixes for any of the prefixes in either range.

P1 and P2 are NOT in the same address family OR
L1 != L2 OR
T1 != T2 OR
A1 != A2 OR
(P1 + ((S1e-S2) << (L1x-L1))) != P2

SIDs in the following range are in conflict:

S2 through MIN(S1e,S2e)

3.3. Preference rule for resolving conflicts

When a conflict is detected the following algorithm is used to select the preferred mapping entry. Evaluation is made in the order specified. Prefix conflicts are evaluated first. SID conflicts are then evaluated on the Active entries remaining after Prefix Conflicts have been resolved.

1. Higher preference value wins
2. Smaller range wins
3. IPv6 entry wins over IPv4 entry
4. Longer prefix length wins
5. Smaller starting address (considered as an unsigned integer value) wins
6. Smaller algorithm wins
7. Smaller starting SID wins
8. If topology IDs are NOT identical both entries MUST be ignored

When applying the preference rule to prefix/SID pairs associated with an advertised mapping entry with a range greater than one, each prefix/SID pair in the range is considered as having the range associated with the advertised mapping entry. For example:

Advertised mapping entry: (128, 192.0.2.1/32, 200, 200, 0, 0)

The advertisement covers 200 prefix/SID pairs:
192.0.2.1/32 200
192.0.2.2/32 201
...
192.0.2.200/32 399

Each of these prefix/SID pairs is considered as having a range of 200 when applying Rule #2 above.

As SIDs associated with prefix reachability advertisements have a preference of 192 and an implied range of 1 while by default SRMS preference is 128, the default behavior is then to prefer SIDs advertised in prefix reachability advertisements over SIDs advertised by SRMSs, but an operator can choose to override this behavior by setting SRMS preference higher than 192.

Preferring advertisements with smaller range has the nice property that a single misconfiguration of an SRMS entry with a large range will not be preferred over a large number of advertisements with smaller ranges.

Since topology identifiers are locally scoped, it is not possible to make a consistent choice network wide when all elements of a mapping entry are identical except for the topology. This is why both entries MUST be ignored in such cases (Rule #8 above). Note that Rule #8 only applies when considering SID conflicts since Prefix conflicts are restricted to a single topology.

3.4. Conflict Resolution Algorithm

The following logical steps MUST be followed in the order specified when resolving conflicts.

Step 1: Resolve Prefix Conflicts (same topology/address family/ algorithm)

For each supported topology/address family/algorithm examine all qualifying mapping entries in the following order:

1)Preference (start w highest)
2)Range (start w smallest)
3)Prefix length (start w longest)
4)Address (start w smallest)
5)SID (start w smallest)

At each step if a prefix conflict is detected the losing prefix/SID pair is declared Inactive and is not considered in any subsequent steps. The remaining prefix/SID pairs are Active.

Mapping entries with Active prefix/SID pairs after completion of Step 1 are fed into ...

Step 2: SID Conflicts (across all topologies/address families/ algorithms)

Examine all Active prefix/SID pairs from Step #1 in the following order:

1)Preference (start w highest)
2)Range (start w smallest)
3)IPv6 entries
  a)Prefix length (start w longest)
  b)Address (start w smallest)
4)IPv4 entries
  a)Prefix Length (start w longest)
  b)Address (start w smallest)
5)Algorithm (start w smallest)
6)SID (start w smallest)

Prefix/SID pairs which are identical and are associated with the same topology are duplicates - both entries MUST be considered as Active.
Prefix/SID pairs which are identical and are associated with different topologies MUST both be considered Inactive.

Active Entries in the database may be used in forwarding. Inactive entries MUST NOT be used in forwarding.

Note that when the database of mapping entries changes the full set of logical steps MUST be reapplied to the entire database as conflict resolution is NOT transitive.

NOTE: Clever implementors may realize optimizations when rerunning the algorithm by evaluating changed entries as to whether they have potential conflicts with any of the existing entries in the database (both active and inactive). Such optimizations are outside the scope of this specification. The normative behavior is defined by the logical algorithm above.

3.5. Example Behavior - Single Topology/Address Family/Algorithm

The following mapping entries exist in the database. For brevity, Topology/Algorithm is omitted and assumed to be (0,0) in all entries.

1. (192, 192.0.2.1/32, 100, 1)

2. (192, 192.0.2.101/32, 200, 1)

3. (128, 192.0.2.1/32, 400, 255) !Prefix conflict with entries 1 and 2

4. (128, 198.51.100.40/32, 200,1) !SID conflict with entry 2

The table below shows what mapping entries will be used in the forwarding plane (Active) and which ones will not be used (Inactive)

Active Entries	Inactive Entries
(192,192.0.2.1/32,100,1)	(128,198.51.100.40/32,200,1)
(192,192.0.2.101/32,200,1)	*(128,192.0.2.1/32,400,1)
*(128,192.0.2.2/32,401,99)	*(128,192.0.2.101/32,500,1)
*(128,192.0.2.102/32,501,154)	

* Derived from (128,192.0.2.1/32,400,255)

3.6. Example Behavior - Multiple Topologies

When using a preference rule the order in which conflict resolution is applied has an impact on what entries are Active when entries for multiple topologies (or algorithms) are present. The following mapping entries exist in the database:

1. (192, 192.0.2.1/32, 100, 1, 0, 0) !Topology 0

2. (192, 192.0.2.1/32, 200, 1, 0, 0) !Topology 0, Prefix Conflict with entry #1

3. (192, 198.51.100.40/32, 200,1,1,0) ! Topology 1, SID conflict with entry 2

The table below shows what mapping entries will be used in the forwarding plane (Active) and which ones will not be used (Inactive) based on the order in which conflict resolution is applied.

Order	Active Entries	Inactive Entries
Prefix-Conflict First	(192,192.0.2.1/32,100,1,0,0) (192,198.51.100.40/32,200,1,1,0)	(192,192.0.2.101/32,200,1,0)
SID-Conflict First	(192,192.0.2.1/32,100,1,0,0)	(192,192.0.2.101/32,200,1,0) (192,198.51.100.40/32,200,1,1,0)

This illustrates the advantage of evaluating prefix conflicts within a given topology (or algorithm) before evaluating topology (or algorithm) independent SID conflicts. It insures that entries which will be excluded based on intratopology preference will not prevent a SID assigned in another topology from being considered Active.

3.7. Guaranteeing Database Consistency

In order to obtain consistent active entries all nodes in a network MUST have the same mapping entry database. Mapping entries can be obtained from a variety of sources.

- SIDs can be configured locally for prefixes assigned to interfaces on the router itself. Only SIDs which are advertised to protocol peers can be considered as part of the mapping entry database.
- SIDs can be received in prefix reachability advertisements from protocol peers. These advertisements may originate from peers local to the area or be leaked from other areas and/or redistributed from other routing protocols.
- SIDs can be received from SRMS advertisements - these advertisements can originate from routers local to the area or leaked from other areas
- In cases where multiple routing protocols are in use mapping entries advertised by all routing protocols MUST be included.

3.8. Minimizing the occurence of conflicts

Conflicts in SID advertisements are always the result of a misconfiguration. Conflicts may occur either in the set of advertisements originated by a single node or between advertisements originated by different nodes.

Conflicts which occur within the set of advertisements (PFX and SRMS) originated by a single node SHOULD be prevented by configuration validation on the originating node.

It is possible to minimize the occurrence of conflicts between advertisements originated by different routers if new configuration is validated against the current state of the conflict resolution database before the configuration is advertised. How this is done is an implementation issue which is out of scope of this document.

4. Scope of SR-MPLS SID Conflicts

The previous section defines the types of SID conflicts and procedures to resolve such conflicts when using an MPLS dataplane. The mapping entry database used MUST be populated with entries for destinations for which the associated SID will be used to derive the labels installed in the forwarding plane of routers in the network. This consists of entries associated with intra-domain routes.

There are cases where destinations which are external to the domain are advertised by protocol speakers running within that network - and it is possible that those advertisements have SIDs associated with those destinations. However, if reachability to a destination is topologically outside the forwarding domain of the protocol instance then the SIDs for such destinations will never be installed in the forwarding plane of any router within the domain - so such advertisements cannot create a SID conflict within the domain. Such entries therefore MUST NOT be installed in the database used for intra-domain conflict resolution.

Consider the case of two sites "A and B" associated with a given [RFC4364] VPN. Connectivity between the sites is via a provider backbone. SIDs associated with destinations in Site A will never be installed in the forwarding plane of routers in Site B. Reachability between the sites (assuming SR is being used across the backbone) only requires using a SID associated with a gateway PE. So a destination in Site A MAY use the same SID as a destination in Site B without introducing any conflict in the forwarding plane of routers in Site A.

Such cases are handled by insuring that the mapping entries in the database used by the procedures defined in the previous section only include entries associated with advertisements within the site.

5. Conflict Resolution and non-forwarding nodes

The previous sections define conflict resolution behavior required of nodes which perform forwarding. But conflict resolution also impacts other entities e.g., controllers. If a controller were to define an explicit path using a SID in a way that is inconsistent with the set of Active entries produced by conflict resolution procedures used by the forwarding nodes then traffic following the explicit path may be misdelivered.

To prevent this such an entity MUST either implement the conflict resolution procedures defined above or implement an alternate form of conflict resolution which produces a subset of the Active entries which result from the conflict resolution procedures defined above. One such alternate form is to consider Inactive any mapping entry which has either a prefix conflict or a SID conflict with any other mapping entry.

6. Security Considerations

The ability to introduce SID conflicts into a deployment may compromise traffic forwarding. Protocol specific security mechanisms

SHOULD be used to insure that all SID advertisements originate from trusted sources.

7. IANA Consideration

This document has no actions for IANA.

8. Acknowledgements

The authors would like to thank Jeff Tantsura, Wim Henderickx, Bruno Decraene, and Stephane Litkowski for their careful review and content suggestions.

9. References

9.1. Normative References

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

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

[SR-BGP] "Segment Routing Prefix SID extensions for BGP, draft-ietf-idr-bgp-prefix-sid-06(work in progress)", June 2017.

[SR-IS-IS] "IS-IS Extensions for Segment Routing, draft-ietf-isis-segment-routing-extensions-13(work in progress)", June 2017.

[SR-MPLS] "Segment Routing with MPLS dataplane, draft-ietf-spring-segment-routing-mpls-10(work in progress)", June 2017.

[SR-OSPF] "OSPF Extensions for Segment Routing, draft-ietf-ospf-segment-routing-extensions-17(work in progress)", June 2017.

[SR-OSPFv3] "OSPFv3 Extensions for Segment Routing, draft-ietf-ospf-ospfv3-segment-routing-extensions-09(work in progress)", March 2017.

9.2. Informational References

[SR-ARCH] "Segment Routing Architecture, draft-ietf-spring-segment-routing-12(work in progress)", June 2017.

Appendix A. Alternative SID Conflict Resolution Policy Discussion

A number of approaches to resolving SID conflicts were considered during the writing of this document. Two general approaches with a total of three policy alternatives were considered. This Appendix documents the alternatives considered. All content in this section is non-normative.

Two general approaches can be used to process conflicting entries.

1. Conflicting entries can be ignored

2. A standard preference algorithm can be used to choose which of the conflicting entries will be used

The following sections discuss these two approaches in more detail.

A.1. Policy: Ignore conflicting entries

In cases where entries are in conflict none of the conflicting entries are used i.e., the network operates as if the conflicting advertisements were not present.

Implementations are required to identify the conflicting entries and ensure that they are not used.

A.2. Policy: Preference Algorithm/Quarantine

For entries which are in conflict properties of the conflicting advertisements are used to determine which of the conflicting entries are used in forwarding and which are "quarantined" and not used. Losing mapping entries with ranges greater than 1 are quarantined in their entirety.

This approach requires that conflicting entries first be identified and then evaluated based on a preference rule. Based on which entry is preferred this in turn may impact what other entries are considered in conflict i.e. if A conflicts with B and B conflicts with C - it is possible that A does NOT conflict with C. Hence if as a result of the evaluation of the conflict between A and B, entry B is not used the conflict between B and C will not be detected.

A.3. Policy: Preference algorithm/ignore overlap only

A variation of the preference algorithm approach when applied to mapping entries with ranges greater than 1 is to quarantine only the portions of the less preferred entry which actually conflict. The original entry is logically considered as a set of entries with a range of 1, each of which inherits the range value of the original entry for purposes of applying the preference rule.

A.4. Example Behavior - Single Topology/Address Family/Algorithm

The following mapping entries exist in the database. For brevity, Topology/Algorithm is omitted and assumed to be (0,0) in all entries.

1. (192, 192.0.2.1/32, 100, 1)

2. (192, 192.0.2.101/32, 200, 1)

3. (128, 192.0.2.1/32, 400, 255) !Prefix conflict with entries 1 and 2

4. (128, 198.51.100.40/32, 200,1) !SID conflict with entry 2

The table below shows what mapping entries will be used in the forwarding plane (Active) and which ones will not be used (Inactive) under the three candidate policies:

Policy	Active Entries	Inactive Entries
Ignore		(192,192.0.2.1/32,100,1) (192,192.0.2.101/32,200,1) (128,192.0.2.1/32,400,255) (128,198.51.100.40/32,200,1)
Quarantine	(192,192.0.1.1/32,100,1) (192,192.0.2.101/32,200,1)	(128,192.0.2.1/32,400,255) (128,198.51.100.40/32,200,1)
Ignore-Overlap-Only	(192,192.0.2.1/32,100,1) (192,192.0.2.101/32,200,1) *(128,192.0.2.2/32,401,99) *(128,192.0.2.102/32, 501,153)	(128,198.51.100.40/32,200,1) *(128,192.0.2.1/32,400,1) *(128,192.0.2.101/32,500,1)

* Derived from (128,192.0.2.1/32,400,300)

A.5. Evaluation of Policy Alternatives

The previous sections have defined three alternatives for resolving conflicts - ignore, quarantine, and ignore overlap-only.

The ignore policy impacts the greatest number of mapping entriesas all prefix/SID pairs contained in an advertisement which has a conflict are considered Inactive.

Quarantine allows forwarding for some destinations which have a conflict to be supported - but losing mapping entries with ranges greater than 1 are declared Inactive in their entirety. This may result in not using individual prefix/SID entries contained within the quarantined advertisement which do not have a conflict.

Ignore-overlap-only maximizes the entries which may be Active as each prefix/SID pair contained within an advertised mapping entry with range greater than 1 is evaluated independent of the other entries within the same advertisement. To implement this alternative advertised mapping entries with a range greater than 1 which have a conflict with other advertised mapping entries have to logically be split into 2 or more "derived mapping entries". The derived mapping entries then fall into two categories - those that are in conflict with other mapping entries and have lost based on the preference rule and those which are either NOT in conflict or have won based on the preference rule. The former are considered Inactive while the latter are considered Active. Each time the underived mapping database is updated the derived entries have to be recomputed based on the updated database. Internal data structures have to be maintained which maintain the relationship between the advertised mapping entry and the set of derived mapping entries. All nodes in the network have to achieve the same behavior regardless of implementation internals.

There is then a tradeoff between a goal of maximizing advertised mapping entry usage and the risks associated with increased implementation complexity.

Consensus of the working group is that maximizing the use of the advertised prefix/SID pairs is the most important deployment consideration - therefore ignore-overlap-only has been specified as the standard policy which MUST be implemented by all nodes which support SR-MPLS.

Authors' Addresses

Les Ginsberg
Cisco Systems
821 Alder Drive
Milpitas, CA 95035
USA

Email: ginsberg@cisco.com

Peter Psenak
Cisco Systems
Apollo Business Center Mlynske nivy 43
Bratislava 821 09
Slovakia

Email: ppsenak@cisco.com

Stefano Previdi
Cisco Systems

Email: stefano@previdi.net

Martin Pilka

Email: martin@infobox.sk

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2017

S. Litkowski
Orange Business Service
Y. Qu
Cisco Systems
P. Sarkar
J. Tantsura
Individual

October 28, 2016

YANG Data Model for Segment Routing
draft-ietf-spring-sr-yang-05

Abstract

This document defines a YANG data model ([RFC6020], [RFC7950]) for segment routing ([I-D.ietf-spring-segment-routing]) configuration and operation. This YANG model is intended to be used on network elements to configure or operate segment routing. This document defines also generic containers that SHOULD be reused by IGP protocol modules to support segment routing.

Requirements Language

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at 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 May 1, 2017.

Table of Contents

1. Introduction

This document defines a YANG data model for segment routing configuration and operation. This document does not define the IGP extensions to support segment routing but defines generic groupings that SHOULD be reused by IGP extension modules. The reason of this design choice is to not require implementations to support all IGP extensions. For example, an implementation may support IS-IS extension but not OSPF.

1.1. Tree diagram

A simplified graphical representation of the data model is presented in Section 2.

The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Design of the Data Model

As the module definition is just starting, it is expected that there will be changes as the module matures.

```
module: ietf-segment-routing
augment /rt:routing:
   +--rw segment-routing
      +--rw transport-type?   identityref
      +--rw msd {msd}?
      |  +--rw node-msd?   uint8
      |  +--rw link-msd
      |     +--rw link-msds* [interface]
      |        +--rw interface    if:interface-ref
      |        +--rw msd?         uint8
      +--rw bindings
      |  +--rw mapping-server {mapping-server}?
      |  |  +--rw policy* [name]
      |  |     +--rw name    string
      |  |     +--rw ipv4
      |  |     |  +--rw mapping-entry* [prefix algorithm]
      |  |     |     +--rw prefix        inet:ipv4-prefix
      |  |     |     +--rw value-type?   enumeration
      |  |     |     +--rw start-sid     uint32
```

```
      |  |     |     +--rw range?        uint32
      |  |     |     +--rw algorithm     identityref
      |  |     +--rw ipv6
      |  |        +--rw mapping-entry* [prefix algorithm]
      |  |           +--rw prefix        inet:ipv6-prefix
      |  |           +--rw value-type?   enumeration
      |  |           +--rw start-sid     uint32
      |  |           +--rw range?        uint32
      |  |           +--rw algorithm     identityref
      |  +--rw connected-prefix-sid-map
      |     +--rw ipv4
      |     |  +--rw ipv4-prefix-sid* [prefix algorithm]
      |     |     +--rw prefix               inet:ipv4-prefix
      |     |     +--rw value-type?          enumeration
      |     |     +--rw start-sid            uint32
      |     |     +--rw range?               uint32
      |     |     +--rw algorithm            identityref
      |     |     +--rw last-hop-behavior?   enumeration {sid-last-hop-behavior}
?
      |     +--rw ipv6
      |        +--rw ipv6-prefix-sid* [prefix algorithm]
      |           +--rw prefix               inet:ipv6-prefix
      |           +--rw value-type?          enumeration
      |           +--rw start-sid            uint32
      |           +--rw range?               uint32
      |           +--rw algorithm            identityref
      |           +--rw last-hop-behavior?   enumeration {sid-last-hop-behavior}
?
      +--rw global-srgb
         +--rw srgb* [lower-bound upper-bound]
            +--rw lower-bound    uint32
            +--rw upper-bound    uint32
augment /rt:routing-state:
   +--ro segment-routing
      +--ro node-capabilities
      |  +--ro transport-planes* [transport-plane]
      |  |  +--ro transport-plane    identityref
      |  +--ro readable-label-stack-depth?   uint8
      +--ro msd
      |  +--ro node-msd?   uint8
      |  +--ro link-msd
      |     +--ro link-msds* [interface]
      |        +--ro interface    if:interface-ref
      |        +--ro msd?         uint8
      +--ro label-blocks*
      |  +--ro lower-bound?   uint32
      |  +--ro upper-bound?   uint32
      |  +--ro size?          uint32
      |  +--ro free?          uint32
      |  +--ro used?          uint32
```

```
      +--ro global-sid-list
         +--ro sid* [target sid source source-protocol binding-type]
            +--ro target             string
            +--ro sid                uint32
            +--ro algorithm?         uint8
            +--ro source             inet:ip-address
            +--ro used?              boolean
            +--ro source-protocol    -> /rt:routing-state/control-plane-protocol
s
                                        /control-plane-protocol/name
            +--ro binding-type       enumeration
notifications:
   +---n segment-routing-global-srgb-collision
   |  +--ro srgb-collisions*
   |     +--ro lower-bound?          uint32
   |     +--ro upper-bound?          uint32
   |     +--ro routing-protocol?     -> /rt:routing-state/control-plane-protocol
s
   |                                    /control-plane-protocol/name
   |     +--ro originating-rtr-id?   router-id
   +---n segment-routing-global-sid-collision
   |  +--ro received-target?       string
   |  +--ro new-sid-rtr-id?        router-id
   |  +--ro original-target?       string
   |  +--ro original-sid-rtr-id?   router-id
   |  +--ro index?                 uint32
   |  +--ro routing-protocol?      -> /rt:routing-state/control-plane-protocols
   |                                  /control-plane-protocol/name
   +---n segment-routing-index-out-of-range
      +--ro received-target?    string
      +--ro received-index?     uint32
      +--ro routing-protocol?   -> /rt:routing-state/control-plane-protocols
                                   /control-plane-protocol/name
```

3. Configuration

This module augments the "/rt:routing:" with a segment-routing container. This container defines all the configuration parameters related to segment-routing.

The segment-routing configuration is split in global configuration and interface configuration.

The global configuration includes :

- segment-routing transport type : The underlying transport type for segment routing. The version of the model limits the transport type to an MPLS dataplane. The transport-type is only defined once for a particular routing-instance and is agnostic to the

control plane used. Only a single transport-type is supported in this version of the model.

- o bindings : Defines prefix to SID mappings. The operator can control advertisement of Prefix-SID independently for IPv4 and IPv6. Two types of mappings are available :
  - * Mapping-server : maps non local prefixes to a segment ID. Configuration of bindings does not automatically allow advertisement of those bindings. Advertisement must be controlled by each routing-protocol instance (see Section 4). Multiple mapping policies may be defined.
  - * Connected prefixes : maps connected prefixes to a segment ID. Advertisement of the mapping will be done by IGP when enabled for segment routing (see Section 4). The SID value can be expressed as an index (default), or an absolute value. The "last-hop-behavior" configuration dictates the PHP behavior: "explicit-null", "php", or "non-php".
- o SRGB (Segment Routing Global Block): Defines a list of label blocks represented by a pair of lower-bound/upper-bound labels. The SRGB is also agnostic to the control plane used. So all routing-protocol instance will have to advertise the same SRGB.

4. IGP Control plane configuration

Support of segment-routing extensions for a particular IGP control plane is done by augmenting routing-protocol configuration with segment-routing extensions. This augmentation SHOULD be part of separate YANG modules in order to not create any dependency for implementations to support all protocol extensions.

This module defines groupings that SHOULD be used by IGP segment routing modules.

The "controlplane-cfg" grouping defines the generic global configuration for the IGP.

The "enabled" leaf enables segment-routing extensions for the routing-protocol instance.

The "bindings" container controls the routing-protocol instance’s advertisement of local bindings and the processing of received bindings.

4.1. IGP interface configuration

The interface configuration is part of the "igp-interface-cfg" grouping and includes Adjacency SID properties.

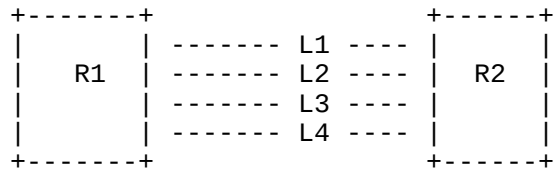
4.1.1. Adjacency SID properties

4.1.1.1. Bundling

This section is a first proposal on how to use S-bit in Adj-SID to create bundles. Authors would like to trigger discussion based on this first proposal.

In case of parallel IP links between routers, an additional Adjacency SID may be advertised representing more than one adjacency (i.e., a bundle of adjacencies). The "advertise-adj-group-sid" configuration controls whether or not an additional adjacency SID is advertised.

The "advertise-adj-group-sid" would be a list of "group-id". The "group-id" will permit to identify interfaces that must be bundled together.



In the figure above, R1 and R2 are interconnected by four links. A routing protocol adjacency is established on each link. Operator would like to create segment-routing Adj-SID that represent some bundles of links. We can imagine two different bundles : L1/L2 and L2/L3. To achieve this behavior, the service provider will configure a "group-id" X for both interfaces L1 and L2 and a "group-id" Y for both interfaces L3 and L3. This will result in R1 advertising an additional Adj-SID for each adjacency, for example a Adj-SID with S flag set and value of 400 will be added to L1 and L2. A Adj-SID with S flag set and value of 500 will be added to L3 and L4. As L1/L2 and L3/L4 does not share the same "group-id", a different SID value will be allocated.

4.1.1.2. Protection

The "advertise-protection" defines how protection for an interface is advertised. It does not control the activation or deactivation of protection. If the "single" option is used, a single Adj-SID will be

advertised for the interface. If the interface is protected, the B-Flag for the Adj-SID advertisement will be set. If the "dual" option is used and if the interface is protected, two Adj-SIDs will be advertised for the interface adjacencies. One Adj-SID will always have the B-Flag set and the other will have the B-Flag clear. This option is intended to be used in the case of traffic engineering where a path must use either protected segments or non-protected segments.

5. States

The operational states contains information reflecting the usage of allocated SRGB labels.

It also includes a list of all global SIDs, their associated bindings, and other information such as the source protocol and algorithm.

6. Notifications

The model defines the following notifications for segment-routing.

- segment-routing-global-srgb-collision: Rasied when a control plan advertised SRGB blocks have conflicts.
- segment-routing-global-sid-collision: Raised when a control plane advertised index is already associated with another target (in this version, the only defined targets are IPv4 and IPv6 prefixes).
- segment-routing-index-out-of-range: Raised when a control plane advertised index fall outside the range of SRGBs configured for the network device.

7. YANG Module

```
<CODE BEGINS> file "ietf-segment-routing-common@2016-10-28.yang"
module ietf-segment-routing-common {
  namespace "urn:ietf:params:xml:ns:"
          + "yang:ietf-segment-routing-common";
  prefix sr-cmn;

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

  organization
   "IETF SPRING Working Group";
```

```
  contact
    "WG List:  <mailto:spring@ietf.org>

    Editor:    Stephane Litkowski
               <mailto:stephane.litkowski@orange.com>

    Author:    Acee Lindem
               <mailto:acee@cisco.com>
    Author:    Yingzhen Qu
               <mailto:yiqu@cisco.com>
    Author:    Pushpasis Sarkar
               <mailto:pushpasis.ietf@gmail.com>
    Author:    Jeff Tantsura
               <jefftant.ietf@gmail.com>

    ";

  description
    "The YANG module defines a collection of types and groupings for
    Segment routing.";

  revision 2016-10-28 {
    description "
      * Add support of MSD (Maximum SID Depth)
      * Update contact info
    ";
    reference
      "RFC XXXX: YANG Data Model for Segment Routing.";
  }

  revision 2016-10-24 {
    description "Initial";
    reference
      "RFC XXXX: YANG Data Model for Segment Routing.";
  }

  /* Identities */
  identity segment-routing-transport {
    description
    "Base identity for segment routing transport.";
  }
  identity segment-routing-transport-mpls {
    base segment-routing-transport;
    description
    "This identity represents MPLS transport for segment
    routing.";
  }
```

```
  identity prefix-sid-algorithm {
    description
    "Base identity for prefix-sid algorithm.";
  }

  identity prefix-sid-algorithm-shortest-path {
    base prefix-sid-algorithm;
    description
    "The default behavior of prefix-sid algorithm.";
  }

  identity prefix-sid-algorithm-strict-spf {
    base prefix-sid-algorithm;
    description
    "This algorithm mandates that the packet is forwared
     according to ECMP-aware SPF algorithm.";
  }

  /* Features */

  feature sid-last-hop-behavior {
    description
      "Configurable last hop behavior.";
  }

  /* Groupings */

  grouping srgb-cfg {
    list srgb {
      key "lower-bound upper-bound";
      ordered-by user;
      leaf lower-bound {
        type uint32;
        description
          "Lower value in the block.";
      }
      leaf upper-bound {
        type uint32;
        description
          "Upper value in the block.";
      }
      description
        "List of global blocks to be
         advertised.";
    }
    description
      "Grouping for SRGB configuration.";
  }
```

```
  grouping sid-value-type {
    leaf value-type {
      type enumeration {
        enum index {
          description
            "The value will be
             interpreted as an index.";
        }
        enum absolute {
          description
            "The value will become
             interpreted as an absolute
             value.";
        }
      }
      default index;
      description
        "This leaf defines how value
         must be interpreted.";
    }
        description
         "Defines how the SID value is expressed.";
  }

  grouping ipv4-sid-cfg {

    leaf prefix {
      type inet:ipv4-prefix;
      description
        "connected prefix sid.";
    }

    uses prefix-sid-attributes;

    description
     "This grouping defines cfg of prefix SID.";
  }

  grouping ipv6-sid-cfg {
    leaf prefix {
      type inet:ipv6-prefix;
      description
        "connected prefix sid.";
    }

    uses prefix-sid-attributes;

    description
```

```
     "This grouping defines cfg of prefix SID.";
  }

  grouping last-hop-behavior {
    leaf last-hop-behavior {
      if-feature sid-last-hop-behavior;
      type enumeration {
        enum explicit-null {
          description
            "Use explicit-null for the SID.";
        }
        enum no-php {
          description
            "Do no use PHP for the SID.";
        }
        enum php {
            description
             "Use PHP for the SID.";
        }
      }
      description
        "Configure last hop behavior.";
    }
        description
         "Defines last hop behavior";
  }

  grouping node-capabilities {
    description "Containing SR node capabilities.";
    container node-capabilities {
      list transport-planes {
        key transport-plane;
        leaf transport-plane {
          type identityref {
            base segment-routing-transport;
          }
          description
            "Transport plane supported";
        }
        description
          "List of supported transport planes.";
      }
      leaf readable-label-stack-depth {
        type uint8;
        description
          "Number of MPLS labels that
           can be read in the stack.";
      }
```

```
      description
        "Shows the SR capability of the node.";
    } // node-capabilities
  } // sr-node-capabilities

  grouping prefix-sid-attributes {
    description "Containing SR attributes for a prefix.";

    uses sid-value-type;
    leaf start-sid {
      type uint32;
      mandatory true;
      description
        "Value associated with
         prefix. The value must
         be interpreted in the
         context of value-type.";
    }

    leaf range {
      type uint32;
      description
        "Describes how many SIDs could be
         allocated.";
    }

    leaf algorithm {
      type identityref {
        base prefix-sid-algorithm;
      }
      description "Prefix-sid algorithm.";
    }
  } //prefix-sid-attributes
}
<CODE ENDS>
<CODE BEGINS> file "ietf-segment-routing@2016-10-28.yang"
module ietf-segment-routing {
  namespace "urn:ietf:params:xml:ns:"
          + "yang:ietf-segment-routing";
  prefix sr;

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

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

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

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-segment-routing-common {
    prefix "sr-cmn";
  }

  organization
   "IETF SPRING Working Group";

  contact
    "WG List:  <mailto:spring@ietf.org>

    Editor:    Stephane Litkowski
               <mailto:stephane.litkowski@orange.com>

    Author:    Acee Lindem
               <mailto:acee@cisco.com>
    Author:    Yingzhen Qu
               <mailto:yiqu@cisco.com>
    Author:    Pushpasis Sarkar
               <mailto:pushpasis.ietf@gmail.com>
    Author:    Jeff Tantsura
               <jefftant.ietf@gmail.com>

    ";

  description
    "The YANG module defines a generic configuration model for
    Segment routing common across all of the vendor
    implementations.";

  revision 2016-10-28 {
    description "
      * Add support of MSD (Maximum SID Depth)
      * Update contact info
    ";
    reference
      "RFC XXXX: YANG Data Model for Segment Routing.";
  }

  revision 2016-10-24 {
    description "
```

```
      * Moved common SR types and groupings to a seperate module
    ";
    reference
      "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-07-07 {
    description "
      * Add support of prefix-sid algorithm configuration
      * change routing-protocols to control-plane-protocols
    ";
    reference
      "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2016-03-17 {
    description "
      * Add notification segment-routing-global-srgb-collision
      * Add router-id to segment-routing-global-sid-collision
      * Remove routing-instance
      * Add typedef router-id
    ";
    reference
      "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2015-10-17 {
    description "
      * Add per-protocol SRGB config feature
      * Move SRBG config to a grouping
    ";
    reference
      "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  revision 2015-06-22 {
    description "
      * Prefix SID config moved to
        connected-prefix-sid-map in global SR cfg
        rather than IGP.
      ";
      reference "draft-litkowski-spring-sr-yang-01";
  }
  revision 2015-04-23 {
    description "
      * Node flag deprecated from prefixSID
      * SR interface cfg moved to protocol
      * Adding multiple binding policies for SRMS
      ";
      reference "";
  }
  revision 2015-02-27 {
```

```
    description "Initial";
    reference "draft-litkowski-spring-sr-yang-00";
  }

  /* Features */

  feature mapping-server {
    description
     "Support of SRMS.";
  }

  feature protocol-srgb {
    description
      "Support per-protocol srgb configuration.";
  }

  feature msd {
    description
      "Support of signaling MSD (Maximum SID Depth) in IGP.";
  }

  /* Type Definitions */

  typedef system-id {
    type string {
      pattern
       '[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.00';
    }
    description
      "This type defines ISIS system id using pattern,
       system id looks like : 0143.0438.AeF0.00";
  }

  typedef router-id {
    type union {
      type system-id;
      type yang:dotted-quad;
    }
    description
      "OSPF/BGP router id or ISIS system ID.";
  }

  /* Groupings */

  grouping controlplane-cfg {
    container segment-routing {
      leaf enabled {
        type boolean;
```

```
        default false;
        description
          "Enables segment-routing
           protocol extensions.";
      }
      container bindings {
        container advertise {
          leaf-list policies {
            type string;
            description
              "List of policies to be advertised.";
          }
          description
            "Authorize the advertise
             of local mappings in binding TLV.";
        }
        leaf receive {
          type boolean;
          default true;
          description
            "Authorize the reception and usage
            of binding TLV.";
        }
        description
          "Control of binding advertisement
           and reception.";
      }

      description
        "segment routing global config.";
    }
    description
      "Defines protocol configuration.";
  }

  grouping igp-interface-cfg {
    container segment-routing {

      container adjacency-sid {
        list advertise-adj-group-sid {
          key group-id;

          leaf group-id {
            type uint32;
            description
              "The value is an internal value to identify
               a group-ID. Interfaces with the same
               group-ID will be bundled together.";
```

```
          }
          description
            "Control advertisement of S flag.
             Enable to advertise a common Adj-SID
             for parallel links.";
        }
        leaf advertise-protection {
          type enumeration {
            enum "single" {
              description
                "A single Adj-SID is associated
                 with the adjacency and reflects
                 the protection configuration.";
            }
            enum "dual" {
              description
                "Two Adj-SIDs will be associated
                 with the adjacency if interface
                 is protected. In this case
                 one will be enforced with
                 backup flag set, the other
                 will be enforced to backup flag unset.
                 In case, protection is not configured,
                 a single Adj-SID will be advertised
                 with backup flag unset.";
            }
          }
          description
            "If set, the Adj-SID refers to an
             adjacency being protected.";
        }
        description
          "Defines the adjacency SID properties.";
      }
      description
        "container for SR interface cfg.";
    }
    description
      "Grouping for IGP interface cfg.";
  }

  grouping msd-cfg {
    leaf node-msd {
      type uint8;
      description
        "Node MSD is the lowest MSD supported by the node.";
    }
```

```
    container link-msd {
      list link-msds {
        key interface;
        leaf interface {
          type if:interface-ref;
          description
            "Name of the interface.";
        }
        leaf msd {
          type uint8;
          description
            "SID depth of the interface associated with the link.";
        }
        description
          "List of link MSDs.";
      }
      description
        "Link MSD is a number represetns the particular link MSD value.";
    }
    description
      "MSD configuration grouping.";
  }

  /* Cfg */

  augment "/rt:routing" {
    description
      "This augments routing-instance
       configuration with segment-routing.";
    container segment-routing {
      leaf transport-type {
        type identityref {
          base sr-cmn:segment-routing-transport;
        }
        default "sr-cmn:segment-routing-transport-mpls";
        description "Dataplane to be used.";
      }
      container msd {
        if-feature msd;
        uses msd-cfg;
        description
          "MSD configuration.";
      }
      container bindings {
        container mapping-server {
          if-feature mapping-server;
          list policy {
            key name;
```

```
            leaf name {
              type string;
              description
                "Name of the mapping policy.";
            }
            container ipv4 {
              list mapping-entry {
                key "prefix algorithm";
                uses sr-cmn:ipv4-sid-cfg;

                description
                  "Mapping entries.";
              }
              description
                "IPv4 mapping entries.";
            }
            container ipv6 {
              list mapping-entry {
                key "prefix algorithm";
                uses sr-cmn:ipv6-sid-cfg;
                description
                  "Mapping entries.";
              }
              description
                "IPv6 mapping entries.";
            }
            description
              "Definition of mapping policy.";
          }
          description
            "Configuration of mapping-server
             local entries.";
        }
        container connected-prefix-sid-map {
          container ipv4 {
            list ipv4-prefix-sid {
              key "prefix algorithm";
              uses sr-cmn:ipv4-sid-cfg;
              uses sr-cmn:last-hop-behavior;
              description
                "List of prefix SID
                 mapped to IPv4 local prefixes.";
            }
            description
              "Parameters associated with IPv4 prefix SID";
          }
          container ipv6 {
            list ipv6-prefix-sid {
```

```
              key "prefix algorithm";
              uses sr-cmn:ipv6-sid-cfg;
              uses sr-cmn:last-hop-behavior;
              description
                "List of prefix SID
                 mapped to IPv6 local prefixes.";
            }
            description
              "Parameters associated with IPv6 prefix SID";
          }
          description
            "Prefix SID configuration.";
        }
        description
          "List of bindings.";
      }
      container global-srgb {
        uses sr-cmn:srgb-cfg;
        description
          "Global SRGB configuration.";
      }
      description
        "segment routing global config.";
    }
  }


  /* Operational states */

  augment "/rt:routing-state" {
    description
     "This augments the operational states
      with segment-routing.";
    container segment-routing {
      uses sr-cmn:node-capabilities;
      container msd {
        uses msd-cfg;
        description
          "MSD configuration state.";
      }
      list label-blocks {
        leaf lower-bound {
          type uint32;
          description
           "Lower bound of the label block.";
        }
        leaf upper-bound {
          type uint32;
```

```
          description
           "Upper bound of the label block.";
        }
        leaf size {
          type uint32;
          description
           "Number of indexes in the block.";
        }
        leaf free {
          type uint32;
          description
           "Number of indexes free in the block.";
        }
        leaf used {
          type uint32;
          description
           "Number of indexes used in the block.";
        }
        description
         "List of labels blocks currently
         in use.";
      }

      container global-sid-list {
        list sid {
          key "target sid source source-protocol binding-type";
          ordered-by system;
          leaf target {
            type string;
            description
             "Defines the target of the binding.
             It can be a prefix or something else.";
          }
          leaf sid {
            type uint32;
            description
             "Index associated with the prefix.";
          }
          leaf algorithm {
            type uint8;
            description
             "Algorithm to be used for the prefix
             SID.";
          }
          leaf source {
            type inet:ip-address;
            description
             "IP address of the router than own
```

```
              the binding.";
          }
          leaf used {
            type boolean;
            description
             "Defines if the binding is used
              in forwarding plane.";
          }
          leaf source-protocol {
            type leafref {
              path "/rt:routing-state/rt:control-plane-protocols/"
                 + "rt:control-plane-protocol/rt:name";
            }
            description
             "Rtg protocol that owns the binding";
          }
          leaf binding-type {
            type enumeration {
              enum prefix-sid {
                description
                 "Binding is learned from
                 a prefix SID.";
              }
              enum binding-tlv {
                description
                 "Binding is learned from
                 a binding TLV.";
              }
            }
            description
             "Type of binding.";
          }
          description
            "Binding.";

        }
        description
         "List of prefix and SID associations.";
      }
      description
       "Segment routing operational states.";
    }
  }


  /* Notifications */

  notification segment-routing-global-srgb-collision {
```

```
    list srgb-collisions {
      leaf lower-bound {
        type uint32;
          description
            "Lower value in the block.";
      }
      leaf upper-bound {
        type uint32;
          description
            "Upper value in the block.";
      }
      leaf routing-protocol {
        type leafref {
          path "/rt:routing-state/rt:control-plane-protocols/"
             + "rt:control-plane-protocol/rt:name";
        }
        description
          "Routing protocol reference that received the event.";
      }
      leaf originating-rtr-id {
        type router-id;
        description
          "Originating router id of this SRGB block.";
      }
      description
        "List of SRGB blocks that conflict.";
    }
    description
      "This notification is sent when received SRGB blocks from
       a router conflict.";
  }
  notification segment-routing-global-sid-collision {
    leaf received-target {
      type string;
      description
      "Target received in the controlplane that
                          caused SID collision.";
    }
    leaf new-sid-rtr-id {
      type router-id;
      description
        "Router Id that advertising the conflicting SID.";
    }
    leaf original-target {
      type string;
      description
      "Target already available in database that have the same SID
      as the received target.";
```

```
    }
    leaf original-sid-rtr-id {
      type router-id;
      description
        "Original router ID that advertised the conflicting SID.";
    }
    leaf index {
      type uint32;
      description
      "Value of the index used by two different prefixes.";
    }
    leaf routing-protocol {
      type leafref {
        path "/rt:routing-state/rt:control-plane-protocols/"
           + "rt:control-plane-protocol/rt:name";
      }
      description
       "Routing protocol reference that received the event.";
    }
    description
      "This notification is sent when a new mapping is learned
      , containing mapping
      where the SID is already used.
      The notification generation must be throttled with at least
       a 5 second gap. ";
  }
  notification segment-routing-index-out-of-range {
    leaf received-target {
      type string;
      description
      "Target received in the controlplane
                        that caused SID collision.";
    }
    leaf received-index {
      type uint32;
      description
      "Value of the index received.";
    }
    leaf routing-protocol {
      type leafref {
        path "/rt:routing-state/rt:control-plane-protocols/" +
        "rt:control-plane-protocol/rt:name";
      }
      description
       "Routing protocol reference that received the event.";
    }
    description
      "This notification is sent when a binding
```

```
      is received, containing a segment index
      which is out of the local configured ranges.
      The notification generation must be throttled with at least
       a 5 second gap. ";
  }
}
<CODE ENDS>
```

8. Security Considerations

TBD.

9. Acknowledgements

Authors would like to thank Derek Yeung, Acee Lindem, Greg Hankins, Hannes Gredler, Uma Chunduri, Jeffrey Zhang, Shradda Hedge for their contributions.

10. IANA Considerations

TBD.

11. Normative References

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-09 (work in progress), July 2016.

[I-D.tantsura-isis-segment-routing-msd]
Tantsura, J. and U. Chunduri, "Signaling MSD (Maximum SID Depth) using IS-IS", draft-tantsura-isis-segment-routing-msd-02 (work in progress), September 2016.

[I-D.tantsura-ospf-segment-routing-msd]
Tantsura, J. and U. Chunduri, "Signaling MSD (Maximum SID Depth) using OSPF", draft-tantsura-ospf-segment-routing-msd-01 (work in progress), September 2016.

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

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.

Authors' Addresses

Stephane Litkowski
Orange Business Service

Email: stephane.litkowski@orange.com

Yingzhen Qu
Cisco Systems

Email: yiqu@cisco.com

Pushpasis Sarkar
Individual

Email: pushpasis.ietf@gmail.com

Jeff Tantsura
Individual

Email: jefftant.ietf@gmail.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2020

S. Litkowski
Orange Business Service
Y. Qu
Futurewei
A. Lindem
Cisco
P. Sarkar
Individual
J. Tantsura
Apstra

July 7, 2019

YANG Data Model for Segment Routing
draft-ietf-spring-sr-yang-13

Abstract

This document defines a YANG data model ([RFC6020], [RFC7950]) for segment routing ([RFC8402]) configuration and operation. This YANG model is intended to be used on network elements to configure or operate segment routing. This document defines also generic containers that SHOULD be reused by IGP protocol modules to support segment routing.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 8, 2020.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG data model for segment routing configuration and operation. This document does not define the IGP extensions to support segment routing but defines generic groupings that SHOULD be reused by IGP extension modules. The reason of this design choice is to not require implementations to support all IGP extensions. For example, an implementation may support IS-IS extension but not OSPF.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Terminology and Notation

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

2.1. Tree diagram

Tree diagrams used in this document follow the notation defined in [RFC8340].

2.2. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
if	ietf-interfaces	[RFC8343]
rt	ietf-routing	[RFC8349]
rt-types	ietf-routing-types	[RFC8294]
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and Corresponding YANG Modules

3. Design of the Data Model

As the module definition is just starting, it is expected that there will be changes as the module matures.

```
module: ietf-segment-routing
  augment /rt:routing:
    +--rw segment-routing
       +--rw transport-type?      identityref
       +--ro node-capabilities
       |  +--ro transport-planes* [transport-plane]
       |  |  +--ro transport-plane    identityref
       |  +--ro entropy-readable-label-depth?   uint8
       +--rw msd {max-sid-depth}?
       |  +--rw node-msd?   uint8
       |  +--rw link-msd
       |     +--rw link-msds* [interface]
```

```
|        +--rw interface    if:interface-ref
|        +--rw msd?         uint8
+--rw bindings
|  +--rw mapping-server {mapping-server}?
|  |  +--rw policy* [name]
|  |     +--rw name       string
|  |     +--rw entries
|  |        +--rw mapping-entry* [prefix algorithm]
|  |           +--rw prefix        inet:ip-prefix
|  |           +--rw value-type?   enumeration
|  |           +--rw start-sid     uint32
|  |           +--rw range?        uint32
|  |           +--rw algorithm     identityref
|  +--rw connected-prefix-sid-map
|  |  +--rw connected-prefix-sid* [prefix algorithm]
|  |     +--rw prefix               inet:ip-prefix
|  |     +--rw value-type?          enumeration
|  |     +--rw start-sid            uint32
|  |     +--rw range?               uint32
|  |     +--rw algorithm            identityref
|  |     +--rw last-hop-behavior?   enumeration
|  |                                {sid-last-hop-behavior}?
|  +--rw local-prefix-sid
|     +--rw local-prefix-sid* [prefix algorithm]
|        +--rw prefix        inet:ip-prefix
|        +--rw value-type?   enumeration
|        +--rw start-sid     uint32
|        +--rw range?        uint32
|        +--rw algorithm     identityref
+--rw global-srgb
|  +--rw srgb* [lower-bound upper-bound]
|     +--rw lower-bound    uint32
|     +--rw upper-bound    uint32
+--rw srlb
|  +--rw srlb* [lower-bound upper-bound]
|     +--rw lower-bound    uint32
|     +--rw upper-bound    uint32
+--ro label-blocks*
|  +--ro lower-bound?   uint32
|  +--ro upper-bound?   uint32
|  +--ro size?          uint32
|  +--ro free?          uint32
|  +--ro used?          uint32
|  +--ro scope?         enumeration
+--ro sid-list
   +--ro sid* [target sid source source-protocol binding-type]
      +--ro target             string
      +--ro sid                uint32
```

```
               +--ro algorithm?         uint8
               +--ro source             inet:ip-address
               +--ro used?              boolean
               +--ro source-protocol    -> /rt:routing
                                           /control-plane-protocols
                                           /control-plane-protocol/name
               +--ro binding-type       enumeration
               +--ro scope?             enumeration

     notifications:
       +---n segment-routing-global-srgb-collision
       |  +--ro srgb-collisions*
       |     +--ro lower-bound?          uint32
       |     +--ro upper-bound?          uint32
       |     +--ro routing-protocol?     -> /rt:routing
       |                                    /control-plane-protocols
       |                                    /control-plane-protocol/name
       |     +--ro originating-rtr-id?   router-id
       +---n segment-routing-global-sid-collision
       |  +--ro received-target?       string
       |  +--ro new-sid-rtr-id?        router-id
       |  +--ro original-target?       string
       |  +--ro original-sid-rtr-id?   router-id
       |  +--ro index?                 uint32
       |  +--ro routing-protocol?      -> /rt:routing
       |                                  /control-plane-protocols
       |                                  /control-plane-protocol/name
       +---n segment-routing-index-out-of-range
          +--ro received-target?    string
          +--ro received-index?     uint32
          +--ro routing-protocol?   -> /rt:routing
                                       /control-plane-protocols
                                       /control-plane-protocol/name
```

4. Configuration

This module augments the "/rt:routing:" with a segment-routing container. This container defines all the configuration parameters related to segment-routing.

The segment-routing configuration is split in global configuration and interface configuration.

The global configuration includes :

- segment-routing transport type : The underlying transport type for segment routing. The version of the model limits the transport

type to an MPLS dataplane. The transport-type is only defined once for a particular routing-instance and is agnostic to the control plane used. Only a single transport-type is supported in this version of the model.

- bindings : Defines prefix to SID mappings. The operator can control advertisement of Prefix-SID independently for IPv4 and IPv6. Two types of mappings are available :
  - Mapping-server : maps non local prefixes to a segment ID. Configuration of bindings does not automatically allow advertisement of those bindings. Advertisement must be controlled by each routing-protocol instance (see Section 5). Multiple mapping policies may be defined.
  - Connected prefixes : maps connected prefixes to a segment ID. Advertisement of the mapping will be done by IGP when enabled for segment routing (see Section 5). The SID value can be expressed as an index (default), or an absolute value. The "last-hop-behavior" configuration dictates the PHP behavior: "explicit-null", "php", or "non-php".
- SRGB (Segment Routing Global Block): Defines a list of label blocks represented by a pair of lower-bound/upper-bound labels. The SRGB is also agnostic to the control plane used. So all routing-protocol instance will have to advertise the same SRGB.
- SRLB (Segment Routing Local Block): Defines a list of label blocks represented by a pair of lower-bound/upper-bound labels, reserved for lcoal SIDs.

5. IGP Control plane configuration

Support of segment-routing extensions for a particular IGP control plane is done by augmenting routing-protocol configuration with segment-routing extensions. This augmentation SHOULD be part of separate YANG modules in order to not create any dependency for implementations to support all protocol extensions.

This module defines groupings that SHOULD be used by IGP segment routing modules.

The "controlplane-cfg" grouping defines the generic global configuration for the IGP.

The "enabled" leaf enables segment-routing extensions for the routing-protocol instance.

The "bindings" container controls the routing-protocol instance's advertisement of local bindings and the processing of received bindings.

5.1. IGP interface configuration

The interface configuration is part of the "igp-interface-cfg" grouping and includes Adjacency SID properties.

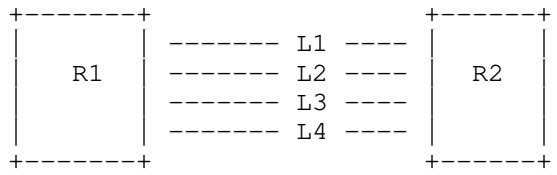
5.1.1. Adjacency SID properties

5.1.1.1. Bundling

This section is a first proposal on how to use S-bit in Adj-SID to create bundles. Authors would like to trigger discussion based on this first proposal.

In case of parallel IP links between routers, an additional Adjacency SID may be advertised representing more than one adjacency (i.e., a bundle of adjacencies). The "advertise-adj-group-sid" configuration controls whether or not an additional adjacency SID is advertised.

The "advertise-adj-group-sid" would be a list of "group-id". The "group-id" will permit to identify interfaces that must be bundled together.



In the figure above, R1 and R2 are interconnected by four links. A routing protocol adjacency is established on each link. Operator would like to create segment-routing Adj-SID that represent some bundles of links. We can imagine two different bundles : L1/L2 and L2/L3. To achieve this behavior, the service provider will configure a "group-id" X for both interfaces L1 and L2 and a "group-id" Y for both interfaces L3 and L3. This will result in R1 advertising an additional Adj-SID for each adjacency, for example a Adj-SID with S flag set and value of 400 will be added to L1 and L2. A Adj-SID with S flag set and value of 500 will be added to L3 and L4. As L1/L2 and L3/L4 does not share the same "group-id", a different SID value will be allocated.

5.1.1.2. Protection

The "advertise-protection" defines how protection for an interface is advertised. It does not control the activation or deactivation of protection. If the "single" option is used, a single Adj-SID will be advertised for the interface. If the interface is protected, the B-Flag for the Adj-SID advertisement will be set. If the "dual" option is used and if the interface is protected, two Adj-SIDs will be advertised for the interface adjacencies. One Adj-SID will always have the B-Flag set and the other will have the B-Flag clear. This option is intended to be used in the case of traffic engineering where a path must use either protected segments or non-protected segments.

6. States

The operational states contains information reflecting the usage of allocated SRGB labels.

It also includes a list of all global SIDs, their associated bindings, and other information such as the source protocol and algorithm.

7. Notifications

The model defines the following notifications for segment-routing.

- segment-routing-global-srgb-collision: Rasied when a control plan advertised SRGB blocks have conflicts.

- segment-routing-global-sid-collision: Raised when a control plane advertised index is already associated with another target (in this version, the only defined targets are IPv4 and IPv6 prefixes).

- segment-routing-index-out-of-range: Raised when a control plane advertised index fall outside the range of SRGBs configured for the network device.

8. YANG Module

The following RFCs and drafts are not referenced in the document text but are referenced in the ietf-segment-rouuting-common.yang and/or ietf-segment-routing.yang module: [RFC6991], [RFC8294], [RFC8476], and [RFC8491].

<CODE BEGINS> file "ietf-segment-routing-common@2019-07-06.yang"
module ietf-segment-routing-common {

```
yang-version 1.1;
namespace
  "urn:ietf:params:xml:ns:yang:ietf-segment-routing-common";
prefix sr-cmn;

import ietf-inet-types {
  prefix inet;
}

organization
  "IETF SPRING - SPRING Working Group";

contact
  "WG Web:   <http://tools.ietf.org/wg/spring/>
   WG List:  <mailto:spring@ietf.org>

   Editor:    Stephane Litkowski
              <mailto:stephane.litkowski@orange.com>
   Editor:    Yingzhen Qu
              <mailto:yingzhen.qu@futurewei.com>

   Author:    Acee Lindem
              <mailto:acee@cisco.com>
   Author:    Pushpasis Sarkar
              <mailto:pushpasis.ietf@gmail.com>
   Author:    Jeff Tantsura
              <jefftant.ietf@gmail.com>

  ";
description
  "The YANG module defines a collection of generic types and
   grouping for Segment Routing (SR) as described in RFC 8402.

   Copyright (c) 2019 IETF Trust and the persons identified as
   authors of the code.  All rights reserved.

   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject
   to the license terms contained in, the Simplified BSD License
   set forth in Section 4.c of the IETF Trust's Legal Provisions
   Relating to IETF Documents
   (http://trustee.ietf.org/license-info).

   This version of this YANG module is part of RFC XXXX;
   see the RFC itself for full legal notices.";

reference "RFC XXXX";
```

```
  revision 2019-07-06 {
    description
      "Initial version";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }

  feature sid-last-hop-behavior {
    description
      "Configurable last hop behavior.";
  }

  identity segment-routing-transport {
    description
      "Base identity for segment routing transport.";
  }

  identity segment-routing-transport-mpls {
    base segment-routing-transport;
    description
      "This identity represents MPLS transport for segment
       routing.";
  }

  identity segment-routing-transport-ipv6 {
    base segment-routing-transport;
    description
      "This identity represents IPv6 transport for segment
       routing.";
  }

  identity prefix-sid-algorithm {
    description
      "Base identity for prefix-sid algorithm.";
  }

  identity prefix-sid-algorithm-shortest-path {
    base prefix-sid-algorithm;
    description
      "Shortest Path First (SPF) prefix-sid algorithm. This
       is the default algorthm.";
  }

  identity prefix-sid-algorithm-strict-spf {
    base prefix-sid-algorithm;
    description
      "This algorithm mandates that the packet is forwarded
       according to ECMP-aware SPF algorithm.";
  }
```

```
    grouping srlr {
      description
        "Grouping for SR Label Range configuration.";
      leaf lower-bound {
          type uint32;
          description
            "Lower value in the label range.";
      }
      leaf upper-bound {
          type uint32;
          description
            "Upper value in the label range.";
      }
    }

    grouping srgb {
      description
        "Grouping for SR Global Label range.";
      list srgb {
        key "lower-bound upper-bound";
        ordered-by user;
        description
          "List of global blocks to be advertised.";
        uses srlr;
      }
    }

    grouping srlb {
      description
        "Grouping for SR Local Block range.";
      list srlb {
        key "lower-bound upper-bound";
        ordered-by user;
        description
          "List of SRLBs.";
        uses srlr;
      }
    }

    grouping sid-value-type {
      description
        "Defines how the SID value is expressed.";
      leaf value-type {
        type enumeration {
          enum "index" {
            description
              "The value will be interpreted as an index.";
          }
```

```
            enum "absolute" {
              description
                "The value will become interpreted as an absolute
                 value.";
            }
          }
          default "index";
          description
            "This leaf defines how value must be interpreted.";
        }
      }

      grouping prefix-sid {
        description
          "This grouping defines cfg of prefix SID.";
        leaf prefix {
          type inet:ip-prefix;
          description
            "connected prefix sid.";
        }
        uses prefix-sid-attributes;
      }

      grouping ipv4-sid {
        description
          "Grouping for an IPv4 prefix SID.";
        leaf prefix {
          type inet:ipv4-prefix;
          description
            "Connected IPv4 prefix sid.";
        }
        uses prefix-sid-attributes;
      }
      grouping ipv6-sid {
        description
          "Grouping for an IPv6 prefix SID.";
        leaf prefix {
          type inet:ipv6-prefix;
          description
            "Connected ipv6 prefix sid.";
        }
        uses prefix-sid-attributes;
      }

      grouping last-hop-behavior {
        description
          "Defines last hop behavior";
        leaf last-hop-behavior {
```

```
        if-feature "sid-last-hop-behavior";
        type enumeration {
          enum "explicit-null" {
            description
              "Use explicit-null for the SID.";
          }
          enum "no-php" {
            description
              "Do not use Penultimate Hop Popping (PHP)
               for the SID.";
          }
          enum "php" {
            description
              "Use PHP for the SID.";
          }
        }
        description
          "Configure last hop behavior.";
      }
    }

    grouping node-capabilities {
      description
        "Containing SR node capabilities.";
      container node-capabilities {
        config false;
        description
          "Shows the SR capability of the node.";
        list transport-planes {
          key "transport-plane";
          description
            "List of supported transport planes.";
          leaf transport-plane {
            type identityref {
              base segment-routing-transport;
            }
            description
              "Transport plane supported";
          }
        }
        leaf entropy-readable-label-depth {
          type uint8;
          description
            "Maximum label stack depth that a router can read.";
        }
      }
    }
```

```
    grouping prefix-sid-attributes {
      description
        "Grouping for Segment Routing (SR) prefix attributes.";
      uses sid-value-type;
      leaf start-sid {
        type uint32;
        mandatory true;
        description
          "Value associated with prefix. The value must be
           interpreted in the context of value-type.";
      }
      leaf range {
        type uint32;
        description
          "Indicates how many SIDs can be allocated.";
      }
      leaf algorithm {
        type identityref {
          base prefix-sid-algorithm;
        }
        description
          "Prefix-sid algorithm.";
      }
    }
  }
  <CODE ENDS>
  <CODE BEGINS> file "ietf-segment-routing@2019-07-06.yang"
  module ietf-segment-routing {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-segment-routing";
    prefix sr;

    import ietf-inet-types {
      prefix inet;
    }
    import ietf-routing {
      prefix rt;
    }
    import ietf-interfaces {
      prefix if;
    }
    import ietf-routing-types {
      prefix rt-types;
    }
    import ietf-segment-routing-common {
      prefix sr-cmn;
    }
```

```
  organization
    "IETF SPRING - SPRING Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/spring/>
     WG List:  <mailto:spring@ietf.org>

     Editor:    Stephane Litkowski
                <mailto:stephane.litkowski@orange.com>
     Editor:    Yingzhen Qu
                <mailto:yingzhen.qu@futurewei.com>

     Author:    Acee Lindem
                <mailto:acee@cisco.com>
     Author:    Pushpasis Sarkar
                <mailto:pushpasis.ietf@gmail.com>
     Author:    Jeff Tantsura
                <jefftant.ietf@gmail.com>

    ";
  description
    "The YANG module defines a generic configuration model for
     Segment routing common across all of the vendor
     implementations.

     Copyright (c) 2019 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX;
     see the RFC itself for full legal notices.";

  reference "RFC XXXX";

  revision 2019-07-06 {
    description
      "Initial Version";
    reference "RFC XXXX: YANG Data Model for Segment Routing.";
  }
  feature mapping-server {
    description
      "Support for Segment Routing Mapping Server (SRMS).";
  }
```

```
feature protocol-srgb {
  description
    "Support for per-protocol Segment Routing Global Block
     (SRGB) configuration.";
}

feature max-sid-depth {
  description
    "Support for signaling MSD (Maximum SID Depth) in IGP.";
}

typedef system-id {
  type string {
    pattern
      '[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}';
  }
  description
    "This type defines IS-IS system-id using pattern,
     An example system-id is 0143.0438.AEF0";
}

typedef router-id {
  type union {
    type system-id;
    type rt-types:router-id;
  }
  description
    "OSPF/BGP router-id or ISIS system ID.";
}

grouping sr-controlplane {
  description
    "Defines protocol configuration.";
  container segment-routing {
    description
      "Segment Routing global configuration.";
    leaf enabled {
      type boolean;
      default "false";
      description
        "Enables segment-routing protocol extensions.";
    }
    container bindings {
      description
        "Control of binding advertisement and reception.";
      container advertise {
        description
          "Control advertisement of local mappings
```

```
               in binding TLVs.";
            leaf-list policies {
              type string;
              description
                "List of binding advertisement policies.";
            }
          }
          leaf receive {
            type boolean;
            default "true";
            description
              "Allow the reception and usage of binding TLVs.";
          }
        }
      }
    }

    grouping igp-interface {
      description
        "Grouping for IGP interface configuration.";
      container segment-routing {
        description
          "Container for SR interface configuration.";
        container adjacency-sid {
          description
            "Adjacency SID configuration.";
          list adj-sids {
            key "value";
            uses sr-cmn:sid-value-type;
            leaf value {
              type uint32;
              description
                "Value of the Adj-SID.";
            }
            leaf protected {
              type boolean;
              default false;
              description
                "It is used to protect the mannual adj-SID.";
            }
            description
              "List of adj-sid configuration.";
          }
          list advertise-adj-group-sid {
            key "group-id";
            description
              "Control advertisement of S flag. Enable advertisement
               of a common Adj-SID for parallel links.";
```

```
            leaf group-id {
              type uint32;
              description
                "The value is an internal value to identify a
                 group-ID. Interfaces with the same group-ID will be
                 bundled together.";
            }
          }
          leaf advertise-protection {
            type enumeration {
              enum "single" {
                description
                  "A single Adj-SID is associated with the adjacency
                   and reflects the protection configuration.";
              }
              enum "dual" {
                description
                  "Two Adj-SIDs will be associated with the adjacency
                   if the interface is protected. In this case, will
                   be advertised with backup flag set, the other will
                   be advertised with theo backup flag clear. In case
                   protection is not configured, single Adj-SID will
                   be advertised with the backup flag clear.";
              }
            }
            description
              "If set, the Adj-SID refers to a protected adjacency.";
          }
        }
      }
    }

    grouping max-sid-depth {
      description
        "Maximum SID Depth (MSD)D configuration grouping.";
      leaf node-msd {
        type uint8;
        description
          "Node MSD is the lowest MSD supported by the node.";
      }
      container link-msd {
        description
          "MSD supported by an individual interface.";
        list link-msds {
          key "interface";
          description
            "List of link MSDs.";
          leaf interface {
```

```
            type if:interface-ref;
            description
              "Reference to device interface.";
          }
          leaf msd {
            type uint8;
            description
              "MSD supported by the interface.";
          }
        }
      }
    }

    augment "/rt:routing" {
      description
        "This augments routing data model (RFC 8349)
         with Segment Routing (SR).";
      container segment-routing {
        description
          "Segment Routing global configuration.";
        leaf transport-type {
          type identityref {
            base sr-cmn:segment-routing-transport;
          }
          default "sr-cmn:segment-routing-transport-mpls";
          description
            "Dataplane to be used.";
        }
        uses sr-cmn:node-capabilities;
        container msd {
          if-feature "max-sid-depth";
          description
            "MSD configuration.";
          uses max-sid-depth;
        }
        container bindings {
          description
            "List of bindings.";
          container mapping-server {
            if-feature "mapping-server";
            description
              "Configuration of mapping-server local entries.";
            list policy {
              key "name";
              description
                "List mapping-server policies.";
              leaf name {
                type string;
```

```
            description
              "Name of the mapping policy.";
          }
          container entries {
            description
              "IPv4/IPv6 mapping entries.";
            list mapping-entry {
              key "prefix algorithm";
              description
                "Mapping entries.";
              uses sr-cmn:prefix-sid;
            }
          }
        }
      }
      container connected-prefix-sid-map {
        description
          "Prefix SID configuration.";
        list connected-prefix-sid {
          key "prefix algorithm";
          description
            "List of prefix SID mapped to IPv4/IPv6
             local prefixes.";
          uses sr-cmn:prefix-sid;
          uses sr-cmn:last-hop-behavior;
        }
      }
      container local-prefix-sid {
        description
          "Local sid configuration.";
        list local-prefix-sid {
          key "prefix algorithm";
          description
            "List of local IPv4/IPv6 prefix-sids.";
          uses sr-cmn:prefix-sid;
        }
      }
    }
    container global-srgb {
      description
        "Global SRGB configuration.";
      uses sr-cmn:srgb;
    }
    container srlb {
      description
        "Segment Routing Local Block (SRLB) configuration.";
      uses sr-cmn:srlb;
    }
```

```
        list label-blocks {
          config false;
          description
            "List of label blocks currently in use.";
          leaf lower-bound {
            type uint32;
            description
              "Lower bound of the label block.";
          }
          leaf upper-bound {
            type uint32;
            description
              "Upper bound of the label block.";
          }
          leaf size {
            type uint32;
            description
              "Number of indexes in the block.";
          }
          leaf free {
            type uint32;
            description
              "Number of free indexes in the block.";
          }
          leaf used {
            type uint32;
            description
              "Number of indexes in use in the block.";
          }
          leaf scope {
            type enumeration {
              enum "global" {
                description
                  "Global SID.";
              }
              enum "local" {
                description
                  "Local SID.";
              }
            }
            description
              "Scope of this label block.";
          }
        }
        container sid-list {
          config false;
          description
            "List of prefix and SID associations.";
```

```
          list sid {
            key "target sid source source-protocol binding-type";
            ordered-by system;
            description
              "SID Binding.";
            leaf target {
              type string;
              description
                "Defines the target of the binding. It can be a
                 prefix or something else.";
            }
            leaf sid {
              type uint32;
              description
                "Index associated with the prefix.";
            }
            leaf algorithm {
              type uint8;
              description
                "Algorithm to be used for the prefix SID.";
            }
            leaf source {
              type inet:ip-address;
              description
                "IP address of the router that owns the binding.";
            }
            leaf used {
              type boolean;
              description
                "Indicates if the binding is install in the
                 forwarding plane.";
            }
            leaf source-protocol {
              type leafref {
                path "/rt:routing/rt:control-plane-protocols/"
                   + "rt:control-plane-protocol/rt:name";
              }
              description
                "Routing protocol that owns the binding";
            }
            leaf binding-type {
              type enumeration {
                enum "prefix-sid" {
                  description
                    "Binding is learned from a prefix SID.";
                }
                enum "binding-tlv" {
                  description
```

```
                    "Binding is learned from a binding TLV.";
                }
              }
              description
                "Type of binding.";
            }
            leaf scope {
              type enumeration {
                enum "global" {
                  description
                    "Global SID.";
                }
                enum "local" {
                  description
                    "Local SID.";
                }
              }
              description
                "SID scoping.";
            }
          }
        }
      }
    }

    notification segment-routing-global-srgb-collision {
      description
        "This notification is sent when SRGB blocks received from
         routers conflict.";
      list srgb-collisions {
        description
          "List of SRGB blocks that conflict.";
        leaf lower-bound {
          type uint32;
          description
            "Lower value in the block.";
        }
        leaf upper-bound {
          type uint32;
          description
            "Upper value in the block.";
        }
        leaf routing-protocol {
          type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
               + "rt:control-plane-protocol/rt:name";
          }
          description
```

```
            "Routing protocol reference for SRGB collision.";
        }
        leaf originating-rtr-id {
          type router-id;
          description
            "Originating Router ID of this SRGB block.";
        }
      }
    }
    notification segment-routing-global-sid-collision {
      description
        "This notification is sent when a new mapping is learned
         containing s mapping where the SID is already used.
         The notification generation must be throttled with at least
         a 5 second gap between notifications.";
      leaf received-target {
        type string;
        description
          "Target received in the router advertisement that caused
           the SID collision.";
      }
      leaf new-sid-rtr-id {
        type router-id;
        description
          "Router ID that advertised the conflicting SID.";
      }
      leaf original-target {
        type string;
        description
          "Target already available in the database with the same SID
           as the received target.";
      }
      leaf original-sid-rtr-id {
        type router-id;
        description
          "Router-ID for the router that originally advertised the
           conflicting SID, i.e., the instance in the database.";
      }
      leaf index {
        type uint32;
        description
          "Value of the index used by two different prefixes.";
      }
      leaf routing-protocol {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
             + "rt:control-plane-protocol/rt:name";
        }
```

```
        description
          "Routing protocol reference for conflicting SID.";
      }
    }
    notification segment-routing-index-out-of-range {
      description
        "This notification is sent when a binding is received
         containing a segment index  which is out of the local
         configured ranges. The notification generation must be
         throttled with at least a 5 second gap between
         notifications.";
      leaf received-target {
        type string;
        description
          "Target received in the router advertisement with
           the out-of-range index.";
      }
      leaf received-index {
        type uint32;
        description
          "Value of the index received.";
      }
      leaf routing-protocol {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
             + "rt:control-plane-protocol/rt:name";
        }
        description
          "Routing protocol reference for out-of-range indexd.";
      }
    }
  }
  <CODE ENDS>
```

9. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in the modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations.

Some of the readable data nodes in the modules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes.

10. Acknowledgements

Authors would like to thank Derek Yeung, Greg Hankins, Hannes Gredler, Uma Chunduri, Jeffrey Zhang, Shradda Hedge, Les Ginsberg for their contributions.

11. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

```
URI: urn:ietf:params:xml:ns:yang:ietf-segment-routing-commmon
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-segment-routing
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
name: ietf-segment-routing-common
namespace: urn:ietf:params:xml:ns:yang:ietf-segment-routing-common
prefix: sr-cmn
reference: RFC XXXX

name: ietf-segment-routing
namespace: urn:ietf:params:xml:ns:yang:ietf-segment-routing
prefix: sr
reference: RFC XXXX
```

12. References

12.1. Normative References

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

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

[RFC8476] Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling Maximum SID Depth (MSD) Using OSPF", RFC 8476, DOI 10.17487/RFC8476, December 2018, <https://www.rfc-editor.org/info/rfc8476>.

[RFC8491] Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling Maximum SID Depth (MSD) Using IS-IS", RFC 8491, DOI 10.17487/RFC8491, November 2018, <https://www.rfc-editor.org/info/rfc8491>.

12.2. Informative References

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Authors' Addresses

Stephane Litkowski
Orange Business Service

Email: stephane.litkowski@orange.com

Yingzhen Qu
Futurewei

Email: yingzhen.qu@futurewei.com

Acee Lindem
Cisco
301 Mindenhall Way
Cary, NC 27513
US

Email: acee@cisco.com

Pushpasis Sarkar
Individual

Email: pushpasis.ietf@gmail.com

Jeff Tantsura
Apstra

Email: jefftant.ietf@gmail.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2017

S. Litkowski
Orange
Mustapha Aissaoui
Nokia

October 31, 2016

Implementing non protected paths using SPRING
draft-litkowski-spring-non-protected-paths-00

Abstract

Segment Routing (SR) leverages the source routing paradigm. A node can steer a packet on a specific path by prepending the packet with an SR header. In the framework of traffic-engineering use cases, a customer may request its service provider to implement some non protected paths. This means that in case of a failure within the network, fast-reroute (or similar) techniques should not be activated for those paths. This document analyzes the different options to implement a non protected path with Segment Routing and in a future release will provide a recommandation on the best option.

Requirements Language

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

Status of This Memo

Table of Contents

1. Problem statement

In some cases, a customer may prefer to manage to react on network failures using its own mechanism. In such cases, the customer usually has two disjoint paths, so a path can take over the traffic in case of failure of the other. The disjoint paths can be provided by a single provider or by multihoming to different providers as displayed in the figure below.

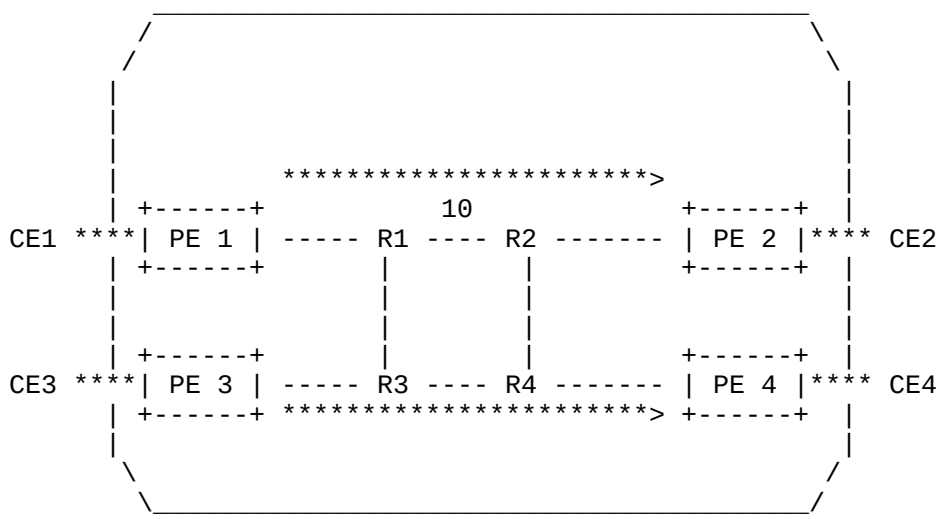


Figure 1 - Disjoint paths provided by a single provider

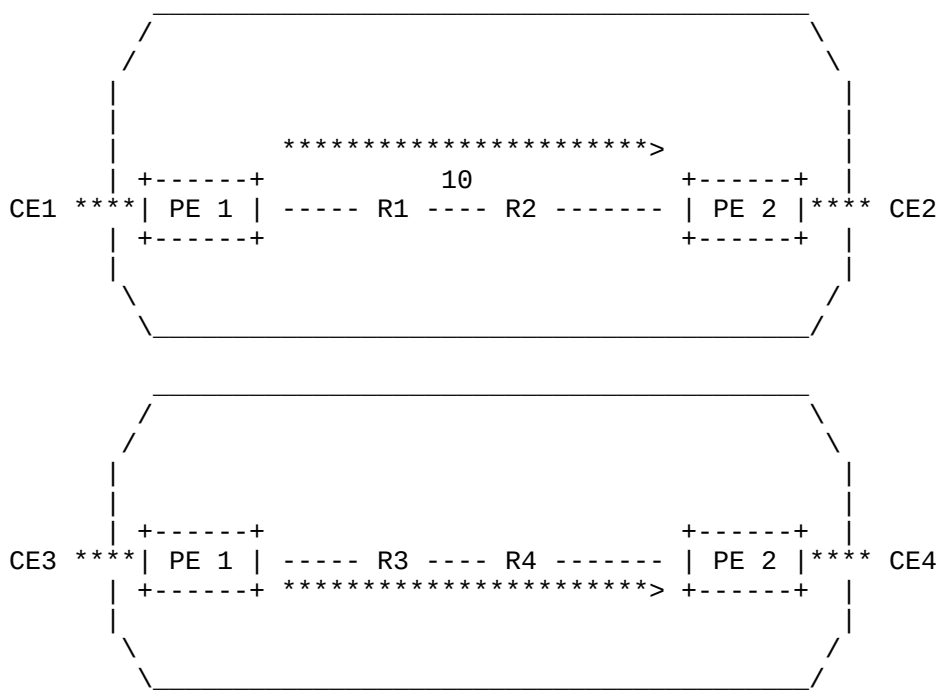


Figure 2 - Disjoint paths provided by using two providers

As the traffic protection is ensured by an end-to-end mechanism at the customer level, the customer requests the service provider to not protect the paths. However the service provider is allowed to restore the service automatically when the primary path is failing by computing and installing a new path in the network. A variant to this scheme is for the service provider to provision a unprotected secondary path between PE1 and PE2 which is also disjoint from the primary path. In the latter case, the end-to-end service protection at the CE becomes the last resort. How the end-to-end protection is handled is out of scope of this document and will be under the customer responsibility.

A segment-routing path is expressed as a list of segment identifiers (SID) from different types (Node-SID, Adj-SID, Binding-SID ...). In order to ensure that the segment routing path is not protected, we need to ensure that it does not contain any segment representing a protected path. As an example, in the Figure 1, we consider a path from PE1 to PE2 expressed with the following segment list: {Adj_R1R3,Node_R2,Adj_R2PE2}. If we want to ensure that this path is non protected, we need to ensure that the segment represented by Adj_R1R3 represents a non protected, as well as the segment Node_R2 and Adj_R2PE2.

The segment routing path may be computed by a Path Computation Element (PCE). In order to fulfil the non protected path constraint, the PCE needs to be aware of the available SIDs in the network and their protection status.

Several techniques may be used to represent a non protected path with a segment identifier. We propose to analyze the different options.

2. Options to create a non protected path with Segment Routing

2.1. Using only non protected adjacency segments

The adjacency segment was created so a node can advertise multiple adjacency segments for a particular link with different properties. The non-protected property is already defined as part of the protocol encodings ([I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions] and [I-D.gredler-idr-bgp-ls-segment-routing-extension]) through the B flag. However, from an implementation perspective, advertising a protected adjacency segment, a non protected adjacency segment or both for each link is optional.

It is important to note that even if an adjacency segment has the B flag set (protected), it remains up to a local policy of the advertising router to implement the protection or not.

If both protected and non protected Adj-SID are advertised, every node in the network and a PCE can be aware of the adjacency segments protection property. When a non protected path is requested, the path computation module can choose to encode the path with a list a non protected adjacency segment only.

One of the advantage of using only adjacency segments is the insurance that the traffic will never go transiently outside the path defined by the computation module reponsible of the path.

One of the drawbacks of using only adjacency segments is the resulting label stack depth as each hop should require a segment in the stack: crossing 15 nodes, means stacking 15 labels for the transport. Having such a deep stack may be a problem for current hardwares and softwares for either pushing the stack (because the head end is limited in the number of labels it can push) or loadbalancing flows on transit nodes (as deep packet inspection or entropy label look up may be difficult with a deep label stack). Another drawback of advertising both protected and non protected adjacency segments is the additional controlplane and dataplane resource consumption used in the network.

2.2. Using a combination of node segments and adjacency segments

Using a combination of node segments and adjacency segments is the usual way when creating a segment routing path. However the well known Node-SID (algorithm type Shortest Path) may be protected by a local-repair mechanism by any transit node or may have multiple ECMP next-hops which may be a problem when used for a non protected path. Protecting a particular Node-SID is a matter of a local policy configuration on every node. The following discusses a number of possible approaches.

2.2.1. Using Strict SPF Node SID

[I-D.ietf-spring-segment-routing] defines a Strict Shortest Path algorithm which mandates that the packet is forwarded according to ECMP-aware SPF algorithm and instructs any router in the path to ignore any possible local policy overriding SPF decision. The provided definition in this document does not clearly state that the Strict Shortest Path Node-SID cannot use a protection path but makes think that the path cannot be overriden by a local policy including a fast-reroute policy. If the scope of the Strict SPF Node-SID is clarified to mean one or more primary next-hops is selected with no local-repair, combining Strict SPF Node-SID with non protected Adj-SID may be a viable option to encode a non protected path. The selection of the primary next-hop(s) may be left to the local SPF calculation otherwise an adjacency SID can be used to exercise a specific next-hop or set of next-hops.

If this solution is viable, when a network wants to implement both protected and non protected paths, the network requires the advertisement of two Node SIDs per node (one with SPF algorithm for protected paths, and one with Strict SPF algorithm for non protected paths).

2.2.2. Adding a protection flag in the Node SID

As for adjacency segments, a new flag may be added in the Prefix-SID to encode the willigness of protection. Each node will then advertise two Node-SID (using SPF algorithm), one with the protection flag set, the other without the protection flag set. The same discussion regarding ECMP is also applicable here.

2.2.3. Using two Node-SIDs with different local policies

Having two instances of the Node-SID (protected and not protected) is a requirement when using Node-SID in protected and non protected paths. The protection of a Node-SID is a matter of a local policy configuration on every node in the network. A service provider may configure two Node-SIDs per node and may adjust the local-repair on every node to protect one Node-SID but not the other. As the protection of the Node-SID is inherited from the protection of the associated prefix, the service provider will need to deploy a new set of prefixes to all nodes to deploy the new set of Node-SIDs. Then it will need to maintain the local-repair policy on every node to ensure

that the prefixes associated to the non protected Node-SID are not using the local-repair.

2.2.4. Advantages and drawbacks

One advantage of combining adjacency and node segments is the reduction of the label stack size.

The drawbacks are the increase of the controlplane and dataplane resource consumption, leading possibly to higher convergence time. One of the other drawback is that a Node-SID may be transiently rerouted on a path that does not fit the constraints anymore if a transit node converges faster than the head-end: this concern is not new and applies to all traffic-engineering use cases.

2.3. Using a combination of adjacency segments and binding-SID

[I-D.ietf-spring-segment-routing] defines the binding segment with multiple use cases. One of the use case of the binding segment is to advertise a tunnel as a segment. When a computation engine computes a non protected path and if the resulting label stack using only non protected adjacency segments is too deep for the network, an external component may create shortcuts in the network by creating a binding segment representing a list of non protected adjacency segments.

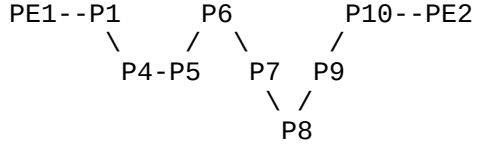


Figure 3 - Use of Binding SID

In the example above, the path from PE1 to PE2 must be expressed with the stack: {Adj_P1P4,Adj_P4P5,Adj_P5P6,Adj_P6P7,Adj_P7P8,Adj_P8P9,Adj_P9P10,Adj_P10PE2}. This stack is too deep due to the limitations of the network. An external component may create a binding Binding1 on P5 that represents the non protected path (P5->P6->P7->P8->P9->P10). When the binding is created and advertised in the topology, the computation engine can use this binding SID in a path, resulting for a PE1 to PE2 path to the stack: {Adj_P1P4,Adj_P4P5,Binding1,Adj_P10PE2}. The usage of the binding SID in the stack allowed to reduce its size to an acceptable value.

One advantage of combining adjacency and binding segments is the reduction of the label stack size.

The drawbacks are the increase of the controlplane and dataplane resource consumption. This controlplane and dataplane resource consumption will be linked to the intelligence of the external controller and computation engines and especially how the placement of the bindings is done to maximize the sharing between LSPs. Moreover any optimization try in the binding segment may introduce churn in the network controlplane (Make Before Break can be used to ensure that dataplane is not affected).

3. Recommended option(s)

TBD.

4. Security Considerations

TBD.

5. Acknowledgements

6. IANA Considerations

N/A

7. Normative References

[I-D.gredler-idr-bgp-ls-segment-routing-extension]
Gredler, H., Ray, S., Previdi, S., Filsfils, C., Chen, M., and J. Tantsura, "BGP Link-State extensions for Segment Routing", draft-gredler-idr-bgp-ls-segment-routing-extension-02 (work in progress), October 2014.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and j. jefftant@gmail.com, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-08 (work in progress), October 2016.

[I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions-09 (work in progress), July 2016.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-09 (work in progress), July 2016.

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

Author’s Address

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Mustapha Aissaoui
Nokia

Email: mustapha.aissaoui@nokia.com

SPRING Working Group
Internet-Draft
Intended status: Informational
Expires: February 10, 2018

S. Litkowski
Orange
M. Aissaoui
Nokia

August 9, 2017

Implementing non protected paths using SPRING
draft-litkowski-spring-non-protected-paths-02

Abstract

Segment Routing (SR) leverages the source routing paradigm. A node can steer a packet on a specific path by prepending the packet with an SR header. In the framework of traffic-engineering use cases, a customer may request its service provider to implement some non protected paths. This means that in case of a failure within the network, fast-reroute (or similar) techniques should not be activated for those paths. This document analyzes the different options to implement a non protected path with Segment Routing and in a future release will provide a recommandation on the best option.

Requirements Language

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

Status of This Memo

Table of Contents

1. Problem statement

In some cases, a customer may prefer to react on network failures using its own mechanism. In such cases, the customer usually has two disjoint paths, so a path can take over the traffic in case of failure of the other. The disjoint paths can be provided by a single provider or by multihoming to different providers as displayed in the figure below.

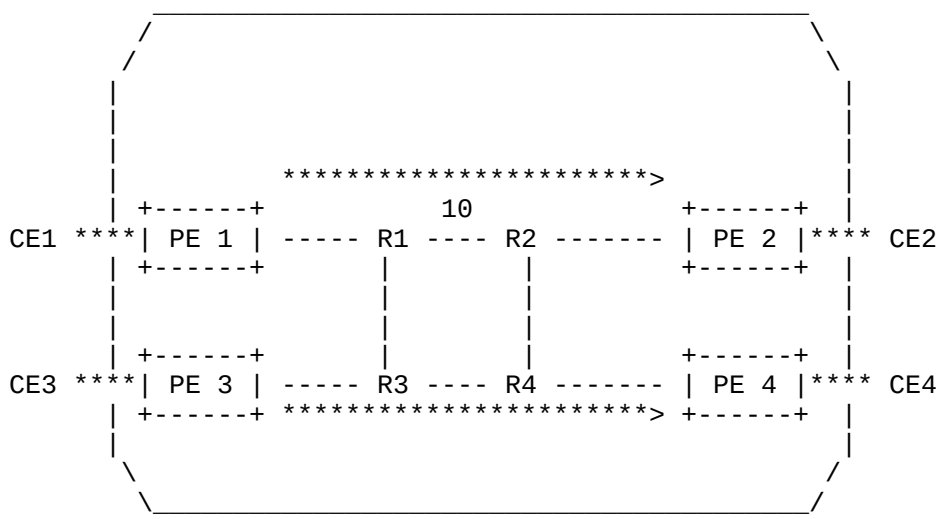


Figure 1 - Disjoint paths provided by a single provider

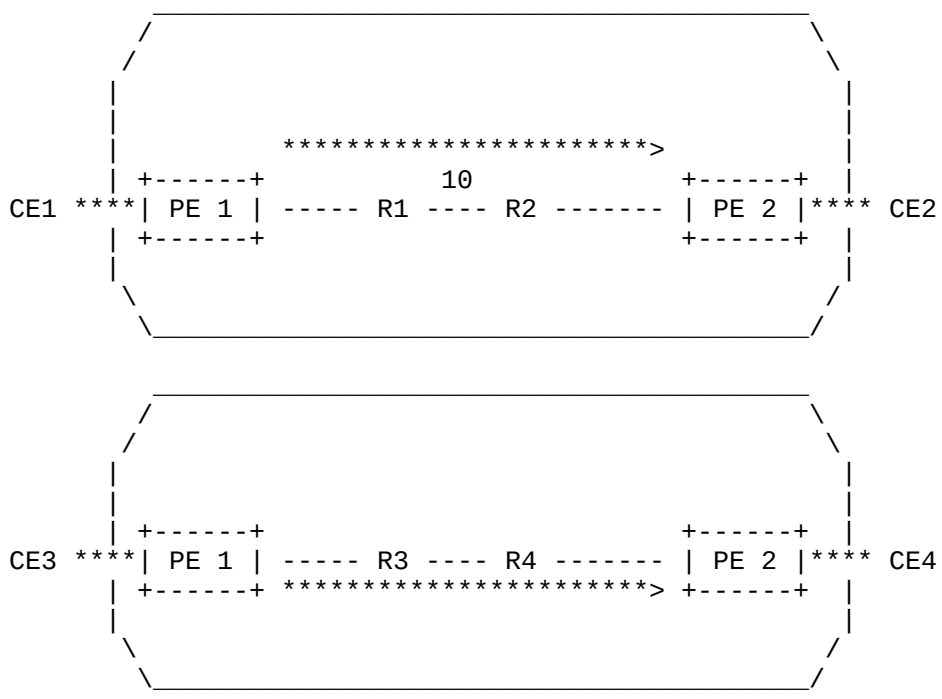


Figure 2 - Disjoint paths provided by using two providers

As the traffic protection is ensured by an end-to-end mechanism at the customer level, the customer requests the service provider to not protect the paths. This is particularly required to avoid both protection mechanisms (customer level and provider level) to be activated at the same time which may lead to unpredictable side effects. However the service provider is allowed to restore the end-to-end path automatically when the primary path is failing by computing and installing a new primary path at the head-end. How the end-to-end protection is handled is out of scope of this document and will be under the customer responsibility.

Another use case could be a service provider selling the traffic protection as a service option. So by default, the provided IP/MPLS path is not protected by any fast-reroute mechanism but the customer can subscribe to an option to activate fast-reroute for its traffic. In the figure 3, the Customer1 service between PE1 and PE2 is protected, in case of failure between R1 and R2, the LSP can use a bypass through R3-R4 nodes until the convergence occurs. The Customer2 did not subscribe to the traffic protection option. If

R3-R4 fails, the traffic between CE3 and CE4 will be disrupted until the convergence occurs.

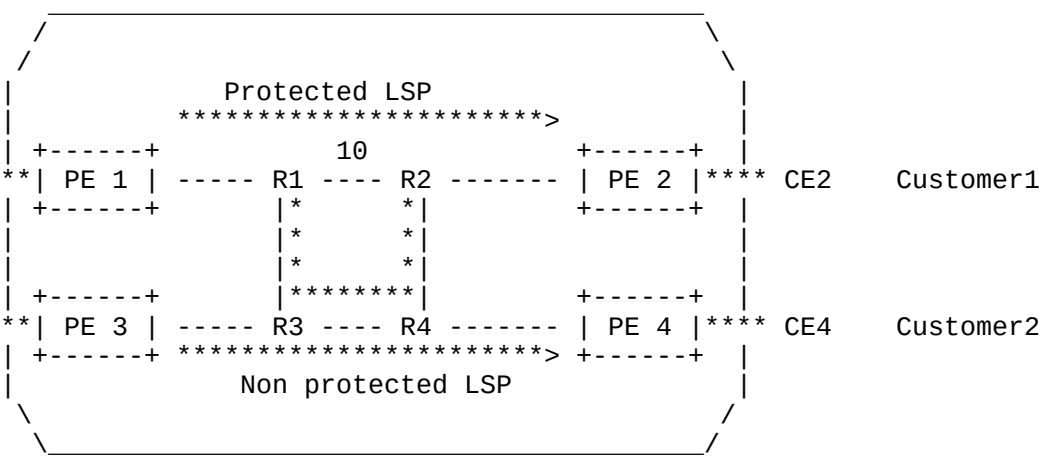


Figure 3 - Provider selling traffic protection as an option

A service provider may also propose a traffic protection service based on path protection rather than local repair on each transit node. In the figure 4, on PE1, two LSPs were created to ensure the customer traffic protection between PE1 and PE2. The primary LSP is used to carry the traffic in the nominal situation. The protection LSP is built as disjoint from the primary LSP and may be preestablished (from controlplane and/or dataplane point of view). When the primary LSP fails, PE1 is responsible to switch the traffic to the protection LSP. As the protection is provided by PE1, both primary and protection LSPs should be setup as non protected so transit nodes will not activate any local-repair mechanism for those LSPs.

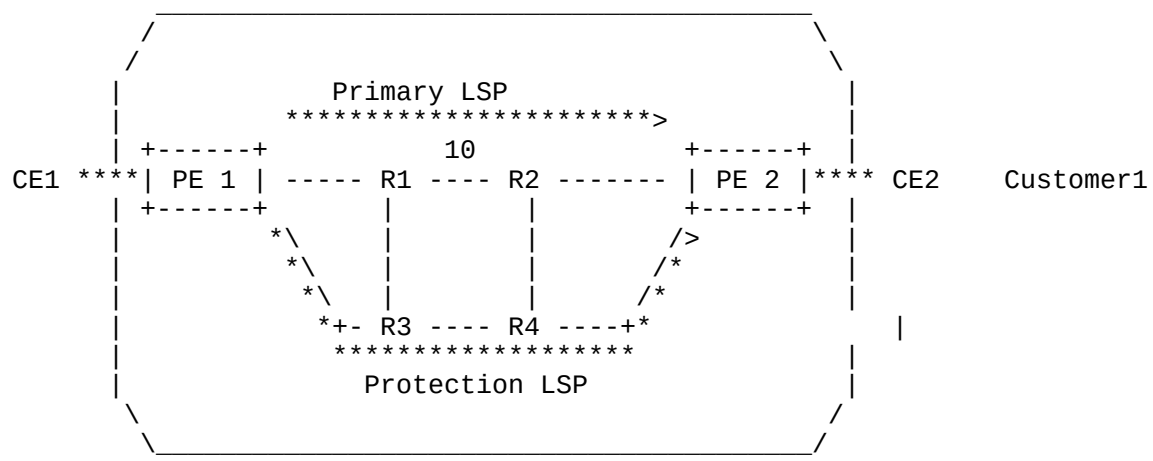


Figure 4 - Provider selling traffic protection as an option

A segment-routing path is expressed as a list of segment identifiers (SID) from different types (Node-SID, Adj-SID, Binding-SID ...). In order to ensure that the segment routing path is not protected, we need to ensure that it does not contain any segment representing a protected path. As an example, in the Figure 1, we consider a path from PE1 to PE2 expressed with the following segment list: {Adj_R1R3,Node_R2,Adj_R2PE2}. If we want to ensure that this path is not protected, we need to ensure that the segment represented by Adj_R1R3 represents a non protected segment, as well as the segments Node_R2 and Adj_R2PE2.

The segment routing path may be computed by a Path Computation Element (PCE). In order to fulfil the non protected path constraint, the PCE needs to be aware of the available SIDs in the network and their protection status.

Several techniques may be used to represent a non protected path with a segment identifier. We propose to analyze the different options.

2. Requirements for a non protected LSP

- o A non protected LSP SHOULD follow a primary path defined based on the constraints of the LSP. This path can be the shortest path (as per the IGP metric) or a more constrained path (explicit path) to fulfil for example a bandwidth, latency or disjointness requirement.

- Upon a failure, a non protected LSP SHOULD be reestablished over a new suitable non-protected path that still fulfils the constraints of the LSP.

- Upon a failure (link, node, srlg...), the traffic of a non protected LSP MUST NOT use any local-repair or any local-rerouting mechanism on transit nodes.

- The computation of a new primary path for the LSP will be handled by the computation node responsible of this LSP (it could be the head-end or a PCE).

- Upon any other traffic-engineering topology change (metric change, overload status change, bandwidth change, latency change...), the non protected LSP MAY be reoptimized to a better path.

2.1. ECMP considerations

When equal cost paths are available within the end-to-end path, implementations may reuse a fast-reroute like mechanism in the dataplane, so when one of the outgoing interface fails, the dataplane switches traffic immediately to the remaining outgoing interfaces in the ECMP set. This behavior is usually hardcoded and cannot be disabled. Based on this assumption, a non protected LSP SHOULD avoid ECMPs.

3. Options to create a non protected path with Segment Routing

3.1. Using only non protected adjacency segments

A node can advertise multiple adjacency segments for a particular link with different properties. The non-protected property is already defined as part of the protocol encodings ([I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions] and [I-D.gredler-idr-bgp-ls-segment-routing-extension]) through the B flag. However, from an implementation perspective, advertising a protected adjacency segment, a non protected adjacency segment or both for each link is optional.

It is important to note that even if an adjacency segment has the B flag set (protected), it remains up to a local policy of the advertising router to implement the protection or not.

If both protected and non protected Adj-SID are advertised, every node in the network (including PCEs) can be aware of the adjacency segments protection property. When a non protected path is

requested, the path computation module can choose to encode the path with a list a non protected adjacency segments only.

One of the advantage of using only adjacency segments is the insurance that the traffic will never go transiently outside the path defined by the computation module responsible of the path. This solution is fully compliant with the requirements sets in Section 2.

One of the drawbacks of using only adjacency segments is the resulting label stack depth as each hop should require a segment in the stack: crossing 15 nodes, means stacking 15 labels to encode the SR tunnel. Having such a deep stack may be a problem for current hardwares and softwares for either pushing the stack (because the head end is limited in the number of labels it can push) or loadbalancing flows on transit nodes (as deep packet inspection or entropy label look up may be difficult with a deep label stack). Another drawback of advertising both protected and non protected adjacency segments is the additional controlplane and dataplane resource consumption used in the network. As the adjacency SIDs have a local significance, this resource consumption can be considered as negligeable from a data plane point of view. From a control plane point of view, this can also be considered as negligeable with the current CPU and memory usually available on routers.

3.2. Using a combination of node segments and adjacency segments

Using a combination of node segments and adjacency segments is the usual way of creating a segment routing path. However the well known Node-SID (algorithm type Shortest Path) may be protected by a local-repair mechanism by any transit node or may use ECMPs which may be a problem when used for a non protected path. Protecting a particular Node-SID is a matter of a local policy configuration on every node. The following discusses a number of possible approaches.

3.2.1. Adding a protection flag in the Node SID

As for adjacency segments, a new flag may be added in the Prefix-SID to encode the willingness of protection. Each node will then advertise two Node-SIDs (using SPF algorithm), one with the protection flag set, the other without the protection flag set. The same discussion regarding ECMP is also applicable here.

The remaining flag space in the Prefix-SID is small, so adding a new flag requires analysis but this should not be considered as a showstopper.

3.2.2. Using Strict SPF Node SID

[I-D.ietf-spring-segment-routing] defines a Strict Shortest Path algorithm which mandates that the packet is forwarded according to ECMP-aware SPF algorithm and instructs any router in the path to ignore any possible local policy overriding SPF decision. The use of a local-repair for a strict SPF Node-SID is allowed as long as the FRR mechanism enforces the post convergence path to the destination.

This solution does not bring any benefit compared to the regular Node-SID (as it has similar properties).

3.2.3. Using two Node-SIDs with different local policies

Having two instances of the Node-SID (protected and not protected) is a requirement when using Node-SID in protected and non protected paths. The protection of a Node-SID is a matter of a local policy configuration on every node in the network. A service provider may configure two Node-SIDs per node and may adjust the local-repair on every node to protect one Node-SID but not the other. As the protection of the Node-SID is inherited from the protection of the associated prefix, the service provider will need to deploy a new set of prefixes to all nodes to deploy the new set of Node-SIDs. Then it will need to maintain the local-repair policy on every node to ensure that the prefixes associated to the non protected Node-SID are not using the local-repair.

The path computation engine (head-end or PCE) must be aware of the policy defined by the service provider so it can select the right SIDs/prefixes when computing a path.

3.2.4. Advantages and drawbacks

One advantage of combining adjacency and node segments is the reduction of the label stack size.

The drawbacks are the increase of the controlplane and dataplane resource consumption. Whereas having two adjacency SIDs introduces a negligeable impact, having two nodes SIDs increases controlplane and dataplane processing as each node in the network will have to install an MPLS->MPLS and IP->MPLS entry for each additional Node-SID. The regular IP convergence time of the network may be doubled in the worst case while the newly deployed node-SIDs are only used for traffic-engineering applications. One of the other drawback is that a Node-SID may be transiently rerouted on a path that does not fit the constraints anymore if a transit node converges faster than the head-end: this concern is not new and applies to all traffic-engineering use cases. Note that there is a high chance for a

transit node to reroute faster than the head-end as it has usually less computations to run (SPF+CSPFs) and less prefixes to rewrite; it may also run less features leaving more CPU slots for IGP reconvergence. The transient rerouting of the Node-SID may lead to microloops in the network that may impact the customer traffic. Node-SIDs are subject to ECMP and a local-repair mechanism may be implemented for equal cost paths with no way to disable it. If the requirement of preventing any local-repair or ECMP is strict, the path computation engine needs to prevent the usage of all Node-SIDs or needs to detect that a particular Node-SID will be subject to ECMP and enforce the usage of additional adjacency SIDs to break the ECMP. In any case, more adjacency-SIDs will be required in the stack to avoid the ECMP, leading to a deeper label stack.

3.3. Using a combination of adjacency segments and binding-SID

[I-D.ietf-spring-segment-routing] defines the binding segment with multiple use cases. One of the use case of the binding segment is to advertise a tunnel as a segment. When a computation engine computes a non protected path and if the resulting label stack using only non protected adjacency segments is too deep for the network, an external component may create shortcuts in the network by creating a binding segment representing a list of non protected adjacency segments.

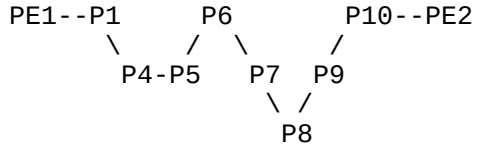


Figure 3 - Use of Binding SID

In the example above, the path from PE1 to PE2 must be expressed with the stack: {Adj_P1P4,Adj_P4P5,Adj_P5P6,Adj_P6P7,Adj_P7P8,Adj_P8P9,Adj_P9P10,Adj_P10PE2}. This stack is too deep due to the limitations of the network. An external component may create a binding Binding1 on P5 that represents the non protected path (P5->P6->P7->P8->P9->P10). When the binding is created and advertised in the topology, the computation engine can use this binding SID in a path, resulting for a PE1 to PE2 path to the stack: {Adj_P1P4,Adj_P4P5,Binding1,Adj_P10PE2}. The usage of the binding SID in the stack allowed to reduce its size to an acceptable value.

One advantage of combining adjacency and binding segments is the reduction of the label stack size. The label stack size can be

reduced to a small amount of labels at some price (creating some states on transit nodes).

The drawbacks are the increase of the controlplane and dataplane resource consumption. This controlplane and dataplane resource consumption are variable and will be linked to the intelligence of the external controller and computation engines and especially how the placement of the bindings is done to maximize the sharing between LSPs. Moreover any optimization try in the binding segment may introduce churn in the network controlplane (Make Before Break can be used to ensure that dataplane is not affected). Programming a binding-SID on a transit node is feasible only if the programming node has the necessary protocol sessions to do so. When a head-end router is performing a path computation, it is usually not the case. When a controller (PCE) is used, it may not have a session to all LSRs in the network, as only edge nodes may require a path computation. The controller may be limited for the placement of the binding SID to the nodes it has a protocol session with (it cannot setup a PCEP session by itself). A full deployment of protocol sessions with the controller may not be feasible for technical reasons (scaling, ...) or economical reasons. A potential mitigation could be to allow protocol sessions to be setup dynamically (when requirement comes) to an authorized subset of nodes in the network: some protocol modifications may be necessary to allow this behavior.

4. Comparison

The following table tries to summarize the various solution pros/cons within a comparison table:

- Solution 1: using adjacency-SIDs only
- Solution 2: using adjacency-SIDs + Node-SIDs with strict SPF algorithm
- Solution 3: using adjacency-SIDs + Node-SIDs with new protection flag
- Solution 4: using adjacency-SIDs + two regular Node-SIDs with a different policy
- Solution 5: using adjacency-SIDs + Binding-SIDs

We consider a network with N nodes and L links, with an average of l links per node.

Criteria	Soluti	Solution	Solution 3	Solution 4	Solutio

	on 1	2			n 5
Label stack size	One label per hop	Reduced	Reduced	Reduced	Reduced
Controlplane	Negligible	Potential additional computation + 2*N entries in RIB	+ 2*N entries in RIB	+ 2*N entries in RIB	Adds states in the LSRs
Dataplane	+l entries	+2*N entries	+2*N entries	+2*N entries	Variable
IP convergence time	None	Double	Double	Double	None
Computation engine	Needs to select Adj-SIDs with B=0	Needs to select Adj-SIDs with B=0 and Node-SIDs with strict SPF	Needs to select Adj-SIDs with B=0 and Node-SIDs with B=0	Needs to select Adj-SIDs with B=0 and needs to understand policy from the SP to select the right Node-SIDs	Needs to select Adj-SIDs with B=0 and place the binding SID in a smart way
Protocol	None	None	Need a new flag	None	None
ECMP avoidance	Supported	Supported at the price of increasing the label stack	Supported at the price of increasing the label stack	Supported at the price of increasing the label stack	Supported
Requirem	Yes	Partially	Partially	Partially	Yes

ents ful filment		(allows ECMP+transient rerouting)	(allows ECMP+transient rerouting)	(allows ECMP+transient rerouting)	
Others	None	None	None	None	Requires a controller with sessions to all nodes (even transit)

Comparison of solutions

5. Recommended option(s)

Based on the analysis in Section 4, we only have two solutions that fulfill the requirements expressed in Section 2: usage of adjacency-SIDs only, usage of a combination of adjacency SIDs and binding SIDs.

As using only Adjacency-SIDs may reduce today the possibility of creating a path (due to the hardware/software limitations), authors would like to encourage the usage of a combination of adjacency-SIDs and binding-SIDs (Section 3.3) as a short-term solution.

However this approach has also several drawbacks, but authors think that these drawbacks can be reduced by enhancing existing protocols.

As a long term solution, authors would like to encourage vendors to support the ability for a node to push a significant number of labels, up to the full network diameter.

6. Security Considerations

TBD.

7. Acknowledgements

Authors would like to thank Bruno Decraene for his valuable comments.

8. IANA Considerations

N/A

9. Normative References

[I-D.gredler-idr-bgp-ls-segment-routing-extension]
Gredler, H., Ray, S., Previdi, S., Filsfils, C., Chen, M., and J. Tantsura, "BGP Link-State extensions for Segment Routing", draft-gredler-idr-bgp-ls-segment-routing-extension-02 (work in progress), October 2014.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and j. jefftant@gmail.com, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-13 (work in progress), June 2017.

[I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions-18 (work in progress), July 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-12 (work in progress), June 2017.

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

Authors' Addresses

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Mustapha Aissaoui
Nokia

Email: mustapha.aissaoui@nokia.com