SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 7, 2016

C. Bowers
P. Sarkar
H. Gredler
Juniper Networks
U. Chunduri
Ericsson Inc

October 5, 2015

Advertising Per-Topology and Per-Algorithm Label Blocks
draft-bowers-spring-adv-per-algorithm-label-blocks-02

Abstract

When segment routing is used in a network that is controlled by a link state IGP (such as ISIS or OSPF), each node in the network can be assigned one or more index numbers, known as "node-SIDs". The node-SIDs are unique within the network, and are known to all the nodes in the network. If an ingress node has a data packet to be sent to an egress node, the ingress node may select a node-SID corresponding to the egress node, and "translate" that node-SID to an MPLS label. The MPLS label represents a particular path to the egress node; the path is determined by applying a routing algorithm to a particular view of the network topology and a particular set of metric assignments to the links of that topology. The packet can then be forwarded by pushing the label on the packet’s label stack and transmitting the packet to the next hop on the corresponding path to the egress node. This document compares two different procedures for translating a node-SID to the MPLS label that represents a path chosen by a particular algorithm operating on a particular topology. It also specifies the ISIS extensions needed to support one of the procedures (known as the "per-topology/per-algorithm label block" procedure).

This Internet-Draft will expire on April 7, 2016.

Table of Contents

1. Introduction

[I-D.ietf-spring-segment-routing] describes the segment routing architecture. When segment routing is used in a network that is controlled by a link state IGP (such as ISIS or OSPF), each node in the network can be assigned one or more index numbers, known as "node-SIDs". The node-SIDs are unique within the network, and are

known to all the nodes in the network. If an ingress node has a data packet to be sent to an egress node, the ingress node may select a node-SID corresponding to the egress node, and "translate" that node-SID to an MPLS label. The MPLS label represents a particular path to the egress node; the path is determined by applying a routing algorithm to a particular view of the network topology and a particular set of metric assignments to the links of that topology. The packet can then be forwarded by pushing the label on the packet's label stack and transmitting the packet to the next hop on the corresponding path to the egress node.

When a particular network is using a single routing algorithm and a single topology, the procedure for translating a node-SID to an MPLS label is straightforward. Figure 1 shows the formula used to translate a node-SID into an MPLS label when the paths are selected by using the default routing algorithm (Dijkstra's shortest path first algorithm) and the default topology.

```
SPF_Label(X,D) = Label_Block(X) + Node_Index(D)

D is the destination node
X is the next-hop along the path to D
```

Figure 1: Translating Node-SID to Label: The Default Case

As a simple example, when the computing node (Y) needs to forward a packet ultimately destined for node D, Y first determines the shortest path next-hop node to reach D, which in this example is X. Y then adds the Node_Index value advertised by D to the Label_Block value advertised by X to determine the label value to apply to the packet before sending it to X.

2. Destination-based forwarding using other algorithms

Figure 2 shows two options for generalizing the above formula, to determine locally significant labels corresponding to forwarding next-hops computed using other algorithms.

```
        Option 1a: per-algorithm node index
        Label(X,D,A) = Label_Block(X) + Node_Index(D,A)

        Option 2a: per-algorithm label block
        Label(X,D,A) = Label_Block(X,A) + Node_Index(D)

    A is the algorithm for computing destination-based
      forwarding next-hops
    D is the destination node
    X is the next hop along the path to D that is
      determined by algorithm A
```

Figure 2: Translating Node-SID to Label: Algorithm-Specific Options

Suppose router Y needs to forward a packet to node D along a path computed by algorithm A. Using either option, Y determines the next-hop computed by algorithm A to reach D, which in this example is X. Y then needs to figure out the correct label to apply to the packet so that so that X will also understand that the packet is to be sent to node D along a path computed by algorithm A. The two options shown in Figure 2 differ in how Y determines that label value.

In Option 1a each node advertises a single label block, but advertises a different node index for each algorithm. Y determines the label value of local significance to X to reach D using algorithm A by adding the Node_Index advertised by node D for algorithm A to the Label_Block advertised by node X. We refer to this as the per-algorithm node index option.

In Option 2a each node advertises only a single node index, but advertises a different label block for each algorithm. Y determines the label value of local significance to X to reach D using algorithm A by adding the Node_Index advertised by node D to the Label_Block for algorithm A advertised by node X. We refer to this as the per-algorithm label block option.

The extensions currently defined in [I-D.ietf-isis-segment-routing-extensions] and [I-D.ietf-ospf-segment-routing-extensions] specify encodings for Option 1a, the per-algorithm node index option. This draft proposes extensions that can be used to support option 2a, the per-algorithm label block option. However, before discussing those extensions, we generalize the formula in Figure 2 further to take into account multiple topologies. This will allow us to define extensions that address the use of both multiple topologies and multiple algorithms.

3. Multi-topology routing

The IGP extensions to support multi-topology routing are defined in [RFC4915] for OSPF and [RFC5120] for IS-IS. Figure 3 further generalizes the formulas above to take into account multiple topologies. It shows two options for determining locally significant labels for different topologies and algorithms.

```
Option 1: per-topology / per-algorithm node index
Label(X,D,T,A) = Label_Block(X) + Node_Index(D,T,A)

Option 2: per-topology / per-algorithm label block
Label(X,D,T,A) = Label_Block(X,T,A) + Node_Index(D)

T is the topology
A is the algorithm for computing destination-based
  forwarding next-hops
D is the destination node
X is the next hop along the path to D that is
  determined by algorithm A for topology T
```

Figure 3: Translating Node-SID to Label: Topology and Algorithm-Specific Options

In Option 1 each node advertises a single label block, but advertises a different node index for each combination of topology and algorithm used. In order for Y to determine the label value that tells X to reach D via the path chosen by algorithm A for topology T, Y adds the Node_Index advertised by node D for topology T and algorithm A to the Label_Block advertised by node X. We refer to this as the per-topology/per-algorithm node index option.

In Option 2 each node advertises a single node index and a unique label block along for each combination of topology and algorithm used. In order for Y to determine the label value that tells X to reach D via the path chosen by algorithm A for topology T, Y adds the Node_Index advertised by node D to the Label_Block advertised by node X for topology T and algorithm A. We refer to this as the per-topology/per-algorithm label block option.

Note that the formulas in Figure 3 can of course be applied even if there is only one algorithm and/or only one topology. For example, if the use case uses multiple topologies but only uses the default shortest path algorithm (algorithm=0), then option 2 can be written as: Label(X,D,T,0) = Label_Block(X,T,0) + Node_Index(D), which is

independent of algorithm. Similarly, if the use case only uses the default topology (topology=0) but uses different algorithms, then option 2 can be written as Label(X,D,0,A) = Label_Block(X,0,A) + Node_Index(D).

4. Example: Adding Nodes when Multiple Algorithms are In Use

The following example illustrates the practical difficulties associated with using the per-topology/per-algorithm node index option alone (option 1 in Figure 3). This example is intentionally simplified to illustrate the need for some kind of convention to manage the assignment of the unique node index values required by option 1, even in a simple scenario. The sections below discuss a more complex example, as well as a specific proposal to manage the assignment of unique node index values. This simplified example assumes that the operator does not use multi-topology routing, i.e. that the default topology is used.

Suppose an operator has a network with 100 nodes, which we will refer to as R0-R99. The operator assigns the unique node index values 0-99 to those nodes for algorithm=0, in order to accomplish shortest path routing based on IGP metrics with SR labels. Each node will need to advertise a label block of size=100.

Assume that at some future point in time, the IETF defines algorithm=2 to mean shortest path routing based on latency, and vendors implement this. (See section Section 10 for more discussion of this example.) Suppose that the operator wants to use latency-based SPF routes for some traffic and metric-based SPF routes for other traffic. The operator will need to define a new set of unique node index values for algorithm=2. A reasonable choice would be to assign node index values of 100-199 to R0-R99 for algorithm=2. Each node will now need to advertise a label block of size=200. So far the need for per-algorithm node index values is an annoyance, but not too difficult to deal with.

Now assume that the operator needs to add 10 new nodes to the SR domain, specifically nodes R100-R109. Each node will now need to advertise a label block of size=220. The main issue is deciding how to assign per-algorithm node index for the 10 new nodes. One option is to redo the node index numbering scheme so that R0-R109 have node index values 0-109 for algorithm=0 and node index values 110-229 for algorithm=2. However, this requires renumbering existing nodes. The other option is to avoid renumbering of nodes by assigning nodes R100-R109 node index values 200-209 for algorithm=0 and node index values 210-219 for algorithm=1. Each of these approaches has drawbacks. The first requires renumbering existing nodes, while the

second is difficult to maintain since there is no obvious relationship between the node index values for different algorithms.

In order to reduce the complexity associated with option 1 in this simple example, a certain amount of pre-planning together with some convention for assigning node index values to algorithms or topologies would be useful. Specific proposals for managing unique node index values when using option 1 are discussed below. First however, we illustrate the advantages of option 2 for this simple example.

The use of per-algorithm label blocks avoids the problems associated with assigning and maintaining unique node index values for each forwarding algorithm.

When the SR domain is initially deployed, R0-R99 can be assigned node index values 0-99, as one would expect. When support for algorithm=1 gets added, the operator does not need to assign and configure any new node index values. Instead, the routers automate the process by advertising different label blocks for each forwarding algorithm.

When another 10 nodes are added to the SR domain, R100-R109 get assigned node index values 100-109 as one would expect. And the router advertises a label block of size=110 for each algorithm, as one would expect. Adding new nodes in the presence of multiple forwarding algorithms is simplified significantly with the use of per-algorithm label blocks.

5. Proposed configured offset mapping method for assigning per-topology/per-algorithm node-SIDs when using Option 1

If a network operator uses option 1, which requires the assignment of unique per-topology/per-algorithm node-SIDs, then it is clear that a common convention or methodology would be useful to help assign and maintain those unique node-SIDs. The methodology described in this section represents the authors' understanding of a proposal to manage assignment of node-SIDs when using option 1, as discussed on the SPRING mailing list.

The proposed method for managing the assignment of unique node index values for each topology/algorithm pair involves configuring a mapping from each topology/algorithm pair to an offset value. This offset mapping would need to be configured identically on every router in the network. Figure 4 shows the formula for a router Y to compute its own unique node index value for each topology/algorithm pair. Y would then treat those computed node index values as if they were directly configured via CLI or via Netconf/Yang, advertising

them into the IGP and installing the appropriate label operations in the FIB.

```
Node_Index(Y,T,A) = Configured_Offset(T,A) + Base_Node_Index(Y)

Y is the computing router
T is the topology
A is the algorithm
```

Figure 4: Proposed configured offset mapping method to manage assignment of unique per-topology/per-algorithm node index values when using Option 1

We illustrate the operation of the configured offset mapping method with a specific example. In this example, the operator has a network with 500 nodes, and wants to support four different topologies using different algorithms. The default topology (topology=0) needs to support algorithms 0, 4, and 5. Topology 2 and topology 6 need to support algorithm 0, while topology 7 needs to support algorithm 2. There are a total of six topology/algorithm pairs. In order to avoid renumbering the network in the event of unanticipated increases in the number of nodes or the number of topology/algorithm pairs, the operator sizes the label offsets and overall label block size to accomodate 1000 nodes and 12 topology/algorithm pairs.

Figure 5 shows the configuration data required on each of the 500 routers using option 1 together with the configured offset mapping method to manage node index assignment.

```
base_node_index=123
label_block_size=12000
topology=0 algorithm=0 offset=0
topology=0 algorithm=4 offset=1000
topology=0 algorithm=5 offset=2000
topology=2 algorithm=0 offset=3000
topology=6 algorithm=0 offset=4000
topology=7 algorithm=2 offset=5000
```

Figure 5: Required configuration data using option 1

The base_node_index value is the unique node index for a given node, and will thus be different for each node. The other values define the overall size of the label block and associate topology algorithm pairs with an offset value. This set of values must be configured identically across all routers in the network in order avoid advertising duplicate node index values. Advertisement of duplicate

node index values would disrupt forwarding. The configuration above would result in R123 computing node index values of 123, 1123, 2123, 3123, 4123, and 5123 for the corresponding topology/algorithm pairs.

For comparison, Figure 6 shows the configuration data required on each of the 500 routers using option 2. Since the per-topology/per-algorithm label blocks are advertised independently by each node, option 2 requires no additional configuration beyond what is required for default topology shortest path forwarding (topology=0, algorithm=0).

```
node_index=123
label_block_size=1000
```

Figure 6: Required configuration data using option 2

6. Flexibility to create easy-to-interpret label values

For some applications, it may be desirable to arrange things so that the meaning of label values used for forwarding can be readily understood by people trouble-shooting the network. When using the configured offset mapping method with option 1, if one configures a meaningful base value for the single label block, then the configured offset values can also be chosen to provide understandable label values. In the example above with 500 nodes and 6 topology/algorithm pairs, if the single logically advertised label block consists of a single numerically contiguous label block from 20000 through 31999 across all routers in the network, then the label values corresponding to forwarding to R123 using different topology/algorithm pairs will be meaningful to a people. They will be 20123, 21123, 22123, 23123, 24123, and 25123 for the corresponding topology/algorithm pairs, so an operator who remembers the mapping between topology/algorithm pair and offset can tell that 25123 is the label corresponding to topology=7, algorithm=2, node=123.

When using option 2 (per-topology/per-algorithm label blocks) and requiring that the topology, algorithm, and node associated with a label value be easy to interpret, each topology/algorithm pair needs to have an associated label_block_base configured on every router. Figure 7 show an example configuration of a mapping from topology/algorithm pairs to label_block_base values.

```
node_index=123
label_block_size=1000
topology=0 algorithm=0 label_block_base=100000
topology=0 algorithm=4 label_block_base=104000
topology=0 algorithm=5 label_block_base=105000
topology=2 algorithm=0 label_block_base=120000
topology=6 algorithm=0 label_block_base=160000
topology=7 algorithm=2 label_block_base=172000
```

Figure 7: Configuration data for 500 node example with option 2,

Note in this example that we have taken advantage of the additional flexibility of option 2 to create label values that are more readable than from option 1. In this example, a first digit of "1" indicates that this is a SPRING node label. The second and third digits are readable as the topology and algorithm, while the last three digits encode the node number. So 172123 would indicate the node label for topology=7, algorithm=2, node=123.

In the above example, we have illustrated the flexibility of option 2 to create more readable labels in a hypothetical network with no constraints on label space. However, it is likely that in a multi-vendor network with multiple generations of hardware supporting different MPLS applications there will exist constraints regarding the location and size of contiguous label blocks for use by SPRING. This would impose constraints on one’s ability to construct readable label values using option 1 with the configured offset mapping. Option 2 provides more flexibility to construct easy-to-interpret label values in such a network.

7. Robustness against misconfiguration

Option 2 is much more robust against misconfiguration than is option 1. This is true both in scenarios that require easy-to-interpret label values and in scenarios that do not.

In the simple case where the application does not require easy-to-interpret label values, option 2 has clear advantages over option 1 in terms of robustness again misconfiguration. Option 1 requires identical offset mapping configurations on all routers for proper forwarding. Option 2 requires no configuration, so there is nothing to misconfigure.

In scenarios requiring easy-to-interpret label values, where option 2 requires a label_block_base mapping configuration, option 2 is still more robust against misconfiguration than option 1. Misconfiguration of the label_block_base mapping in option 2 does not affect forwarding. The explicit advertisement of the per-topology/per-

algorithm label blocks ensures that forwarding will continue to work properly.

8. ISIS extensions to encode per-topology/per-algorithm label blocks

Below is a concrete proposal for encoding per-topology/per-algorithm label blocks in ISIS compatible with the encodings in [I-D.ietf-isis-segment-routing-extensions].

The newly-defined Topology-Algorithm-Label-Block sub-TLV is shown in Figure 8. It is carried in the IS-IS Router Capability TLV-242. It contains a 12-bit MT-ID field as well as an 8-bit Algorithm field which associates the SRGB Descriptor entries carried in the sub-TLV with a particular topology and algorithm. Otherwise, the structure and interpretation of the Topology-Algorithm-Label-Block sub-TLV is identical to that of the SR-Capabilities sub-TLV defined in section 3.1 of [I-D.ietf-isis-segment-routing-extensions].

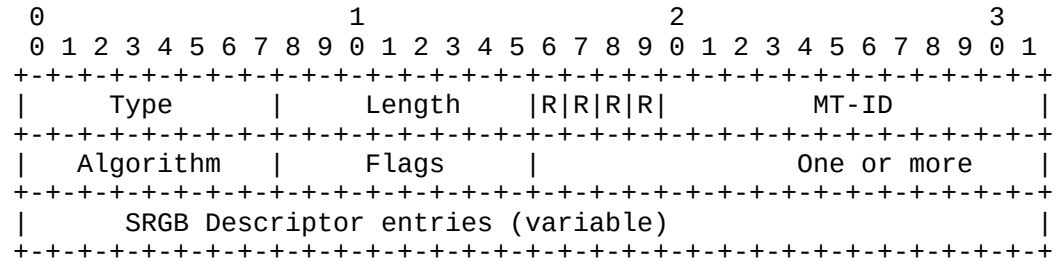


Figure 8: Topology-Algorithm-Label-Block sub-TLV

A single network must use either option 1 or option 2 for all routers. Mixed mode operation is not supported. When the Topology-Algorithm-Label-Block sub-TLV is present with a given pair of topology and algorithm values, routers MUST determine the label values associated with that topology/algorithm pair using the per-topology/per-algorithm label block method and the concatenated label block carried by the Topology-Algorithm-Label-Block sub-TLV. The node indices used in this calculation are those carried in Node-SID advertisements with algorithm value=0 in TLV-135(IPv4) or TLV-236(IPv6).

The Topology-Algorithm-Label-Block sub-TLV MUST NOT be advertised with both MT-ID=0 and Algorithm value=0. In this way, the concatenated label block used to compute the label values for the default topology and algorithm=0 can only be carried by the SR Capabilities sub-TLV.

When using the Topology-Algorithm-Label-Block sub-TLV in a network, nodes SHOULD only advertise a node index value corresponding to algorithm=0 in Node-SID advertisements in TLV-135(IPv4) and/or TLV-236(IPv6). Node index values (with algorithm=0 or any other value) SHOULD NOT be advertised in TLV-235(MT-IPv4) and TLV-237(MT-IPv6). If a node originates the Topology-Algorithm-Label-Block sub-TLV (meaning that it supports option 2), then it MUST ignore the receipt of node indices for non-zero algorithms in TLV-135 and TLV-236 and any node index values in TLV-235 and TLV-237.

9. OSPF extensions to encode per-topology/per-algorithm label blocks

OSPF extensions to encode per-topology/per-algorithm label blocks will be provided in a future version of this draft.

10. A note on algorithms and topologies

The example given in Section 4 supposes that at some point in the future the IETF defines algorithm=2 to mean shortest path routing based on latency. This simple example was chosen since it is easy to understand. However, the same result could also have been achieved by defining a second topology which uses latency as the metric for that topology, and running the default SPF algorithm on that second topology.

In general, when using other algorithms for computing next-hops for destination-based forwarding, it is not possible to achieve the same results by simply defining a new topology with modified metrics and running the default SPF algorithm. An example of such an algorithm is that used to compute Maximally Redundant Trees (MRTs), as defined in [I-D.ietf-rtgwg-mrt-frr-algorithm].

11. IANA Considerations

This document requests the following registration in the "sub-TLVs for TLV 242" registry.

Value: TBA (suggested value 20)

Description: Topology-Algorithm-Label-Block

Reference: This document (Section 8)

12. Management Considerations

This document proposes the use of per-topology/per-algorithm label blocks (option 2) to support destination-based forwarding along next-hops computed using different algorithms for different topologies.

The automated advertisement of per-topology/per-algorithm label blocks significantly simplifies network management compared to configuration and maintenance of unique per-topology/per-algorithm node indices.

13. Security Considerations

TBD

14. Acknowledgements

The authors thank John Scudder and Eric Rosen for their helpful review of this document and suggestions.

15. References

15.1. Normative References

[RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, DOI 10.17487/RFC4915, June 2007, <http://www.rfc-editor.org/info/rfc4915>.

[RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <http://www.rfc-editor.org/info/rfc5120>.

15.2. Informative References

[I-D.ietf-isis-segment-routing-extensions] Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-05 (work in progress), June 2015.

[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-05 (work in progress), June 2015.

[I-D.ietf-rtgwg-mrt-frr-algorithm] Envedi, G., Csaszar, A., Atlas, A., Bowers, C., and A. Gopalan, "Algorithms for computing Maximally Redundant Trees for IP/LDP Fast- Reroute", draft-ietf-rtgwg-mrt-frr-algorithm-05 (work in progress), July 2015.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and r. rjs@rob.sh, "Segment Routing Architecture", draft-ietf-spring-segment-routing-04 (work in progress), July 2015.

Authors' Addresses

Chris Bowers
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US

Email: cbowers@juniper.net

Pushpasis Sarkar
Juniper Networks
Embassy Business Park
Bangalore, KA 560093
India

Email: psarkar@juniper.net

Hannes Gredler
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US

Email: hannes@juniper.net

Uma Chunduri
Ericsson Inc
300 Holger Way
San Jose, CA 95134
US

Email: uma.chunduri@ericsson.com@

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 2, 2019

C. Filsfils, Ed.
S. Previdi
Cisco Systems, Inc.
G. Dawra, Ed.
LinkedIn
W. Henderickx
Nokia
D. Cooper
Level 3
August 29, 2018

Interconnecting Millions Of Endpoints With Segment Routing
draft-filsfils-spring-large-scale-interconnect-12

Abstract

This document describes an application of Segment Routing to scale the network to support hundreds of thousands of network nodes, and tens of millions of physical underlay endpoints. This use-case can be applied to the interconnection of massive-scale DCs and/or large aggregation networks. Forwarding tables of midpoint and leaf nodes only require a few tens of thousands of entries.

Table of Contents

1. Introduction

This document describes how SR can be used to interconnect millions of endpoints.The following terminology is used in this document:

2. Terminology

The following terms and abbreviations are used in this document:

Term	Definition
Agg	Aggregation
BGP	Border Gateway Protocol
DC	Data Center
DCI	Data Center Interconnect
ECMP	Equal Cost MultiPathing
FIB	Forwarding Information Base
LDP	Label Distribution Protocol
LFIB	Label Forwarding Information Base
MPLS	Multi-Protocol Label Switching
PCE	Path Computation Element
PCEP	Path Computation Element Protocol
PW	Pseudowire
SLA	Service level Agreement
SR	Segment Routing
SRTE Policy	Segment Routing Traffic Engineering Policy
TE	Traffic Engineering
TI-LFA	Topology Independent - Loop Free Alternative

3. Reference Design

The network diagram here below describes the reference network topology used in this document:

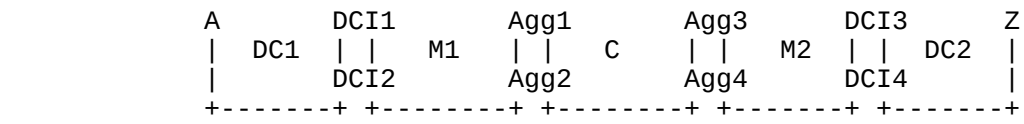


Figure 1: Reference Topology

The following applies to the reference topology above:

Independent ISIS-OSPF/SR instance in core (C) region.

Independent ISIS-OSPF/SR instance in Metro1 (M1) region.

Independent ISIS-OSPF/SR instance in Metro2 (M2) region.

BGP/SR in DC1.

BGP/SR in DC2.

Agg routes (Agg1, Agg2, Agg3, Agg4) are redistributed from C to M (M1 and M2) and from M to DC domains.

No other route is advertised or redistributed between regions.

The same homogeneous SRGB is used throughout the domains (e.g. 16000-23999).

Unique SRGB sub-ranges are allocated to each metro (M) and core (C) domains:

16000-16999 range is allocated to the core (C) domain/region.

17000-17999 range is allocated to the M1 domain/region.

18000-18999 range is allocated to the M2 domain/region.

Specifically, Agg1 router has SID 16001 allocated and Agg2 router has SID 16002 allocated.

Specifically, Agg3 router has SID 16003 allocated and the anycast SID for Agg3 and Agg4 is 16006.

Specifically, DCI3 router has SID 18003 allocated and the anycast SID for DCI3 and DCI4 is 18006.

Specifically, at Agg1 router Binding SID 4001 leads to DCI Pair DCI3, DCI4 via specific low-latency path {16002, 16003, 18006}.

The same SRGB sub-range is re-used within each DC (DC1 and DC2) region. for each DC: e.g. 20000-23999. Specifically, range 20000-23999 range is used in both DC1 and DC2 regions and nodes A and Z have both SID 20001 allocated to them.

4. Control Plane

This section provides a high-level description of a how a control plane could be implemented using protocol components already defined in other RFCs.

The mechanism through which SRTE Policies are defined, computed and programmed in the source nodes, are outside the scope of this document.

Typically, a controller or a service orchestration system programs node A with a pseudowire (PW) to a remote next-hop Z with a given SLA contract (e.g. low-latency path, be disjoint from a specific core plane, be disjoint from a different PW service, etc.).

Node A automatically detects that it does not have reachability to Z. It then automatically sends a PCEP request to an SR PCE for an SRTE policy that provides reachability to Z with the requested SLA.

The SR PCE [RFC4655] is made of two components. A multi-domain topology and a computation engine. The multi-domain topology is continuously refreshed through BGP-LS [RFC7752] feeds from each domain. The computing engine implements Traffic Engineering (TE) algorithms designed specifically for SR path expression. Upon receiving the PCEP [RFC5440] request, the SR PCE computes the requested path. The path is expressed through a list of segments (e.g. {16003, 18006, 20001} and provided to node A.

The SR PCE logs the request as a stateful query and hence is capable to recompute the path at each network topology change.

Node A receives the PCEP reply with the path (expressed as a segment list). Node A installs the received SRTE policy in the dataplane. Node A then automatically steers the PW into that SRTE policy.

5. Illustration of the scale

According to the reference topology described in Figure 1 the following assumptions are made:

There’s 1 core domain and 100 of leaf (metro) domains.

The core domain includes 200 nodes.

Two nodes connect each leaf (metro) domain. Each node connecting a leaf domain has a SID allocated. Each pair of nodes connecting a leaf domain also has a common anycast SID. This brings up to 300 prefix segments in total.

A core node connects only one leaf domain.

Each leaf domain has 6000 leaf node segments. Each leaf-node has 500 endpoints attached, thus 500 adjacency segments. In total, it is 3 millions endpoints for a leaf domain.

Based on the above, the network scaling numbers are as follows:

6,000 leaf node segments multiplied by 100 leaf domains: 600,000 nodes.

600,000 nodes multiplied by 500 endpoints: 300 millions of endpoints.

The node scaling numbers are as follows:

Leaf node segment scale: 6,000 leaf node segments + 300 core node segments + 500 adjacency segments = 6,800 segments

Core node segment scale: 6,000 leaf domain segments + 300 core domain segments = 6,300 segments

In the above calculations, the link adjacency segments are not taken into account. These are local segments and, typically, less than 100 per node.

It has to be noted that, depending on leaf node FIB capabilities, leaf domains could be split into multiple smaller domains. In the above example, the leaf domains could be split into 6 smaller domains so that each leaf node only need to learn 1000 leaf node segments + 300 core node segments + 500 adjacency segments which gives a total of 1,800 segments.

6. Design Options

This section describes multiple design options to the illustration of previous section.

6.1. Segment Routing Global Block(SRGB) Size

In the simplified illustrations of this document, we picked a small homogeneous SRGB range of 16000-23999. In practice, a large-scale design would use a bigger range such as 16000-80000, or even larger. Larger range provides allocations for various Traffic Engineering applications within a given domain

6.2. Redistribution of Agg nodes routes

The operator might choose to not redistribute the Agg nodes routes into the Metro/DC domains. In that case, more segments are required in order to express an inter-domain path.

For example, node A would use an SRTE Policy {DCI1, Agg1, Agg3, DCI3, Z} in order to reach Z instead of {Agg3, DCI3, Z} in the reference design.

6.3. Sizing and hierarchy

The operator is free to choose among a small number of larger leaf domains, a large number of small leaf domains or a mix of small and large core/leaf domains.

The operator is free to use a 2-tier design (Core/Metro) or a 3-tier (Core/Metro/DC).

6.4. Local Segments to Hosts/Servers

Local segments can be programmed at any leaf node (e.g. node Z) in order to identify locally-attached hosts (or VM's). For example, if node Z has bound a local segment 40001 to a local host ZH1, then node A uses the following SRTE Policy in order to reach that host: {16006, 18006, 20001, 40001}. Such local segment could represent the NID (Network Interface Device) in the context of the SP access network, or VM in the context of the DC network.

6.5. Compressed SRTE policies

As an example and according to Section 3, we assume node A can reach node Z (e.g., with a low-latency SLA contract) via the SRTE policy consisting of the path: Agg1, Agg2, Agg3, DCI3/4(anycast), Z. The path is represented by the segment list: {16001, 16002, 16003, 18006, 20001}.

It is clear that the control-plane solution can install an SRTE Policy {16002, 16003, 18006} at Agg1, collect the Binding SID allocated by Agg1 to that policy (e.g. 4001) and hence program node A with the compressed SRTE Policy {16001, 4001, 20001}.

From node A, 16001 leads to Agg1. Once at Agg1, 4001 leads to the DCI pair (DCI3, DCI4) via a specific low-latency path {16002, 16003, 18006}. Once at that DCI pair, 20001 leads to Z.

Binding SID's allocated to "intermediate" SRTE Policies allow to compress end-to-end SRTE Policies.

The segment list {16001, 4001, 20001} expresses the same path as {16001, 16002, 16003, 18006, 20001} but with 2 less segments.

The Binding SID also provide for an inherent churn protection.

When the core topology changes, the control-plane can update the low-latency SRTE Policy from Agg1 to the DCI pair to DC2 without updating the SRTE Policy from A to Z.

7. Deployment Model

It is expected that this design be deployed as a green field but as well in interworking (brown field) with MPLS design across multiple domains.

8. Benefits

The design options illustrated in this document allow the interconnection on a very large scale. Millions of endpoints across different domains can be interconnected.

8.1. Simplified operations

Two protocols have been removed from the network: LDP and RSVP-TE. No new protocol has been introduced. The design leverage the core IP protocols: ISIS, OSPF, BGP, PCEP with straightforward SR extensions.

8.2. Inter-domain SLA

Fast reroute and resiliency is provided by TI-LFA with sub-50msec FRR upon Link/Node/SRLG failure. TI-LFA is described in [I-D.bashandy-rtgwg-segment-routing-ti-lfa].

The use of anycast SID’s also provide an improvement in availability and resiliency.

Inter-domain SLA’s can be delivered, e.g., latency vs. cost optimized path, disjointness from backbone planes, disjointness from other services, disjointness between primary and backup paths.

Existing inter-domain solutions do not provide any support for SLA contracts. They just provide a best-effort reachability across domains.

8.3. Scale

In addition to having eliminated two control plane protocols, per-service midpoint states have also been removed from the network.

8.4. ECMP

Each policy (intra or inter-domain, with or without TE) is expressed as a list of segments. Since each segment is optimized for ECMP, then the entire policy is optimized for ECMP. The ECMP gain of anycast prefix segment should also be considered (e.g. 16001 load-shares across any gateway from M1 leaf domain to Core and 16002 load-shares across any gateway from Core to M1 leaf domain.

9. IANA Considerations

This document does not make any IANA request.

10. Manageability Considerations

This document describes an application of Segment Routing over the MPLS data plane. Segment Routing does not introduce any change in the MPLS data plane. Manageability considerations described in [RFC8402] apply to the MPLS data plane when used with Segment Routing.

11. Security Considerations

This document does not introduce additional security requirements and mechanisms other than the ones described in [RFC8402].

12. Acknowledgements

We would like to thank Giles Heron, Alexander Preusche, Steve Braaten and Francis Ferguson for their contribution to the content of this document.

13. Contributors

The following people have substantially contributed to the editing of this document:

Dennis Cai
Individual

Tim Laberge
Individual

Steven Lin
Google Inc.

Steven Lin
Google Inc.

Bruno Decraene
Orange

Luay Jalil
Verizon

Jeff Tantsura
Individual

Rob Shakir
Google

14. Informative References

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

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <https://www.rfc-editor.org/info/rfc4655>.

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

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

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

Authors' Addresses

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

Email: cfilsfil@cisco.com

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

Email: stefano@previdi.net

Gaurav Dawra (editor)
LinkedIn
USA

Email: gdawra.ietf@gmail.com

Wim Henderickx
Nokia
Copernicuslaan 50
Antwerp 2018
Belgium

Email: wim.henderickx@nokia.com

Dave Cooper
Level 3

Email: Dave.Cooper@Level3.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2015

S. Litkowski
Orange Business Service
A. Lindem
Cisco Systems
P. Sarkar
Juniper Networks
I. Chen
Ericsson
March 05, 2015

YANG Data Model for Segment Routing
draft-litkowski-spring-sr-yang-00

Abstract

This document defines a YANG data model for segment routing configuration and operation. This YANG model is intended to be used on network elements to configure or operate 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 September 6, 2015.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG data model for segment routing configuration and operation.

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

This is the initial version of this module and and its relationship to the protocol modules. It is expected that there will be changes as the module matures.

```
module: ietf-segment-routing
augment /rt:routing/rt:routing-instance:
   +--rw segment-routing
      +--rw transport-type?   identityref
      +--rw bindings
      |  +--rw mapping-server {mapping-server}?
      |     +--rw ipv4
      |     |  +--rw mapping-entry* [prefix]
      |     |     +--rw prefix       inet:ipv4-prefix
      |     |     +--rw start-sid?   uint32
      |     |     +--rw range?       uint32
      |     +--rw ipv6
      |        +--rw mapping-entry* [prefix]
      |           +--rw prefix       inet:ipv6-prefix
      |           +--rw start-sid?   uint32
      |           +--rw range?       uint32
      +--rw srgb* [lower-bound upper-bound]
      |  +--rw lower-bound    uint32
      |  +--rw upper-bound    uint32
      +--rw interfaces
         +--rw interface* [name]
            +--rw name             if:interface-ref
            +--rw adjacency-sid
            |  +--rw advertise-adj-group-sid* [group-id]
            |  |  +--rw group-id    uint32
            |  +--rw advertise-protection?      enumeration
            +--rw prefix-sid
               +--rw ipv4
               |  +--rw prefix-sid* [value]
               |     +--rw value-type?          enumeration
```

```
               |     +--rw value                uint32
               |     +--rw node-flag?           boolean
               |     +--rw last-hop-behavior?   enumeration
               +--rw ipv6
                  +--rw prefix-sid* [value]
                     +--rw value-type?          enumeration
                     +--rw value                uint32
                     +--rw node-flag?           boolean
                     +--rw last-hop-behavior?   enumeration
augment /rt:routing/rt:routing-instance/rt:routing-protocols/rt:routing-protocol
/isis:isis/isis:instance:
   +--rw segment-routing
      +--rw enabled?    boolean
      +--rw bindings
         +--rw advertise?   boolean
         +--rw receive?     boolean
augment /rt:routing/rt:routing-instance/rt:routing-protocols/rt:routing-protocol
/ospf:ospf/ospf:instance:
   +--rw segment-routing
      +--rw enabled?    boolean
      +--rw bindings
         +--rw advertise?   boolean
         +--rw receive?     boolean
augment /rt:routing-state/rt:routing-instance:
   +--ro segment-routing
      +--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    leafref
            +--ro binding-type       enumeration
notifications:
   +---n segment-routing-global-sid-collision
   |  +--ro received-target?    string
   |  +--ro original-target?    string
   |  +--ro index?              uint32
   |  +--ro routing-protocol?   leafref
   +---n segment-routing-index-out-of-range
      +--ro received-target?    string
      +--ro received-index?     uint32
      +--ro routing-protocol?   leafref
```

3. Configuration

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

The segment-routing configuration is split in global routing-instance 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 how external information is mapped to a segment ID. The current version supports a mapping-server where static prefix-to-SID bindings can be defined. Configuration of bindings does not allow advertisement of those bindings. Advertisement must be controlled by each routing-protocol instance.

- 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.

The interface configuration includes :

- Adjacency SID properties

- Prefix SID properties

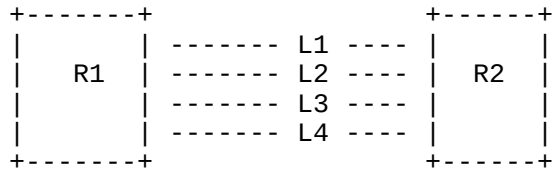
3.1. Adjacency SID properties

3.1.1. 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.

3.1.2. 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.

3.2. Prefix SID properties

An interface may have associated IP prefixes. By default, no Prefix-SID will be advertised for any IP prefix associated with an interface.

The operator can control the advertisement of IP prefixes by setting "prefix-sid" in the interface configuration.

The operator can control advertisement of Prefix-SID independently for IPv4 and IPv6. When specified, the "prefix-sid" value must be included.

The value can be expressed as an index (default), or an absolute value. The operator can also control if the "node-flag" is set for the prefix. As the network device owns the prefix, the default is to advertise the prefix with the "node-flag" set.

The "last-hop-behavior" configuration dictates the PHP behavior: "explicit-null", "php", or "non-php".

4. Control plane configuration

Activation of segment-routing extensions for a particular control plane is done by augmenting routing-protocol configuration with segment-routing.

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.

This model supports ISIS ([I-D.ietf-isis-segment-routing-extensions]) and OSPF as controlplane ([I-D.ietf-ospf-segment-routing-extensions] and [I-D.psenak-ospf-segment-routing-ospfv3-extension]) for segment-routing.

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 proposes two notifications for segment-routing.

- 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).

o 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@2015-03-04.yang"

```
module ietf-segment-routing {
  namespace "urn:ietf:params:xml:ns:"
          + "yang:ietf-segment-routing";
  prefix sr;

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

  import ietf-routing {
    prefix "rt";
  }

  import ietf-interfaces {
  prefix "if";
  }

  import ietf-isis {
  prefix "isis";
  }

  import ospf {
  prefix "ospf";
  }

  organization
   "IETF SPRING Working Group";

  contact
    "WG List:  &lt;mailto:spring@ietf.org&gt;

    Editor:    Stephane Litkowski
          &lt;mailto:stephane.litkowski@orange.com&gt;

        Acee Lindem
          &lt;mailto:acee@cisco.com&gt;
        Pushpasis Sarkar
          &lt;mailto:psarkar@juniper.net&gt;
        Ing-Wher Chen
          &lt;mailto:ing-wher.chen@ericsson.com&gt;
```

```
    ";

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

  revision 2015-02-27 {
    description "Initial";
    reference "draft-litkowski-spring-sr-yang-00";
  }

  /* 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.";
  }

  /* Features */

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

  /* Groupings */

  grouping controlplane-cfg {
    container segment-routing {
      leaf enabled {
        type boolean;
        default false;
        description
         "Enables segment-routing
         protocol extensions.";
      }
      container bindings {
        leaf advertise {
          type boolean;
          default true;
          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 prefix-sid-cfg {
      list prefix-sid {
        key value;
        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.";
        }

        leaf value {
          type uint32;
```

```
          mandatory true;
          description
          "Value associated with
          prefix. The value must
          be interpreted in the
          context of value-type.";
        }
        leaf node-flag {
          type boolean;
          default true;
          description
          "Set prefix as a node
           representative prefix.";
        }
        leaf 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
         "List of prefix SID.";
      }
      description
       "This grouping defines cfg of prefix SID.";
    }

    /* Cfg */

    augment "/rt:routing/rt:routing-instance" {
      description
       "This augments routing-instance
       configuration with segment-routing.";
      container segment-routing {
        leaf transport-type {
          type identityref {
```

```
            base segment-routing-transport;
          }
          default "segment-routing-transport-mpls";
          description "Dataplane to be used.";
        }
        container bindings {
          container mapping-server {
            if-feature mapping-server;
            container ipv4 {
              list mapping-entry {
                key prefix;

                leaf prefix {
                  type inet:ipv4-prefix;
                  description
                   "Base prefix used for mapping.";
                }
                leaf start-sid {
                  type uint32;
                  description
                   "Starting SID value to be associated
                   with prefix.";
                }
                leaf range {
                  type uint32;
                  description
                   "Describes how many SIDs could be
                   allocated.";
                }
                description
                 "Mapping entries.";
              }
              description
               "IPv4 mapping entries.";
            }
            container ipv6 {
              list mapping-entry {
                key prefix;

                leaf prefix {
                  type inet:ipv6-prefix;
                  description
                   "Base prefix used for mapping.";
                }
                leaf start-sid {
                  type uint32;
                  description
                   "Starting SID value to be associated
```

```
                   with prefix.";
                }
                leaf range {
                  type uint32;
                  description
                   "Describes how many SIDs could be
                   allocated.";
                }
                description
                 "Mapping entries.";
              }
              description
               "IPv6 mapping entries.";
            }
            description
             "Configuration of mapping-server
              local entries.";
          }
          description
           "List of bindings.";
        }
        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.";
        }

        container interfaces {
          list interface {
            key "name";
            leaf name {
              type if:interface-ref;

              description
               "Reference to the interface within
                the routing-instance.";
```

```
            }
            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.";
            }
            container prefix-sid {
```

```
              container ipv4 {
                uses prefix-sid-cfg;
                description
                 "Parameters associated with IPv4 prefix SID";
              }
              container ipv6 {
                uses prefix-sid-cfg;
                description
                 "Parameters associated with IPv6 prefix SID";
              }
              description
               "Prefix SID configuration.";
            }
            description
             "List of interfaces.";
          }
          description
           "Interface configuration.";
        }
        description
        "segment routing global config.";
      }
    }


    augment "/rt:routing/rt:routing-instance/" +
          "rt:routing-protocols/rt:routing-protocol"+
          "/isis:isis/isis:instance" {
      when "rt:type = 'isis:isis'" {
        description
        "This augment ISIS routing protocol when used";
      }
      description
       "This augments ISIS protocol configuration
        with segment routing.";

      uses controlplane-cfg;
    }

    augment "/rt:routing/rt:routing-instance/rt:routing-protocols" +
          "/rt:routing-protocol/ospf:ospf/ospf:instance" {
      when "rt:type = 'ospf:ospfv2' or rt:type = 'ospf:ospfv3'" {
        description
        "This augment ISIS routing protocol when used";
      }
      description
       "This augments ISIS protocol configuration
        with segment routing.";
```

```
      uses controlplane-cfg;
    }


    /* Operational states */

    augment "/rt:routing-state/rt:routing-instance" {
      description
       "This augments the operational states
        with segment-routing.";
      container segment-routing {
        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:routing-instance/" +
              "rt:routing-protocols/rt:routing-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-sid-collision {
    leaf received-target {
      type string;
      description
      "Target received in the controlplane that
             caused SID collision.";
    }
    leaf original-target {
      type string;
      description
      "Target already available in database that have the same SID
      as the received target.";
    }
    leaf index {
      type uint32;
      description
      "Value of the index used by two different prefixes.";
    }
    leaf routing-protocol {
      type leafref {
        path "/rt:routing-state/rt:routing-instance/" +
        "rt:routing-protocols/rt:routing-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:routing-instance/" +
            "rt:routing-protocols/rt:routing-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

TBD.

10. IANA Considerations

TBD.

11. Normative References

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-03 (work in progress), October 2014.

[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-04 (work in progress), February 2015.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., Shakir, R., Tantsura, J., and E. Crabbe, "Segment Routing Architecture", draft-ietf-spring-segment-routing-01 (work in progress), February 2015.

[I-D.psenak-ospf-segment-routing-ospfv3-extension]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPFv3 Extensions for Segment Routing", draft-psenak-ospf-segment-routing-ospfv3-extension-02 (work in progress), July 2014.

[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.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Authors' Addresses

Stephane Litkowski
Orange Business Service

Email: stephane.litkowski@orange.com

Acee Lindem
Cisco Systems

Email: acee@cisco.com

Pushpasis Sarkar
Juniper Networks

Email: psarkar@juniper.net

Ing-Wher Chen
Ericsson

Email: ing-wher.chen@ericsson.com

SPRING WG
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

Ting. Liao
Bo. Wu
Fangwei. Hu
ZTE Corporation
Bhumip. Khasnabish
ZTE TX Inc.

July 6, 2015

SPRING SID Allocation
draft-lw-spring-sid-allocation-02.txt

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). And a segment is identified by a Segment Routing ID (SID). This document proposes a method to reduce the SID configuration in a SR domain. Only the selected SR nodes which named Segment Routing Management Nodes (SRMNs) are configured by NMS, while other SRs in the domain need zero-SR-configuration.

Table of Contents

1. Introduction

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). And a segment is identified by a Segment Identifier (SID). A segment can be encoded as a MPLS label or an IPv6 address. Typically a SID is configured by NMS and it very rarely changes. When the SID configured, each node could send out its IGP TLV extensions which had described in [I-D.ietf-isis-segment-routing-extensions] and [I-D.ietf-ospf-segment-routing-extensions].

In some situation where users expect to simply plug in a SR node and have it automatically enable Segment Routing. One of the use cases is described in [I-D.kh-spring-ip-ran-use-case], with hundreds of CSGs in an IP RAN network, the configuration assigned by NMS could be very complexity. And with the complexity of the network such as nodes adding and removing, the management on NMS is a hard work. If the CSGs can plug and play, it will be very useful. And the CSGs

could be uploaded with zero-touch currently, by generating the ip address from MAC by default algorithm and with the ospf process default uploaded, have a mechanism to ensure the uniqueness of ip. Then the CSGs could be connected, and the topology will be collected by the ASGs. With the mechanism described in this draft, there is no SR configuration on the CSGs, the CSGs could be zero-touch still, reduce the complexity of SR related configuration and reduce the complexity of NMS.

This draft describes a mechanism to optimize SID allocation operation.

2. Conventions and Abbreviations

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] .

The following notations and abbreviations are used throughout this draft.

ASG: Aggregation Site/Service Gateway.

BS: Base Station.

CSG: Cell Site Gateway.

IP RAN: Internet Protocol RAN

RAN: Radio Access Network.

RNC: Radio Network Controller.

RSG: Radio Service Gateway.

SR: Segment Routing.

SID: Segment Identifiers.

3. Motivation

As described in [I-D.kh-spring-ip-ran-use-case] , the IP RAN network scenario is shown in the figure 1.

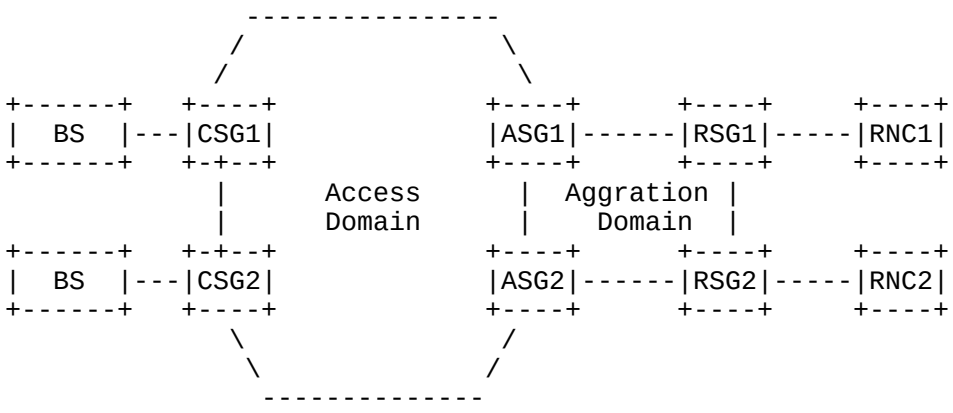


Figure 1 IP RAN Network Scenario

There are hundreds of CSGs (Cell Site Gateway) and a few sets of ASGs (Aggregation Site/Service Gateway) in the Access Domain of an IP RAN network. With the increase of mobile Internet traffic, more CSGs could be added to the Access Domain, and CSGs could be installed in wide area. So, there is a great need to do little or no configuration on CSGs to avoid configuration operation.

In current IP RAN implementation, as the CSGs could be uploaded with zero-touch, by generating the ip address from MAC by default algorithm and with the ospf process default uploaded, have a mechanism to ensure the uniqueness of ip. Then the CSGs could be connected, and the topology will be collected by the ASGs. ASGs are configured by NMS, and the configurations on CSGs are configured by ASGs, typical MPLS protocols like LDP protocol is used. With SR replacing LDP, complexity in LDP configuration could be greatly reduced. But in current SR process, each SR node should be configured by NMS, including the SR related configurations such as SIDs and SR algorithm and SR global blocks.

If the configuration on CSGs still need to be came from ASGs, a specific mechanism is proposed in this draft to fulfill this requirement. Like the network in the above figure, an ASG or both the ASGs could be selected as a SID management node to advertise CSGs and ASGs SID mapping to the whole SR domain.

4. SID generation and allocation

In the proposed mechanism, one or more Segment Routing Management Nodes (SRMNs) reside at SR nodes and advertise the SID mapping information for the various prefix associated with the SR nodes in the SR domain.

4.1. SID generation

The NMS configure the SRGB block to the SRMN. The SRMN will generate SIDs to other nodes'router-ids and to the router-id of itself, the generation principle based on the configuration of the NMS or the default generation rule, the default allocation rule MAY be based on the numerically higher router-id with the higher SID allocated.

4.2. SID allocation

As described in section 3, when the SR node is powdered on, the IGP protocol as default function loaded, each node flooding out each router information in ISIS LSP or OSPF LSA when the ISIS adjacency or OSPF adjacency relations formed, and every node will acquire the router-ids of other nodes from the information (such as the IP address of router id) of IGP TLV.

Then the SRMN generates the SIDs mapping to the router-ids and allocates the SIDs to each SR node by using the extension of SID Allocation TLV or the SID Binding TLV(as described in [I-D.ietf-isis-segment-routing-extensions] and [I-D.ietf-ospf-segment-routing-extensions].)in the IGP packet, flooding the packet out. In the SID Allocation TLV, it carries the router-id and SID mapping, and related algorithm type, and related multi-topology number. And in one SID-Allocation TLV, it can carry one or more IP addresses.

Optionally, If more than one SRMN assigned by the NMS, the SRMNs May flood out another extension which indicating having the capability to allocate SID, this extension is called the SR Allocation Ability extension and be included in the SR capabilities. When other SR nodes receive more than one of the SRMN advertised the extension, the SR node could decide how to choose the Allocated SID, the choosing principle is based on the value of SRMNs' router id or system id, the maximum or the minimum value is chosen, and the SID allocated by the maximum or minimum id's SRMN be chosen. When each SR node received the IGP packet with the SID Allocation TLV, it will know which SID is allocated to itself, and then the SR node sends out the prefix-SID sub-TLV in its IGP packet flooding out in the IGP area.

Then each SR node will know the other SR nodes' SID, and the related algorithm will calculate the path to each SR node's SID.

With the automatic uniform allocation by the SRMN in the IGP area, when a new node is added, the SRMN will know which SIDs had been allocated, and allocate an unused SID in the SRGB to the new node's IP address. And if the node has moved, the SRMN will delete the related SID to the moving node's IP address and recycle this SID.

The SID uniqueness is managed on the SRMN.

If more than one SRMN assigned by the NMS, the SR Allocation Ability extension will be used, the detail information is described in section 4.2.

5. IGP extension

The SID Allocation TLV MAY be originated by any assigned router in an IS-IS domain or an OSPF domain. As [I-D.ietf-ospf-segment-routing-extensions] defines OSPF extension for the purpose of the advertisements of various SID values, new Opaque LSAs [RFC5250] are defined in [I-D.ietf-ospf-prefix-link-attr]. But the SID Allocation TLV is no need to binding with the prefix of the router or re-advertised by the router. The SID Allocation TLV may be advertised in a new Opaque LSA.

5.1. The ISIS SID-allocation TLV

The SID Allocation TLV has Type TBD, and has the following format:

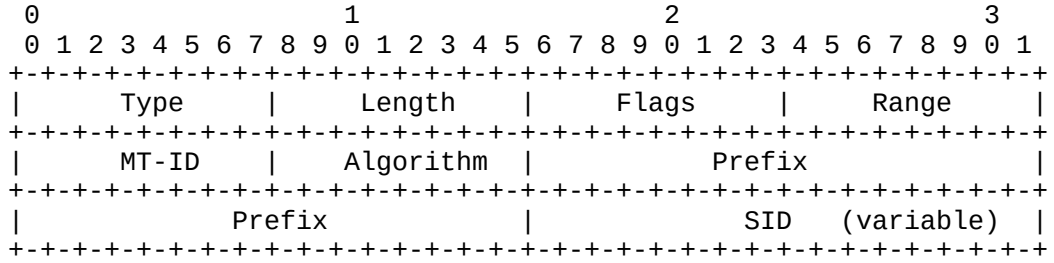


Figure 2 ISIS SID-allocation TLV

- Type: TBD.
- 1 octet of Length: variable, in bytes.
- 1 octet of flags.
- 1 octets of Range: The 'Range' field provides the ability to specify a range of addresses and their allocated SIDs. It is essentially a compression scheme to allocate a continuous Prefix and their continuous, corresponding SID/Label Block. If a single SID is advertised then the range field MUST be set to one. For range advertisements > 1, the number of addresses that need to be

mapped into a Prefix-SID and the starting value of the Prefix-SID range.

- 1 octet of MT-ID: Multi-Topology ID (as defined in [RFC4915]).
- 1 octet of Algorithm: one octet identifying the algorithm type the Prefix-SID is associated. The following value is defined by this document:
  - 0: IGP metric based Shortest Path Tree (SPT).
- 4 octet of Prefix.
- 4 octet of SID: according to the flags, it contains:
  - A 32 bit index defining the offset in the SID/Label space advertised by this router using the encodings defined in Section 3.1.
  - A 24 bit label where the 20 rightmost bits are used for encoding the label value.

5.2. The OSPF SID-Allocation TLV

The SID Allocation TLV has Type TBD, and has the following format:

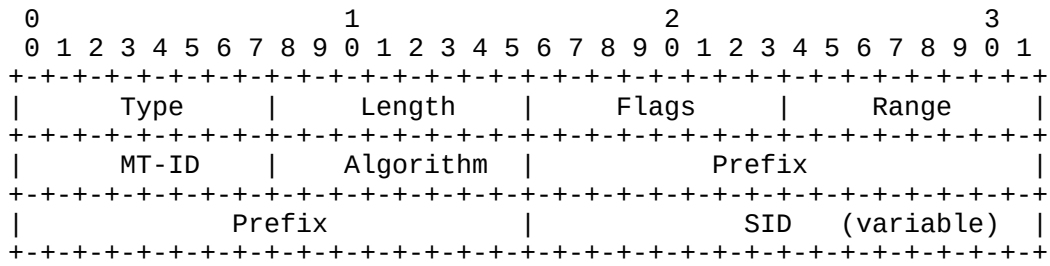


Figure 3 OSPF SID-allocation TLV

- Type: TBD.
- 1 octet of Length: variable, in bytes.
- 1 octet of flags.
- 1 octets of Range: The ’Range’ field provides the ability to specify a range of addresses and their allocated SIDs. It is essentially a compression scheme to allocate a continuous Prefix

and their continuous, corresponding SID/Label Block. If a single SID is advertised then the range field MUST be set to one. For range advertisements > 1, the number of addresses that need to be mapped into a Prefix-SID and the starting value of the Prefix-SID range.

- 1 octet of MT-ID: Multi-Topology ID (as defined in [RFC4915])
- 1 octet of Algorithm: one octet identifying the algorithm type the Prefix-SID is associated. The following value is defined by this document:
  - 0: IGP metric based Shortest Path Tree (SPT).
- 4 octet of Prefix.
- 4 octet of SID: according to the flags, it contains:
  - A 32 bit index defining the offset in the SID/Label space advertised by this router using the encodings defined in Section 3.1.
  - A 24 bit label where the 20 rightmost bits are used for encoding the label value.

6. SID Allocation Ability extension

With the compatibility consideration, nodes in the SR domain need to advertise its SR data-plane capability by using SR-Capabilities TLV in OSPF area or SR-Capabilities sub-TLV in ISIS area. So the assigned router advertises its SID allocation capability, it may be included in SR-Capabilities Sub-TLV of ISIS or in SR-Capabilities TLV of OSPF, with the Special Flag to indicate it is an assigned router.

6.1. The ISIS SR Capabilities Sub-TLV extension

The ISIS SR Capabilities Sub-TLV (Type: TBD) is optional, MAY appear multiple times inside the Router Capability TLV and has following format:

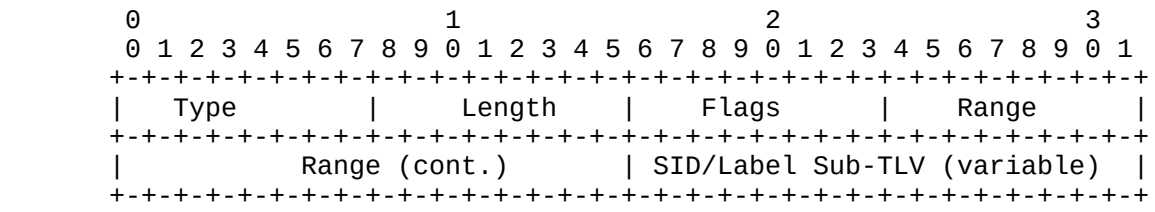
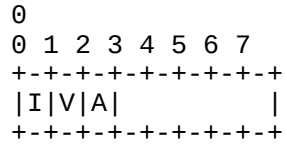


Figure 4 ISIS SR Capabilities Sub-TLV

- Type: TBD.
- 1 octet of Length: variable, in bytes.
- 1 octet of flags, the following are defined:



where:

- I-Flag: IPv4 flag. If set, then the router is capable of outgoing IPv4 encapsulation on all interfaces.
- V-Flag: IPv6 flag. If set, then the router is capable of outgoing IPv6 encapsulation on all interfaces.
- A-Flag: Allocation flag. If set, then the router is capable of allocating SID capability.

3 octets of Range: defining the number of values of the range from the starting value defined in the SID/Label Sub-TLV.

SID/Label Sub-TLV: SID/Label value as defined in [I-D.ietf-isis-segment-routing-extensions].

6.2. The OSPF SR Capabilities TLV extension

As described in [I-D.ietf-ospf-segment-routing-extensions], the SID/Label Range TLV as an additional router capability of SR, it is a top-level TLV of the Router Information Opaque LSA (defined in [RFC4970]).

The SID/Label Range TLV MAY appear multiple times and has the following format:

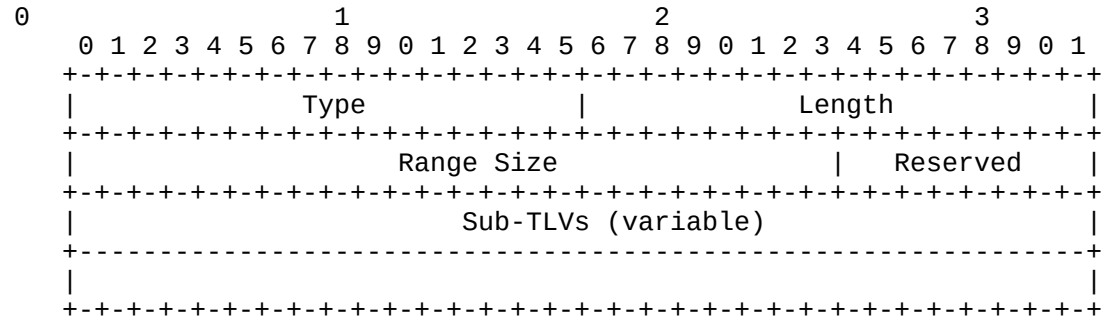
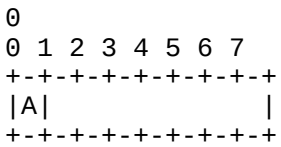


Figure 5 OSPF SR Capabilities TLV

- Type: TBD.
- 1 octet of Length: variable, in bytes.
- Range Size: 3 octets of the SID/label range.
- Reserved: 1 octets, where the following extension are defined:



where:

- A-Flag: Allocation flag. If set, then the router is capable of allocating SID capability.

Sub-TLVs: Initially, the only supported Sub-TLV is the SID/Label TLV as defined in [I-D.ietf-ospf-segment-routing-extensions].

7. Security Considerations

TBD.

8. Acknowledgements

In progress.

9. IANA Considerations

TBD.

10. Normative References

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-05 (work in progress), June 2015.

[I-D.ietf-ospf-prefix-link-attr]
Psenak, P., Gredler, H., Shakir, R., Henderickx, W., Tantsura, J., and A. Lindem, "OSPFv2 Prefix/Link Attribute Advertisement", draft-ietf-ospf-prefix-link-attr-06 (work in progress), June 2015.

[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-05 (work in progress), June 2015.

[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-03 (work in progress), May 2015.

[I-D.kh-spring-ip-ran-use-case]
Khasnabish, B., hu, f., and L. Contreras, "Segment Routing in IP RAN use case", draft-kh-spring-ip-ran-use-case-02 (work in progress), November 2014.

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

[RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, June 2007.

[RFC4970] Lindem, A., Shen, N., Vasseur, JP., Aggarwal, R., and S. Shaffer, "Extensions to OSPF for Advertising Optional Router Capabilities", RFC 4970, July 2007.

[RFC5250] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The OSPF Opaque LSA Option", RFC 5250, July 2008.

Authors' Addresses

Ting Liao
ZTE Corporation
No.50 Software Avenue
Nanjing, Jiangsu 210012
China

Phone: +86 25 88018801
Email: liao.ting@zte.com.cn

Bo Wu
ZTE Corporation
No.50 Software Avenue
Nanjing, Jiangsu 210012
China

Phone: +86 25 88018801
Email: bo.wu@zte.com.cn

Fangwei Hu
ZTE Corporation
No.889 Bibo Rd
Shanghai 201203
China

Phone: +86 21 68896273
Email: hu.fangwei@zte.com.cn

Bhumip Khasnabish
ZTE TX Inc.
55 Madison Avenue, Suite 160
Morristown, New Jersey 07960
USA

Phone: +001-781-752-8003
Email: bhumip.khasnabish@ztetx.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

P. Sarkar, Ed.
H. Gredler
Juniper Networks, Inc.

July 6, 2015

Anycast Segments in MPLS based SPRING
draft-psarkar-spring-mpls-anycast-segments-00

Abstract

Instead of forwarding to a specific device or to all devices in a group, anycast addresses, let network devices forward a packet to (or steer it through) one or more topologically nearest devices in a specific group of network devices. [I-D.ietf-spring-segment-routing] extended the use of anycast addresses to a SPRING network, wherein a group of SPRING-capable devices can represent a anycast address, by having the same SRGB label block provisioned on all the devices and each one of them advertising the same anycast prefix segment (or Anycast SID).

This document describes a proposal for implementing anycast prefix segments in SPRING, without the need to have the same SRGB block (label ranges) provisioned across all the member devices in the group. Each node can be provisioned with a separate SRGB from the label range supported by the specfic hardware platform.

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 7, 2016.

Table of Contents

1. Introduction

Anycast is a network addressing scheme and routing methodology in which packets from a single source device are forwarded to the topologically nearest node in a group of potential receiving devices, all identified by the same anycast address. There are various useful usecases of anycast addresses, and discussion of the same are outside the scope of this document.

[I-D.ietf-spring-segment-routing] extended the use of anycast addresses to SPRING networks. An operator may combine a group of SPRING-enabled nodes to form a anycast group, by picking a anycast

address and a segment identifier (hereon referred to as SID) to represent the group, and then provisioning all the nodes with the same address and SID. Once provisioned, each device in the group advertises the corresponding anycast address in it's IGP link-state advertisements along with the SID provisioned. Source devices on receiving such anycast prefix segment advertisements, finds out the topologically nearest device that originated the anycast segment and forwards packets destined to the same on the shortest-path to the nearest device.

[I-D.ietf-spring-segment-routing] also requires all devices in a given anycast group to implement the exact same SRGB block. While this requirement will always be met in SPRING network deployed over IPV6 forwarding plane [I-D.previdi-6man-segment-routing-header], the same may not be easily met in all SPRING deployments over MPLS dataplane [I-D.ietf-spring-segment-routing-mpls].

In MPLS-based SPRING deployments the segments on a given source router are actually mapped to a MPLS labels allocated from the local label pool carved out by the device for accomodating the SRGB block. In multi-vendor deployments with various types of devices deployed in the same network topology, such a anycast group may contain a good combination of devices from different vendors and have different internal hardware capabilities. In such environments it is not sufficient to assume that all the devices in a anycast group will be able to allocate exactly the same range of labels for implementing the SRGB. In reality, getting a common range of labels among all the various vendors is not feasible.

This documents provides mechanisms to implement a anycast segments with any kind of device in a multi-vendor netwrok deployment without requiring to provision the same exact range of labels for SRGB on all the devices.

2. Problem Statement

To better illustrate the problem let us consider an example topology using anycast segments as shown in Figure 1 below.

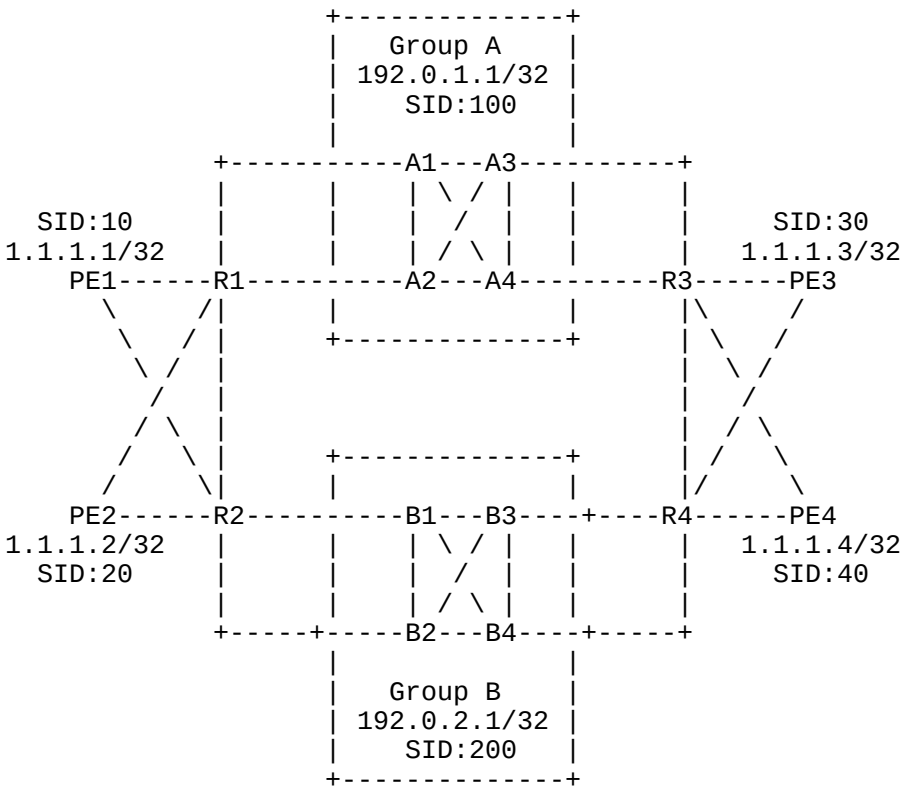


Figure 1: Topology 1

In Figure 1 above, there are two groups of transit devices. Group A consists of devices {A1, A2, A3 and A4}. They are all provisioned with the anycast address 192.0.1.1/32 and the anycast SID 100. Similarly, group B consists of devices {B1, B2, B3 and B4} and are all provisioned with the anycast address 192.0.1.2/32, anycast SID 200. In the above network topology, each PE device is connected to two routers in each of the groups A and B.

Following are all the possible ECMP paths between the various pairs of PE devices.

- P1: via {R1, A1, A3, R3}
- P2: via {R1, A1, A4, R3}
- P3: via {R1, A2, A3, R3}

- P4: via {R1, A2, A4, R3}
- P5: via {R2, B1, B3, R4}
- P6: via {R2, B1, B4, R4}
- P7: via {R2, B2, B3, R4}
- P8: via {R2, B2, B4, R4}

As seen above, there is always eight ECMP paths between each of pair of PE devices. The network operator may not wish to utilize all possible ECMP paths for all possible types of traffic flowing between a given pair of PE devices. It may be more useful for use paths P1, P2, P3 and P4 for certain types of traffic and use paths P5, P6, P7 and P8 for all other types of traffic between the same PE devices. If so desired, operators may use these anycast groups A and B and the corresponding anycast segment to impose a segment-list to forward the respective traffic flows over the desired specific paths as shown below. Figure 2 below depicts a expanded view of the paths via group A. The range labels allocated for SRGB on each of the devices in group A are also mentioned in this diagram.

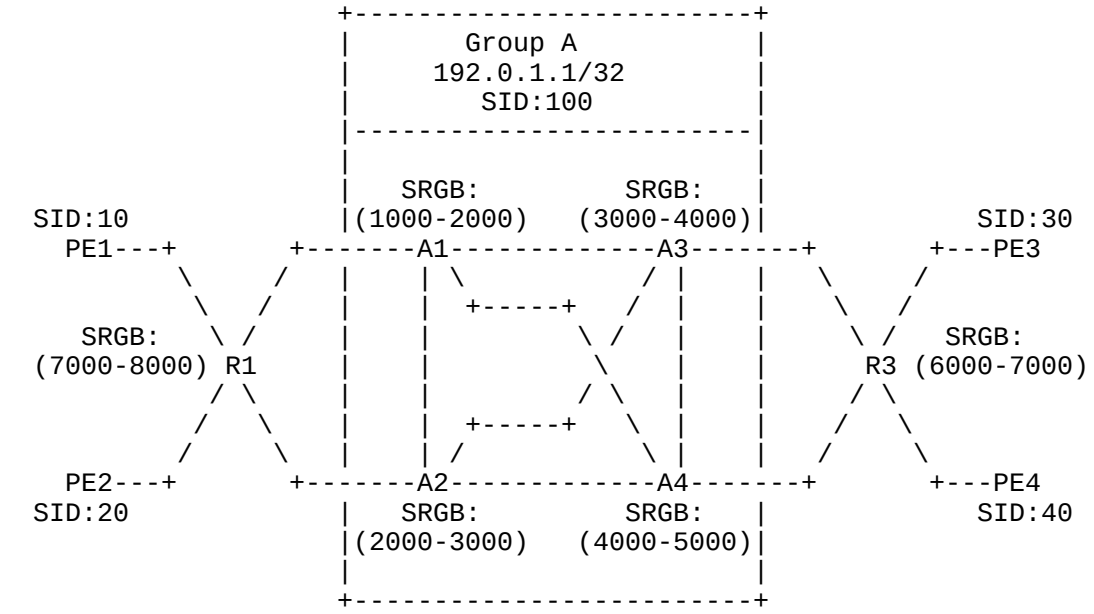


Figure 2: Transit paths via anycast group A

In the above topology, if device PE1 (or PE2) requires to send a packet to the device PE3 (or PE4) it needs to encapsulate the packet in a MPLS payload with the following stack of labels.

- Label allocated R1 for anycast SID 100 (outer label)
- Label allocated by the nearest router in group A for SID 30 (for destination PE3)

While the first label is easy to compute, in this case since there are more than one topologically nearest devices (A1 and A2), unless A1 and A2 implement same exact SRGB, determining the second label is impossible. In all likeness, devices A1 and A2 may be devices from different hardware vendors and it may not implement the same exact SRGB label ranges. In such cases, separate labels are allocated by A1 and A2 (1030 and 2030 respectively, in the above example). Hence, PE1 (or PE2) cannot compute an appropriate label stack to steer the packet exclusively through the group A devices. Same holds true for devices PE3 and PE4 when trying to send a packet to PE1 or PE2.

3. Solution

3.1. Anycast Segment Label

This document introduces the term 'Anycast Segment Label' to define the label allocated by a device to advertise reachability for the specific anycast prefix segment. The value of this label is derived by applying the SID index associated with the anycast prefix segment as an offset to the SRGB of the specific device. Table 1 below shows the labels allocated by the various devices in Figure 2 for the anycast prefix segment with SID 100.

Anycast-SID	Device	SRGB	Anycast-Segment-Label
100	R1	7000-8000	7100
100	A1	1000-2000	1100
100	A2	2000-3000	2100
100	A3	3000-4000	3100
100	A4	4000-5000	4100
100	R3	6000-7000	6100

Table 1: Anycast Segment Label Allocation

3.2. Virtual SID Label Lookup Table

When a MPLS packet on the wire first hits a device, the forwarding hardware reads the topmost label in the MPSL header and looks up the default label lookup table associated with the interface on which the label has been received. This table is generally called LFIB. The range of labels found in the LFIB constitutes the default label space.

This document introduces a separate virtual label lookup table (hereafter referred to as Virtual LFIB or V-LFIB), that represents a label space which is also separate from the actual label space represented by the default LFIB. The label value may be present in both the default and Virtual LFIB. However the forwarding semantics associated with the label under the default and Virtual LFIB may not be same. Following are the fields of a typical entry of this table.

- SID-Index: The SID index assoicated with a prefix segment originated by another device in the same network. This is also the key field for this table.

- Forwarding Semantics: This is once again one or more tuples of following items.

  * Outgoing-Label: The label(s) allocated by the neighbor device(s) on the shortest-path to the topologically nearest originator(s) of the prefix segment.

  * Outgoing-link: The link(s) connecting the device to the neighbor device(s) on the shortest path to the topologically nearest originator(s) of the prefix segment.

This document proposes that, any device, when provisioned with one or more anycast prefix segment (address and SID), it MUST create a Virtual LFIB table. Such a device MUST add an entry in the Virtual LFIB for each unicast and anycast prefix segments learnt from a remote device, if and only if the same prefix has not been provisioned on the device. The device SHOULD NOT add a entry for any of the Anycast or Node prefix segments that it has advertised itself. However if the device has learnt any anycast prefix segment from a remote device, and the same is not provisioned on this device, the device MUST include the same in the Virtual LFIB table.

In cases where a prefix segment is reachable via multiple shortest paths on a given device, the corresponding entry for the prefix SID MUST have as many forwarding entries in the Virtual LFIB table as the number of shortest-paths found for the corresponding prefix on the device. .

Figure 3 below shows how the Virtual LFIB table on each of devices in group A should look like. Please note that some of the prefix segments has multiple forwarding semantics associated with them. For example, on device A1, the prefix SID 10 (originated by PE3) is reachable through its neighbors A3 and A4. And as per the SRGB advertised by A3 and A4, the labels allocated by A3 and A4 are 3030 and 4030 respectively. Hence A1 has added two forwarding entries for the prefix SID 30 in its Virtual LFIB table.

Also please note that none of the devices in the anycast group have included the anycast SID 100 in the Virtual LFIB table, since the same has already been provisioned on these devices.

When a device receives a MPLS packet with the anycast segment label associated with one of the anycast prefix segments provisioned on the same device, the device MUST use the Virtual LFIB table to lookup the next label that follows the anycast segment label in the stack of labels found in the MPLS header. Refer to Section 3.5 for more details.

Following forwarding instructions MUST be installed in the MPLS data-plane for each entry in the Virtual LFIB entry.

- If the label at the top of the stack matches any of the prefix SIDs in the Virtual LFIB table,
  - If there are multiple forwarding tuples associated with matching table entry,
    - Select one forwarding tuple. (Criteria to select one is outside the scope of this document.)
  - Else,
    - Select the single forwarding tuple available.
  - Replace the Prefix SID index found at top of the MPLS label stack in the packet received, with the ’Outgoing-label’ from the selected forwarding tuple.
  - Forward the modified packet onto the ’Outgoing-link’ as specified in the selected forwarding tuple.
  - Ensure the next label lookup is launched on the default LFIB table.

Device	Prefix SID	Forwarding Semantics	
		Outgoing-Label	Outgoing-Link
A1	10	7010	A1->R1
	20	7020	A1->R1
	30	3030	A1->A3
		4030	A1->A4
	40	3040	A1->A3
		4040	A1->A4
A2	10	7010	A2->R1
	20	7020	A2->R1
	30	3030	A2->A3
		4030	A2->A4
	40	3040	A2->A3
		4040	A2->A4
A3	10	1010	A3->A1
		2010	A3->A2
	20	1020	A3->A1
		2020	A3->A2
	30	6030	A3->R3
	40	6040	A3->R3
A4	10	1010	A4->A1
		2010	A4->A2
	20	1020	A4->A1
		2020	A4->A2
	30	6030	A4->R3
	40	6040	A4->R3

Figure 3: Virtual LFIB Table Setup

3.3. Label Stack Computation

Any MPLS device that tries to encapsulate any kind of traffic into a SPRING-based MPLS payload (hereafter referred to as the ingress device) and steer it through a series of SPRING adjacency and/or unicast/anycast prefix segments, needs to compute an appropriate stack of MPLS labels and put it in the outgoing packet. Alternatively, in a SDN environment, the SDN controller may need to compute the label stack and install it on the ingress device.

However in both cases, as illustrated in Section 2, for a given ingress device (e.g. PE1 or PE2), there maybe multiple topolgically nearest devices in a specific anycast group (e.g. A1 and A2), even through there is only out-going link from the source device(e.g. PE1->R1 or PE2-R1). In such case, when the ingress device (or the SDN controller) wants to steer a packet through the anycast group A, it can use the anycast segment label advertised by the downstream neighbor of the ingress device for the specific anycast prefix segment. Since the packet may reach any one of the multiple devices in the group and each of them may have a separate SRGB label range, choosing the MPLS label for the next segment providing reachability to the final destination. Also, since the packet steered through a anycast segment can reach of any of the member device in the anycast group, it is sufficient to assume that the ingress (or the controller) cannot place an adjacency segment immediately after a anycast segment in the outgoing packet.

This document proposes the ingress device (or the SDN controller) to directly use the SID as the label for a prefix segment (can be another anycast)that immediately follows a given anycast segment already encoded into the label stack of the outgoing MPLS packet. The ingress (or the controller) MUST follow the algorithm below to compute the label-stack it must use to steer a packet through a list of SPRING segments.

- Set 'last_segment' ==> NONE.
- For [all 'segments' in Segment_List]
  - If {'segment'.type == Adjacency_Segment}
    - Set 'label' ==> 'segment'.Adjacency_Segment_Label.
  - Else
    - If {'last_segment'.type == Anycast_Prefix_Segment}
      - Set 'label' ==> 'segment'.SID_index.

+ Else

- Set 'label' ==> 'Prefix_Segment_Label'.

o Add 'label' to 'label_stack'.

3.4. Advertising Anycast Prefix Segments

Like unicast prefix segments, anycast prefix segments SHOULD be advertised in IGP Link-state advertsements using IGP protocol extension for SPRING specified in [I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions] and [I-D.ietf-ospf-ospfv3-segment-routing-extensions]. This document does not propose any protocol extension for advertising anycast prefix segments.

However when advertising the anycast segments, the originating device MUST set the corresponding P-Flag(No-PHP) in ISIS Prefix-SID SubTLV and/or the NP-Flag (No-PHP) in OSPFv2 and OSPFv3 Prefix-SID SubTLV to 1 and the E-Flag in the same SubTLVs to 0. Please refer to following for more details on usage of these flags.

o ISIS Prefix-SID SubTLV [I-D.ietf-isis-segment-routing-extensions]

o OSPFv2 Prefix-SID SubTLV [I-D.ietf-ospf-segment-routing-extensions]

o OSPFv3 Prefix-SID SubTLV [I-D.ietf-ospf-ospfv3-segment-routing-extensions]

The proposal above, ensures that a MPLS packet sent to (or taking transit through) a given anycast group, always arrives at the topologically nearest device in the group, with a label that is derived from the device's SRGB, and the SID associated with the corresponding anycast prefix segment.

In Figure 2, when PE1 or PE2 intends to steer a packet destined for PE3 or PE4, through the anycast group A (SID 100), it needs to forward the packet to R1 (SRGB:7000-8000), after putting the label 7100 (derived from R1's SRGB), at top of the label stack in the MPLS header. However when the same packet is forwarded to A1 or A2 (topologicaly nearest devices in group A), R1 shall not POP (or remove) the label 7100. Instead R1 shall replace it with the label 1100 (while forwarding to A1) or 2100 (while forwarding to A2).

3.5. Programming Anycast Prefix Segments

The proposal specified in Section 3.4, ensures that a MPLS packet destined to (or steered via) a anycast prefix segment always arrives at the nearest device in the anycast group with a label derived from the device's SRGB and the SID associated with the corresponding anycast prefix segment, as the top-most label label stack in its MPLS header. If this label is also the bottom-most label (S=1), it means packet has been destined to the anycast segment, and should be consumed by the local device. If the label is not the bottom-most label (S=0), the packet must be forwarded to the next segment, for which the next label in the stack should be consulted. However Section 3.3 specifies that the next label in such case, shall be directly the SID associated with the next segment. Since the SID associated with a prefix segment may directly collide with another label in the default LFIB table, Section 3.2 also proposed to have a Virtual LFIB table to provide a separate label-space for looking up the next label.

This document specifies that a device provisioned with a given prefix segment index MUST implement following forwarding semantics for the anycast segment label (refer to Section 3.1) associated with the anycast prefix segment.

- If the label at the top the stack is a anycast segment label,
  - Pop the label.
  - If bottom-most label in the stack (S=1),
    - Send it to host stack for local consumption, as usual.
  - Else if not the bottom-most label in the stack (S=0),
    - Set the Virtual LFIB table as the lookup table for the next label lookup.
    - Launch a lookup for the next label in the stack.
- Else
  - Lookup the label in the default LFIB table as usual.

3.6. Packet Flow

Figure 4 below ilustrate how SPRING-based MPLS packets destined for PE3 and sourced by PE1 are expected to flow theough when PE1

encapsulates the packet with an appropriate label stack to steer it through group A devices only

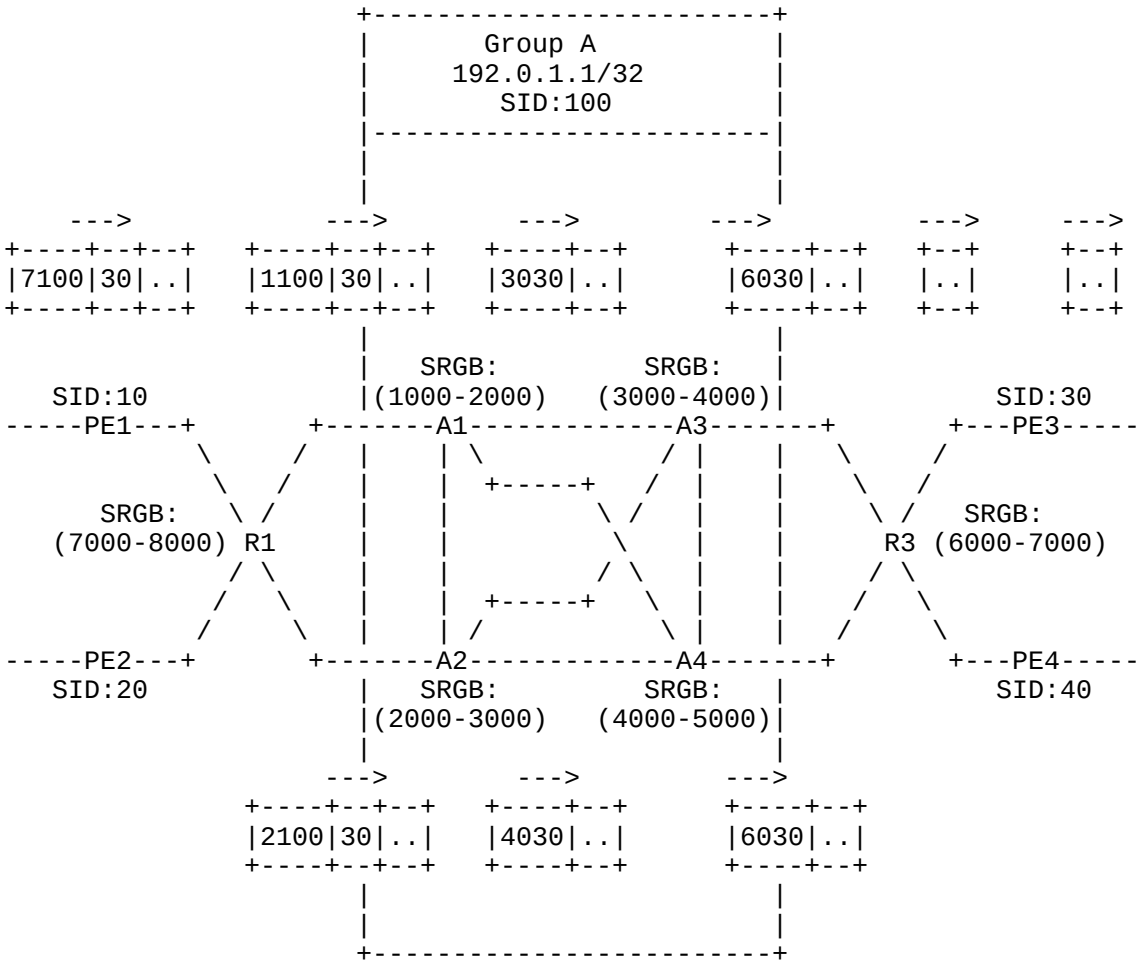


Figure 4: Packet Flow through MPLS-based SPRING Anycast Segments

4. Acknowledgements

Many many thanks to Shraddha Hegde for her valuable inputs.

5. IANA Considerations

N/A. - No protocol changes are proposed in this document.

6. Security Considerations

This document does not introduce any change in any of the protocol specifications. It simply proposes additional inequalities for selecting LFAs for multi-homed prefixes.

7. References

7.1. Normative References

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

7.2. Informative References

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-04 (work in progress), May 2015.

[I-D.ietf-ospf-ospfv3-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPFv3 Extensions for Segment Routing", draft-ietf-ospf-ospfv3-segment-routing-extensions-02 (work in progress), February 2015.

[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-04 (work in progress), February 2015.

[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-03 (work in progress), May 2015.

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., Shakir, R., Tantsura, J., and E. Crabbe, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-01 (work in progress), May 2015.

[I-D.previdi-6man-segment-routing-header]
Previdi, S., Filsfils, C., Field, B., and I. Leung, "IPv6 Segment Routing Header (SRH)", draft-previdi-6man-segment-routing-header-06 (work in progress), May 2015.

Authors' Addresses

Pushpasis Sarkar (editor)
Juniper Networks, Inc.
Electra, Exora Business Park
Bangalore, KA 560103
India

Email: psarkar@juniper.net

Hannes Gredler
Juniper Networks, Inc.
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US

Email: hannes@juniper.net