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].

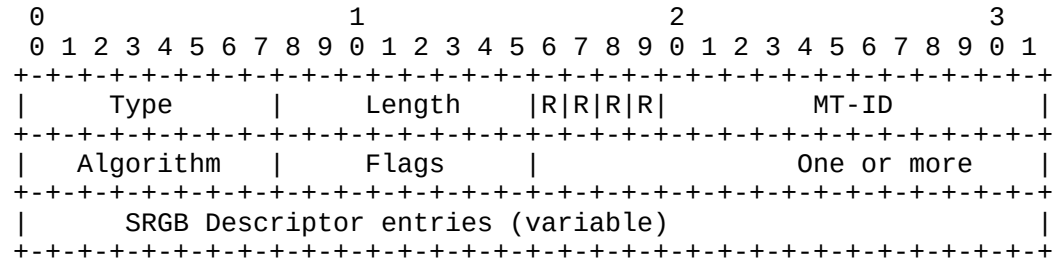

Figure 8: Topology-Algorithm-Label-Block sub-TLV

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@