Spring WG
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2015

Fangwei. Hu
Ran. Chen
Frank. Feng
ZTE Corporation
March 5, 2015

YANG Data model for Segment Routing
draft-hu-spring-yang-00.txt

Abstract

This document defines a YANG data model for segment routing technology. The data model covers configuration data and event notifications.

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

YANG[RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241].

This document defines YANG [RFC6020] data model for the management of SPRING(Source Packet Routing in Networking) [I-D.ietf-spring-segment-routing]. The YANG data design, YANG data model and YANG notification event are introduced in this proposal.

2. Design of Data model

This document proposes a base segment routing YANG data model. The module can be augmented for other segment routing extended features with their specific definitions, such as segment routing OAM, IPv6 segment routing, etc. In addition, a notification event for the segment routing is defined.

The figure below describe the overall structure of the Segment Routing Yang model:

```
module: Spring
 +--rw global-block
 |   +--rw Min
 |   +--rw Max
 +--rw segment
 |   +--rw name
 |   +--rw sid
 |   +--rw type
 |   +--rw scope
 |   +--rw prefix-flag
 |      +--rw R
 |      +--rw N
 |      +--rw P
 |      +--rw E
```

```
    |      +--rw V
    |      +--rw L
    |   +--rw adjacency-flag
    |      +--rw F
    |      +--rw B
    |      +--rw V
    |      +--rw L
    |      +--rw S
    +--rw fec-mapping
    |   +--fec-prefix
    |   +--explicit-route
    |   +--sid-label-binding-flag
    |      +--rw F
    |      +--rw M
    |      +--rw S
    |      +--rw D
    |      +--rw A
    +--rw label-entity
    |   +--rw sid
    |   +--rw incoming
    |   +--rw outgoing
    |   +--rw next-hop
    |   +--rw header-operation
    |   +--rw egress-interface
    +--rw tunnel
    |   +--rw name
    |   +--rw ingress
    |   +--rw egress
    |   +--rw priority
    |   +--rw explicit-route
    |   +--rw path-type
    |   +--rw frr-protection-method
    +

  notifications:
    +--n tunnel-event
    |  +--ro name
    |  +--ro tunnel-creation
    |  +--ro tunnel-deletion
    |  +--ro tunnel-state
    |  +--ro frr-status-change
    |  +--ro path-protection-status-change
    +
```

3. Spring YANG Data model

```
 module ietf-spring {

    namespace "urn:ietf:params:xml:ns:yang:ietf-spring";

    prefix "Spring";

    organization
        "IETF SPRING (Source Packet Routing in Networking) Working Group";

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

         WG Chair: Bruno Decraene
                   <mailto:bruno.decraene@orange.com>

         WG Chair: John Scudder
                   <mailto:jgs@juniper.net>
         Editor:   Fangwei Hu
                   <mailto:hu.fangwei@zte.com.cn>

         Editor:   Ran Chen
                   <mailto:chen.ran@zte.com.cn>

         Editor:   Frank Feng
                   <mailto:feng.chong33@zte.com.cn>";

    description
      "The YANG module defines data model for the management of Spring";

    revision 2015-03-05 {
      description
        "Initial revision";
    }

    //typedefs
    typedef sid {
        description
          "The type of segment index.";
        type uint32;
    }

    typedef tunnel-state {
     type enumeration {
      enum up;
      enum down;
```

```
     }
    }

    typedef frr-status {
     type enumeration {
      enum master;
      enum slave;
     }
    }

    typedef path-protection-status {
     type enumeration {
      enum node;
      enum link;
      }
    }

    //data defs
    container global-block {
     description
       "SID global block, the set of local
        labels reserved for global segments";

    leaf min {
     type uint32;
     description
        "The begin of the SID global block";
     }

    leaf max {
     type uint32;
     description
        "The end of the SID global block";
     }
    }

    list segment {
     key name;
     unique sid;
     leaf name {
     description
       "identify of a segment.";
     type string;
     }

     leaf sid {
     description
       "index of a segment.";
```

```
     config false;
     type sid;
    }

    leaf type {
     description
        "The type of a segment. A segment could be a prefix, node,
         adjacency, and anycast segment or other types defined in the future.";
     mandatory true;

    type enumeration {
      enum prefix;
      enum node;
      enum adjacency;
      enum anycast;
     }
    }

    leaf scope {
      description
        "The scope of segment";
      mandatory true;
      type enumeration {
      enum global;
      enum local;
     }
    }

    leaf prefix-flag {
      description
        "The flag of Prefix Segment ";
      when "type=prefix";
      type bits {
      bit R {
        description
          "Re-advertisement flag, If set, then the prefix to which this
           Prefix-SID is attached, has been propagated by the router
           either from another level (i.e.: from level-1 to level-2 or
           the opposite) or from redistribution (e.g.: from another
           protocol)";
        position 1;
        }

      bit N {
        description
          "Node-SID flag, If set, then the Prefix-SID refers to the router
           identified by the prefix";
        position 2;
```

```
        }

      bit P {
        description
          "no-PHP flag, If set, then the penultimate hop MUST NOT pop the
           Prefix-SID before delivering the packet to the node that advertised
           the Prefix-SID";
        position 3;
        }

      bit E {
        description
          "Explicit-Null Flag, If set, any upstream neighbor of the Prefix-SID
          originator MUST replace the Prefix-SID with a Prefix-SID having an
          Explicit-NULL value (0 for IPv4 and 2 for IPv6) before forwarding
          the packet";
        position 4;
        }

      bit V {
        description
          "Value flag, If set, then the Prefix-SID carries a value (instead of
           an index).";
        position 5;
        }

      bit L {
        description
          "Local Flag, If set, then the value/index carried by the Prefix-SID
           has local significance";
        position 6;
        }
      }
    }

    leaf adjacency-flag {
     description
        "The flag of Adjacency Segment";
     when "type=adjacency";

     type bits {
      bit F {
        description
          "Address-Family flag. If unset, then the Adj-SID refers to an
           adjacency with outgoing IPv4 encapsulation.  If set then the
           Adj-SID refers to an adjacency with outgoing IPv6 encapsulation";
        position 1;
        }
```

```
      bit B {
        description
          "Backup flag. If set, the Adj-SID refers to an adjacency being
           protected.";
        position 2;
        }

      bit V {
        description
          "If set, then the Adj-SID carries a value.";
        position 3;
       }

      bit L {
        description
          "If set, then the value/index carried by the Adj-SID has local
           significance.";
        position 4;
       }

      bit S {
        description
          "Set Flag. When set, the S-Flag indicates that the Adj-SID refers
           to a set of adjacencies.";
        position 5;
       }
     }
    }


   }

   list fec-mapping {
     key fec-prefix;
     leaf fec-prefix {
     description
       "The prefix address of FEC of the router.";
     type uint32;
     }

     leaf-list explicit-route {
       type sid;
     }

     leaf sid-label-binding-flag {
       description
         "The flag of SID/Label Binding ";
       type bits {
```

```
          bit F {
            description
              "Address Family flag. If unset, then the Prefix FEC carries
               an IPv4 Prefix. If set, then the Prefix FEC carries an IPv6 Prefix
";
            position 1;
           }

          bit M {
            description
              "Mirror Context flag. Set if the advertised SID/path corresponds
               to a mirrored context.";
            position 2;
           }

          bit S {
           description
              "If set, the SID/Label Binding TLV SHOULD be flooded across the
               entire routing domain. If the S flag is not set, the SID/Label
               Binding TLV MUST NOT be leaked between levels ";
           position 3;
          }

          bit D {
           description
              "when the SID/Label Binding TLV is leaked from level-2 to level-1,
               the D bit MUST be set. Otherwise, this bit MUST be clear.SID/Label
               Binding TLVs with the D bit set MUST NOT be leaked from level-1 to
               level-2 ";
           position 4;
          }

          bit A {
           description
              "The originator of the SID/Label Binding TLV MAY set the A bit in
               order to signal that the prefixes and SIDs advertised in the
               SID/Label Binding TLV are directly connected to their originators.
";
               position 5;
          }
        }
      }
    }

   list label-entity {
     key sid;
     config false;
     leaf sid {
      description
        "The index of a segment.";
```

```
      type sid;
     }

     leaf incoming {
       description
         "Incoming active segment.";
       type sid;
      }

     leaf outgoing {
       description
         "Outgoing active segment.";
       type sid;
      }

     leaf next-hop {
       description
         "The IP address of the next hop for the continue operation.";
       type uint32;
      }

     leaf header-operation {
       description
         "The operation action for the flow.";
       type enumeration {
         enum push;
         enum next;
         enum continue;
        }
     }

     leaf egress-interface {
       description
         "The outgoing interface for the flow.";
       type uint32;
      }
   }


   list tunnel {
     description
       "The spring tunnel";
     key name;
     leaf name {
     type string;
    }

     leaf ingress {
```

```
      description
        "Ingress SID";
      type sid;
    }

    leaf egress {
      description
        "Egress SID";
      type sid;
    }

    leaf priority {
      description
        "The priority of tunnel.";
      type uint32;
    }

    leaf-list explicit-route {
       type sid;
    }

    leaf path-type {
      description
        "Whether the tunnel is a master or slave path?";
      type frr-status;
    }

    leaf frr-protection-method {
      description
        "whether the protection object is a node or link?";
      type path-protection-status;
    }
  }
```

4. Spring YANG notification

```
      //notifications
      notification tunnels-event {
        list tunnel {
          key name;
          leaf name {
            description
              "The name of tunnel";
            type string;
           }

          choice event {
           mandatory true;
```

```
            case creation {
              leaf creation {
              type empty;
             }
            }
            case deletion {
             leaf deletion {
             type empty;
             }
            }

            case state-transition {
             container state-transition {
              must "from != to";
              leaf from {
                type tunnel-state;
                mandatory true;
               }

              leaf to {
                type tunnel-state;
                mandatory true;
               }
             }
            }

            case frr-status-change {
             container frr-status-change {
              must "from != to";
              leaf from {
               type frr-status;
               mandatory true;
              }

              leaf to {
               type frr-status;
               mandatory true;
              }
             }
            }

            case path-protection-status-change {
             container path-protection-status-change {
              must "from != to";
               leaf from {
                 type path-protection-status;
                 mandatory true;
               }
```

```
              leaf to {
                type path-protection-status;
                mandatory true;
              }
             }
            }
           }
         }
        }
       }
```

5. Security Considerations

TBD.

6. Acknowledgements

TBD.

7. IANA Considerations

TBD.

8. Normative References

[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.

[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.

Authors' Addresses

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

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

Ran Chen
ZTE Corporation
No.50 Software Avenue,Yuhuatai District
Nanjing, Jiangsu Province 210012
China

Phone: +86 025 88014636
Email: chen.ran@zte.com.cn

Frank Feng
ZTE Corporation
No.86 Zijinghua Rd
Nanjing, Jiangsu Province 210012
China

Phone: +86 21 68896273
Email: feng.chong33@zte.com.cn

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 6, 2015

S. Kini, Ed.
Ericsson
K. Kompella
Juniper
S. Sivabalan
Cisco
S. Litkowski
Orange
R. Shakir
B.T.
X. Xu
Huawei
W. Hendrickx
Alcatel-Lucent
J. Tantsura
Ericsson

March 5, 2015

Entropy labels for source routed stacked tunnels
draft-ietf-mpls-spring-entropy-label-00

Abstract

Source routed tunnel stacking is a technique that can be leveraged to provide a method to steer a packet through a controlled set of segments. This can be applied to the Multi Protocol Label Switching (MPLS) data plane. Entropy label (EL) is a technique used in MPLS to improve load balancing. This document examines and describes how ELs are to be applied to source routed stacked tunnels.

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.

Table of Contents

1. Introduction

The source routed stacked tunnels paradigm is leveraged by techniques such as Segment Routing (SR) [I-D.filsfils-spring-segment-routing] to steer a packet through a set of segments. This can be directly applied to the MPLS data plane, but it has implications on label stack depth.

Clarifying statements on label stack depth have been provided in [RFC7325] but they do not address the case of source routed stacked MPLS tunnels as described in [I-D.gredler-spring-mpls] or

[I-D.filsfils-spring-segment-routing] where deeper label stacks are more prevalent.

Entropy label (EL) [RFC6790] is a technique used in the MPLS data plane to provide entropy for load balancing. When using LSP hierarchies there are implications on how [RFC6790] should be applied. One such issue is addressed by [I-D.ravisingh-mpls-el-for-seamless-mpls] but that is when different levels of the hierarchy are created at different LSRs. The current document addresses the case where the hierarchy is created at a single LSR as required by source stacked tunnels.

A use-case requiring load balancing with source stacked tunnels is given in Section 3. A recommended solution is described in Section 4 keeping in consideration the limitations of implementations when applying [RFC6790] to deeper label stacks. Options that were considered to arrive at the recommended solution are documented for historical purposes in Section 5.

1.1. Requirements Language

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

Although this document is not a protocol specification, the use of this language clarifies the instructions to protocol designers producing solutions that satisfy the requirements set out in this document.

2. Abbreviations and Terminology

EL - Entropy Label

ELI - Entropy Label Identifier

ELC - Entropy Label Capability

SR - Segment Routing

ECMP - Equal Cost Multi Paths

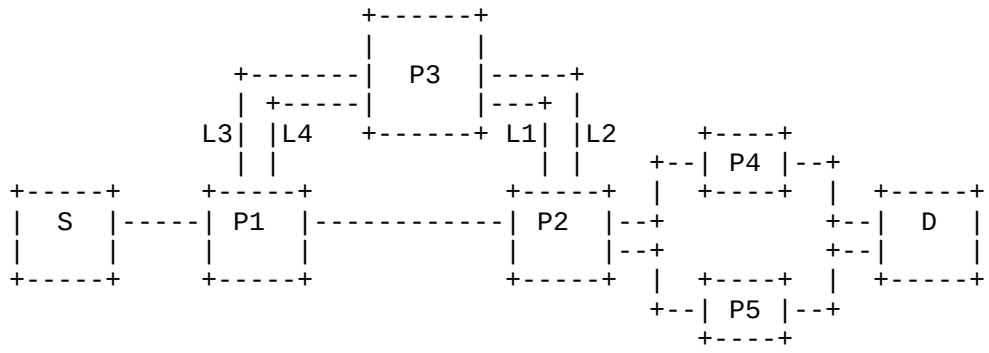
MPLS - Multiprotocol Label Switching

SID - Segment Identifier

RLD - Readable Label Depth

OAM - Operation, Administration and Maintenance

3. Use-case requiring multipath load balancing in source stacked tunnels



S=Source LSR, D=Destination LSR, P1,P2,P3,P4,P5=Transit LSRs,
L1,L2,L3,L4=Links

Figure 1: Traffic engineering use-case

Traffic-engineering (TE) is one of the applications of MPLS and is also a requirement for source stacked tunnels. Consider the topology shown in Figure 1. Lets say the LSR P1 has a limitation that it can only look four labels deep in the stack to do multipath decisions. All other transit LSRs in the figure can read deep label stacks and the LSR S can insert as many <ELI, EL> pairs as needed. The LSR S requires data to be sent to LSR D along a traffic-engineered path that goes over the link L1. Good load balancing is also required across equal cost paths (including parallel links). To engineer traffic along a path that takes link L1, the label stack that LSR S creates consists of a label to the node SID of LSR P3, stacked over the label for the adjacency SID of link L1 and that in turn is stacked over the label to the node SID of LSR D. For simplicity lets assume that all LSRs use the same label space for source stacked tunnels. Lets L_N-P denote the label to be used to reach the node SID of LSR P. Let L_A-Ln denote the label used for the adjacency SID for link Ln. The LSR S must use the label stack <L_N-P3, L_A-L1, L_N-D> for traffic-engineering. However to achieve good load balancing over the equal cost paths P2-P4-D, P2-P5-D and the parallel links L3, L4, a mechanism such as Entropy labels [RFC6790] should be adapted for source stacked tunnels. Multiple ways to apply entropy labels were considered and are documented in Section 5 along with their tradeoffs. A recommended solution is described in Section 4.

4. Recommended EL solution for SPRING

The solution described in this section follows [RFC6790].

An LSR may have a limitation in its ability to read and process the label stack in order to do multipath load balancing. This limitation expressed in terms of the number of label stack entries that the LSR can read is henceforth referred to as the Readable Label Depth (RLD) capability of that LSR. If an EL does not occur within the RLD of an LSR in the label stack of the MPLS packet that it receives, then it would lead to poor load balancing at that LSR. The RLD of an LSR is a characteristic of the forwarding plane of that LSR's implementation and determining it is outside the scope of this document.

In order for the EL to occur within the RLD of LSRs along the path corresponding to a label stack, multiple <ELI, EL> pairs MAY be inserted in the label stack as long as the tunnel's label below which they are inserted are advertised with entropy label capability enabled. The LSR that inserts <ELI, EL> pairs MAY have limitations on the number of such pairs that it can insert and also the depth at which it can insert them. If due to any limitation, the inserted ELs are at positions such that an LSR along the path receives an MPLS packet without an EL in the label stack within that LSR's RLD, then the load balancing performed by that LSR would be poor. Special attention should be paid when a forwarding adjacency LSP (FA-LSP) [RFC4206] is used as a link along the path of a source stacked LSP, since the labels of the FA-LSP would additionally count towards the depth of the label stack when calculating the appropriate positions to insert the ELs. The recommendations for inserting <ELI, EL> pairs are:

- An LSR that is limited in the number of <ELI, EL> pairs that it can insert SHOULD insert such pairs deeper in the stack.

- An LSR SHOULD try to insert <ELI, EL> pairs at positions so that for the maximum number of transit LSRs, the EL occurs within the RLD of the incoming packet to that LSR.

- An LSR SHOULD try to insert the minimum number of such pairs while trying to satisfy the above criteria.

A sample algorithm to insert ELs is shown below. Implementations can choose any algorithm as long as it follows the above recommendations.

```
Initialize the current EL insertion point to the
  bottommost label in the stack that is EL-capable
while (local-node can push more <ELI,EL> pairs OR
          insertion point is not above label stack) {
    insert an <ELI,EL> pair below current insertion point
    move new insertion point up from current insertion point until
        ((last inserted EL is below the RLD) AND (RLD > 2)
                        AND
         (new insertion point is EL-capable))
    set current insertion point to new insertion point
}
```

Figure 2: Algorithm to insert <ELI, EL> pairs in a label stack

When this algorithm is applied to the example described in Section 3 it will result in ELs being inserted in two positions, one below the label L_N-D and another below L_N-P3. Thus the resulting label stack would be <L_N-P3, ELI, EL, L_A-L1, L_N-D, ELI, EL>

The RLD can be advertised via protocols and those extensions would be described in separate documents [I-D.xu-isis-mpls-elc] and [I-D.xu-ospf-mpls-elc].

The recommendations above are not expected to bring any additional OAM considerations beyond those described in section 6 of [RFC6790]. However, the OAM requirements and solutions for source stacked tunnels are still under discussion and future revisions of this document will address those if needed.

5. Options considered

5.1. Single EL at the bottom of the stack of tunnels

In this option a single EL is used for the entire label stack. The source LSR S encodes the entropy label (EL) below the labels of all the stacked tunnels. In the example described in Section 3 it will result in the label stack at LSR S to look like <L_N-P3, L_A-L1, L_N-D, ELI, EL> <remaining packet header>. Note that the notation in [RFC6790] is used to describe the label stack. An issue with this approach is that as the label stack grows due an increase in the number of SIDs, the EL goes correspondingly deeper in the label stack. Hence transit LSRs have to access a larger number of bytes in the packet header when making forwarding decisions. In the example described in Section 3 the LSR P1 would poorly load-balance traffic on the parallel links L3, L4 since the EL is below the RLD of the packet received by P1. A load balanced network design using this approach must ensure that all intermediate LSRs have the capability

to traverse the maximum label stack depth as required for that application that uses source routed stacking.

In the case where the hardware is capable of pushing a single <ELI, EL> pair at any depth, this option is the same as the recommended solution in Section 4.

This option was discounted since there exist a number of hardware implementations which have a low maximum readable label depth. Choosing this option can lead to a loss of load-balancing using EL in a significant part of the network but that is a critical requirement in a service provider network.

5.2. An EL per tunnel in the stack

In this option each tunnel in the stack can be given its own EL. The source LSR pushes an <ELI, EL> before pushing a tunnel label when load balancing is required to direct traffic on that tunnel. In the example described in Section 3, the source LSR S encoded label stack would be <L_N-P3, ELI, EL, L_A-L1, L_N-D, ELI, EL> where all the ELs can be the same. Accessing the EL at an intermediate LSR is independent of the depth of the label stack and hence independent of the specific application that uses source stacking on that network. A drawback is that the depth of the label stack grows significantly, almost 3 times as the number of labels in the label stack. The network design should ensure that source LSRs should have the capability to push such a deep label stack. Also, the bandwidth overhead and potential MTU issues of deep label stacks should be accounted for in the network design.

In the case where the RLD is the minimum value (3) for all LSRs, all LSRs are EL capable and the LSR that is inserting <ELI, EL> pairs has no limit on how many it can insert then this option is the same as the recommended solution in Section 4.

This option was discounted due to the existence of hardware implementations that can push a limited number of labels on the label stack. Choosing this option would result in a hardware requirement to push two additional labels per tunnel label. Hence it would restrict the number of tunnels that can form a LSP and constrain the types of LSPs that can be created. This was considered unacceptable.

5.3. A re-usable EL for a stack of tunnels

In this option an LSR that terminates a tunnel re-uses the EL of the terminated tunnel for the next inner tunnel. It does this by storing the EL from the outer tunnel when that tunnel is terminated and re-inserting it below the next inner tunnel label during the label swap

operation. The LSR that stacks tunnels SHOULD insert an EL below the outermost tunnel. It SHOULD NOT insert ELs for any inner tunnels. Also, the penultimate hop LSR of a segment MUST NOT pop the ELI and EL even though they are exposed as the top labels since the terminating LSR of that segment would re-use the EL for the next segment.

In Section 3 above, the source LSR S encoded label stack would be <L_N-P3, ELI, EL, L_A-L1, L_N-D>. At P1 the outgoing label stack would be <L_N-P3, ELI, EL, L_A-L1, L_N-D> after it has load balanced to one of the links L3 or L4. At P3 the outgoing label stack would be <L_N-D, ELI, EL>. At P2 the outgoing label stack would be <L_N-D, ELI, EL> and it would load balance to one of the nexthop LSRs P4 or P5. Accessing the EL at an intermediate LSR (e.g. P1) is independent of the depth of the label stack and hence independent of the specific use-case to which the stacked tunnels are applied.

This option was discounted due to the significant change in label swap operations that would be required for existing hardware.

5.3.1. EL at top of stack

A slight variant of the re-usable EL option is to keep the EL at the top of the stack rather than below the tunnel label. In this case each LSR that is not terminating a segment should continue to keep the received EL at the top of the stack when forwarding the packet along the segment. An LSR that terminates a segment should use the EL from the terminated segment at the top of the stack when forwarding onto the next segment.

This option was discounted due to the significant change in label swap operations that would be required for existing hardware.

5.4. ELs at readable label stack depths

In this option the source LSR inserts ELs for tunnels in the label stack at depths such that each LSR along the path that must load balance is able to access at least one EL. Note that the source LSR may have to insert multiple ELs in the label stack at different depths for this to work since intermediate LSRs may have differing capabilities in accessing the depth of a label stack. The label stack depth access value of intermediate LSRs must be known to create such a label stack. How this value is determined is outside the scope of this document. This value can be advertised using a protocol such as an IGP. For the same Section 3 above, if LSR P1 needs to have the EL within a depth of 4, then the source LSR S encoded label stack would be <L_N-P3, ELI, EL, L_A-L1, L_N-D, ELI, EL> where all the ELs would typically have the same value.

In the case where the RLD has different values along the path and the LSR that is inserting <ELI, EL> pairs has no limit on how many pairs it can insert, and it knows the appropriate positions in the stack where they should be inserted, then this option is the same as the recommended solution in Section 4.

A variant of this solution was selected which balances the number of labels that need to be pushed against the requirement for entropy.

6. Acknowledgements

The authors would like to thank John Drake, Loa Andersson, Curtis Villamizar, Greg Mirsky, Markus Jork, Kamran Raza and Nobo Akiya for their review comments and suggestions.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

This document does not introduce any new security considerations beyond those already listed in [RFC6790].

9. References

9.1. Normative References

[I-D.filsfils-spring-segment-routing]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., Milojevic, I., Shakir, R., Ytti, S., Henderickx, W., Tantsura, J., and E. Crabbe, "Segment Routing Architecture", draft-filsfils-spring-segment-routing-04 (work in progress), July 2014.

[I-D.gredler-spring-mpls]
Gredler, H., Rekhter, Y., Jalil, L., Kini, S., and X. Xu, "Supporting Source/Explicitly Routed Tunnels via Stacked LSPs", draft-gredler-spring-mpls-06 (work in progress), May 2014.

[I-D.ravisingh-mpls-el-for-seamless-mpls]
Singh, R., Shen, Y., and J. Drake, "Entropy label for seamless MPLS", draft-ravisingh-mpls-el-for-seamless-mpls-04 (work in progress), October 2014.

[I-D.xu-isis-mpls-elc]
Xu, X., Kini, S., Sivabalan, S., Filsfils, C., and S. Litkowski, "Signaling Entropy Label Capability Using IS-IS", draft-xu-isis-mpls-elc-01 (work in progress), September 2014.

[I-D.xu-ospf-mpls-elc]
Xu, X., Kini, S., Sivabalan, S., Filsfils, C., and S. Litkowski, "Signaling Entropy Label Capability Using OSPF", draft-xu-ospf-mpls-elc-01 (work in progress), October 2014.

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

[RFC4206] Kompella, K. and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, October 2005.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

9.2. Informative References

[I-D.filsfils-spring-segment-routing-use-cases]
Filsfils, C., Francois, P., Previdi, S., Decraene, B., Litkowski, S., Horneffer, M., Milojevic, I., Shakir, R., Ytti, S., Henderickx, W., Tantsura, J., Kini, S., and E. Crabbe, "Segment Routing Use Cases", draft-filsfils-spring-segment-routing-use-cases-01 (work in progress), October 2014.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. 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.

[RFC7325] Villamizar, C., Kompella, K., Amante, S., Malis, A., and C. Pignataro, "MPLS Forwarding Compliance and Performance Requirements", RFC 7325, August 2014.

Authors' Addresses

Sriganesh Kini (editor)
Ericsson

Email: sriganesh.kini@ericsson.com

Kireeti Kompella
Juniper

Email: kireeti@juniper.net

Siva Sivabalan
Cisco

Email: msiva@cisco.com

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Rob Shakir
B.T.

Email: rob.shakir@bt.com

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Wim Hendrickx
Alcatel-Lucent

Email: wim.henderickx@alcatel-lucent.com

Jeff Tantsura
Ericsson

Email: jeff.tantsura@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

Z. Li
N. Wu
Huawei

March 9, 2015

Multi-Topology (MT) Segment in Segment Routing
draft-li-spring-multi-topology-segment-00

Abstract

Multi-Topologies (MTs) can be used for computing different paths for unicast traffic, multicast traffic, different classes of service based on flexible criteria, or an in-band network management topology. This document proposes the multi-topology segment for segment routing. MRT FRR using multi-topology segment is described as the use case to explains the procedures of the forwarding plane and the control plane for multi-topology segment..

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 September 10, 2015.

Copyright Notice

Table of Contents

1. Introduction

Segment Routing (SR), introduced by [I-D.ietf-spring-segment-routing], leverages the source routing paradigm. A packet can be steered through an ordered list of instructions, which are also called segments. The node segment, adjacency segment, etc. have been proposed for different usecases.

This document proposes the multi-topology segment for the segment routing. As stated, Multi-Topologies (MTs) can be used for computing different paths for unicast traffic, multicast traffic, different classes of service based on flexible criteria, or an in-band network management topology. The multi-topology segment is used to identify the multi-topology with the segment ID. MRT FRR using multi-topology segment is described as the use case to explains the procedures of the forwarding plane and the control plane for multi-topology segment.

2. Terminology

- Segment: a segment identifies an instruction.

- Multi-Topology Segment, Multi-Topology-SID: a Topology Segment is an IGP segment attached to a IGP multi-topology. A Multi Topology Segment is always global within the SR/IGP domain and identifies an instruction to indicate one specified topology. The Multi-Topology -SID is the SID of the Multi-Topology Segment.

- Maximally Redundant Trees (MRT): A pair of trees where the path from any node X to the root R along the first tree and the path from the same node X to the root along the second tree share the minimum number of nodes and the minimum number of links. Each such shared node is a cut-vertex. Any shared links are cut-links. Any RT is an MRT but many MRTs are not RTs.

- GADAG: Generalized ADAG - a graph that is the combination of the ADAGs of all blocks.

- GADAG root: The root node of GADAG.

- MRT-Red: MRT-Red is used to describe one of the two MRTs; it is used to described the associated forwarding topology and MT-ID. Specifically, MRT-Red is the decreasing MRT where links in the GADAG are taken in the direction from a higher topologically ordered node to a lower one.

- MRT-Blue: MRT-Blue is used to describe one of the two MRTs; it is used to described the associated forwarding topology and MT-ID. Specifically, MRT-Blue is the increasing MRT where links in the GADAG are taken in the direction from a lower topologically ordered node to a higher one.

3. MRT FRR Using Multi-Topology Segment

As described in [I-D.ietf-rtgwg-mrt-frr-architecture], MRT-FRR leverages multi-topologies mechanism to support a pair of Maximally Redundant Trees (MRT) and provides disjoint alternates for a common destination. Unfortunately, IP header could not provides extra bits to indicate its topology. So for the IP network MRT-FRR has to depend on IP tunnels, which will consume more IP addresses.

This dilemma can be solved through allocating segments for the MRT-Red topology and the MRT-Blue topology respectively. These Multi-Topology Segments (Multi-Topology SID) SHOULD have global semantics in the SR domain. Thus multi-topology forwarding can be easily achieved with global segments indicating corresponding topologies. With the help of Segment Routing mechanism, for MRT FRR in the IP network no extra IP address will be introduced and LDP protocol is not necessary to be introduced.

4. Forwarding Mechanisms

4.1. MRT-FRR in SR

As stated before, MRT-FRR in segment routing domain can depend on topology indicating segment to do multi-topology forwarding. There are three types of routers involved into MRT-FRR forwarding.

At the MRT ingress router, the IP packet enters segment routing network for MRT then follows the MRT path to the destination with a Topology Segment inserted into the packet header. In the MRT-FRR scenario, a Multi-Topology Segment indicating MRT-Red or MRT-Blue is used to steer the packet along the alternate path.

At the transit router, a packet is received with one segment indicating the corresponding topology it belongs to. The router will pop the segment, look up the route in the corresponding FIB. When forwarding the packet to the next hop, the multi-topology segment will be pushed again.

At the egress router, the packet will pop the Multi-Topology Segment and continue to forward the packet to the destination.

When MPLS forwarding is applied for the segment routing, the multi-topology segment can map to the MPLS label to indicate the multi-topology.

5. Procedures of Control Plane

IGP extensions should be introduced to carries the topology-id and its corresponding index that determines the actual SID/label value inside the set of all advertised SID/label ranges of a given router. A receiving router uses the index to determine the actual SID/label value in order to identify corresponding topology. The Multi-Topology SID MUST be unique within a given IGP domain. The initiation of advertisement of the Multi-Topology SID binding SHOULD be advertised by a specific node in the SR domain. The node can be specified manually or chosen automatically.

6. IANA Considerations

This document makes no request of IANA.

7. Security Considerations

This document does not introduce new security threat.

8. References

8.1. Normative References

[I-D.ietf-rtgwg-mrt-frr-architecture]
Atlas, A., Kebler, R., Bowers, C., Envedi, G., Csaszar, A., Tantsura, J., and R. White, "An Architecture for IP/LDP Fast-Reroute Using Maximally Redundant Trees", draft-ietf-rtgwg-mrt-frr-architecture-05 (work in progress), January 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.

8.2. Informative References

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

Authors' Addresses

Zhenbin Li
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Nan Wu
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: eric.wu@huawei.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

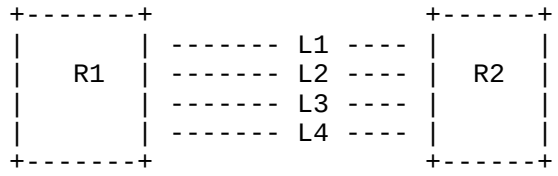
3.1. Adjacency SID properties

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

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

G. Mirsky
J. Tantsura
Ericsson
I. Varlashkin
Google
M. Chen
Huawei

March 5, 2015

Bidirectional Forwarding Detection (BFD) Directed Return Path
draft-mirsky-mpls-bfd-directed-03

Abstract

Bidirectional Forwarding Detection (BFD) is expected to monitor bi-directional paths. When a BFD session monitors in its forward direction an explicitly routed path there is a need to be able to direct egress BFD peer to use specific path as reverse direction of the BFD session.

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

RFC 5880 [RFC5880], RFC 5881 [RFC5881], and RFC 5883 [RFC5883] established the BFD protocol for IP networks and RFC 5884 [RFC5884] set rules of using BFD asynchronous mode over IP/MPLS LSPs. All standards implicitly assume that the egress BFD peer will use the shortest path route regardless of route being used to send BFD control packets towards it. As result, if the ingress BFD peer sends its BFD control packets over explicit path that is diverging from the best route, then reverse direction of the BFD session is likely not to be on co-routed bi-directional path with the forward direction of the BFD session. And because BFD control packets are not guaranteed to cross the same links and nodes in both directions detection of Loss of Continuity (LoC) defect in forward direction may demonstrate positive negatives.

This document defines the extension to LSP Ping [RFC4379], BFD Reverse Path TLV, and proposes that it to be used to instruct the

egress BFD peer to use explicit path for its BFD control packets associated with the particular BFD session. The TLV will be allocated from the TLV and sub-TLV registry defined by RFC 4379 [RFC4379]. As a special case, forward and reverse directions of the BFD session can form bi-directional co-routed associated channel.

1.1. Conventions used in this document

1.1.1. Terminology

BFD: Bidirectional Forwarding Detection

MPLS: Multiprotocol Label Switching

LSP: Label Switching Path

LoC: Loss of Continuity

1.1.2. Requirements Language

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

2. Problem Statement

BFD is best suited to monitor bi-directional co-routed paths. In most cases, given stable environments, the forward and reverse direction between two nodes is likely to be co-routed, this fulfilling the implicit BFD requirements. If BFD is used to monitor unidirectional explicitly routed paths, e.g. MPLS-TE LSPs, its control packets in forward direction would be in-band using the mechanism defined in [RFC5884] and [RFC5586]. But the reverse direction of the BFD session would still follow the shortest path route and that might lead to the following problems detecting failures on the unidirectional explicit path:

- failure detection on the reverse path cannot be interpreted as bi-directional failure and thus trigger, for example, protection switchover of the forward direction;
- if reverse direction is in Down state, the head-end node would not receive indication of forward direction failure from its egress peer.

To address these challenges the egress BFD peer should be instructed to use specific path for its control packets.

3. Direct Reverse BFD Path

3.1. Case of MPLS Data Plane

LSP ping, defined in [RFC4379], uses BFD Discriminator TLV [RFC5884] to bootstrap a BFD session over an MPLS LSP. This document defines a new TLV, BFD Reverse Path TLV, that MUST contain a single sub-TLV that can be used to carry information about reverse path for the specified in BFD Discriminator TLV session.

3.1.1. BFD Reverse Path TLV

The BFD Reverse Path TLV is an optional TLV within the LSP ping protocol. However, if used, the BFD Discriminator TLV MUST be included in an Echo Request message as well. If the BFD Discriminator TLV is not present when the BFD Reverse Path TLV is included, then it MUST be treated as malformed Echo Request, as described in [RFC4379].

The BFD Reverse Path TLV carries the specified path that BFD control packets of the BFD session referenced in the BFD Discriminator TLV are required to follow. The format of the BFD Reverse Path TLV is as presented in Figure 1.

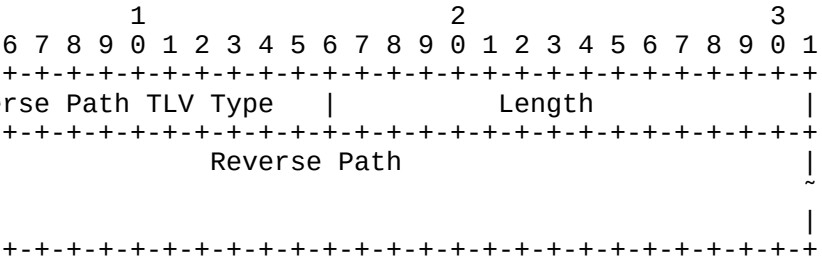


Figure 1: BFD Reverse Path TLV

BFD Reverse Path TLV Type is 2 octets in length and value to be assigned by IANA.

Length is 2 octets in length and defines the length in octets of the Reverse Path field.

Reverse Path field contains a sub-TLV. Any Target FEC sub-TLV, already or in the future defined, from IANA sub-registry Sub-TLVs for TLV Types 1, 16, and 21 of MPLS LSP Ping Parameters registry MAY be used in this field. Only one sub-TLV MUST be included in the Reverse Path TLV. If more than one sub-TLVs are present in the Reverse Path

TLV, then only the first sub-TLV MUST be used and the rest MUST be silently discarded.

If the egress LSR fails to establish the BFD session because path specified in the Reverse Path TLV is not known, the egress MAY establish the BFD session over IP network [RFC5884] and MAY send Echo Reply with the Reverse Path TLV received and the return code set to "Failed to establish the BFD session". The specified reverse path was not found" (TBD4) Section 3.4. If the egress LSR cannot find path specified in the Reverse Path TLV and does not establish BFD session per RFC 5884, it MUST send Echo Reply with the Reverse Path TLV received and the return code set to "Failed to establish the BFD session. The specified reverse path was not found".

3.1.2. Segment Routing Tunnel sub-TLV

With MPLS data plane explicit path can be either Static or RSVP-TE LSP, or Segment Routing tunnel. In case of Static or RSVP-TE LSP [RFC7110] defined sub-TLVs to identify explicit return path. For the Segment Routing with MPLS data plane case a new sub-TLV is defined in this document as presented in Figure 2.

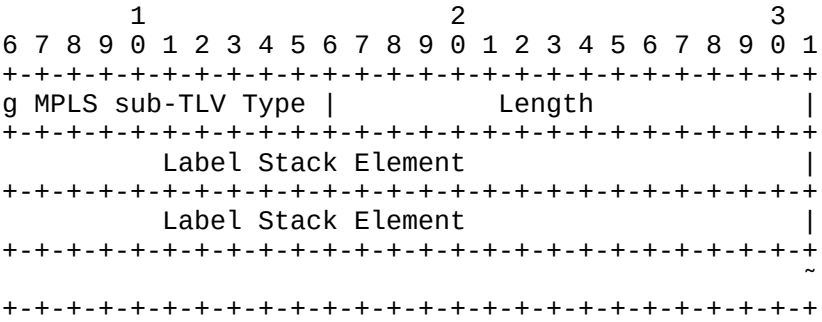


Figure 2: Segment Routing MPLS Tunnel sub-TLV

The Segment Routing Tunnel sub-TLV Type is two octets in length, and will be allocated by IANA.

The Segment Routing Tunnel sub-TLV MAY be used in Reply Path TLV defined in [RFC7110]

3.2. Case of IPv6 Data Plane

IPv6 can be data plane of choice for Segment Routed tunnels [I-D.previdi-6man-segment-routing-header]. In such networks the BFD Reverse Path TLV described in Section 3.1.1 can be used as well. IP networks, unlike IP/MPLS, do not require use of LSP ping with BFD

Discriminator TLV[RFC4379] to bootstrap BFD session. But to specify reverse path of a BFD session in IPv6 environment the BFD Discriminator TLV MUST be used along with the BFD Reverse Path TLV. The BFD Reverse Path TLV in IPv6 network MUST include sub-TLV.

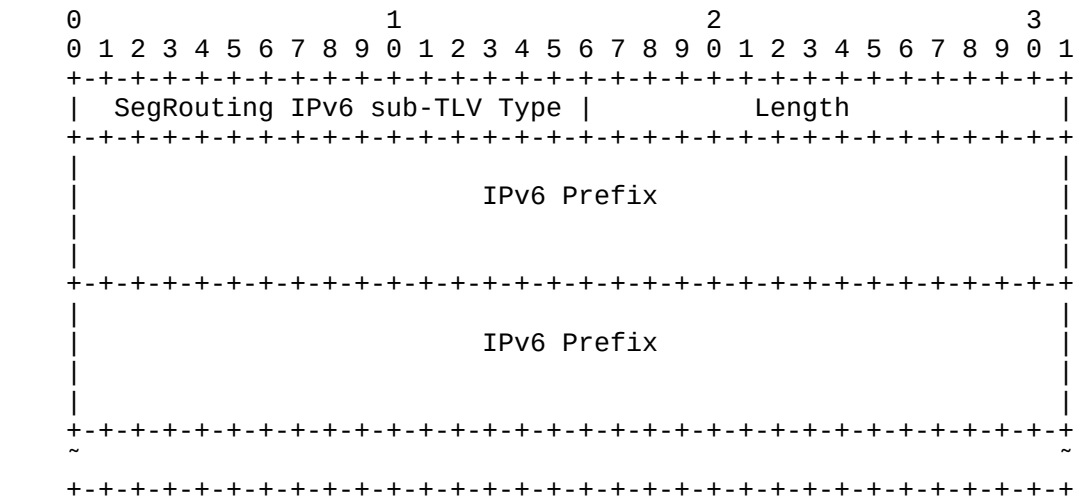


Figure 3: Segment Routing IPv6 Tunnel sub-TLV

3.3. Bootstrapping BFD session with BFD Reverse Path over Segment Routed tunnel

As discussed in [I-D.kumarkini-mpls-spring-lsp-ping] introduction of Segment Routing network domains with MPLS dataplane adds three new sub-TLVs that may be used with Target FEC TLV. Section 6.1 addresses use of new sub-TLVs in Target FEC TLV in LSP ping and LSP traceroute. For the case of LSP ping the [I-D.kumarkini-mpls-spring-lsp-ping] states that:

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

Initiator MAY include FECs corresponding to some or all of segments imposed in the label stack by the initiator to communicate the segments traversed. "

When LSP ping is used to bootstrap BFD session this document updates this and defines that LSP Ping MUST include the FEC corresponding to the destination segment and SHOULD NOT include FECs corresponding to some or all of segment imposed by the initiator. Operationally such restriction would not cause any problem or uncertainty as LSP ping

with FECs corresponding to some or all segments or traceroute may preceed the LSP ping that bootstraps the BFD session.

3.4. Return Codes

This document defines the following Return Codes:

- Failed to establish the BFD session. The specified reverse path was not found, (TBD4) - the specified reverse path was not found, failed to establish the BFD session. When a specified reverse path is not available at the egress LSR, an Echo Reply with the return code set to "Failed to establish the BFD session. The specified reverse path was not found." MAY be sent back to the initiator . (Section 3.1.1)

4. Use Case Scenario

In network presented in Figure 4 node A monitors two tunnels to node H: A-B-C-D-G-H and A-B-E-F-G-H. To bootstrap BFD session to monitor the first tunnel, node A MUST include BFD Discriminator TLV with Discriminator value N and MAY include BFD Reverse Path TLV that references H-G-D-C-B-A tunnel. To bootstrap BFD session to monitor the second tunnel, node A MUST include BFD Discriminator TLV with Discriminator value M and MAY include BFD Reverse Path TLV that references H-G-F-E-B-A tunnel.

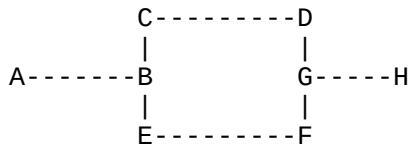


Figure 4: Use Case for BFD Reverse Path TLV

If an operator needs node H to monitor path to node A, e.g. H-G-D-C-B-A tunnel, then by looking up list of known Reverse Paths it MAY find and use existing BFD sessions.

5. IANA Considerations

5.1. TLV

The IANA is requested to assign a new value for BFD Reverse Path TLV from the "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry, "TLVs and sub-TLVs" sub-registry.

Value	Description	Reference
X (TBD1)	BFD Reverse Path TLV	This document

Table 1: New BFD Reverse Type TLV

5.2. Sub-TLV

The IANA is requested to assign two new sub-TLV types from "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry, "Sub-TLVs for TLV Types 1, 16, and 21" sub-registry.

Value	Description	Reference
X (TBD2)	Segment Routing MPLS Tunnel sub-TLV	This document
X (TBD3)	Segment Routing IPv6 Tunnel sub-TLV	This document

Table 2: New Segment Routing Tunnel sub-TLV

5.3. Return Codes

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

Value	Description	Reference
X (TBD4)	Failed to establish the BFD session. The specified reverse path was not found.	This document

Table 3: New Return Code

6. Security Considerations

Secuirity considerations discussed in [RFC5880], [RFC5884], and [RFC4379], apply to this document.

7. Acknowledgements

8. Normative References

[I-D.kumarkini-mpls-spring-lsp-ping]
Kumar, N., Swallow, G., Pignataro, C., Akiya, N., Kini, S., Gredler, H., and M. Chen, "Label Switched Path (LSP) Ping/Trace for Segment Routing Networks Using MPLS Dataplane", draft-kumarkini-mpls-spring-lsp-ping-02 (work in progress), October 2014.

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

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

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, February 2006.

[RFC5586] Bocci, M., Vigoureux, M., and S. Bryant, "MPLS Generic Associated Channel", RFC 5586, June 2009.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, June 2010.

[RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, June 2010.

[RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, June 2010.

[RFC7110] Chen, M., Cao, W., Ning, S., Jounay, F., and S. Delord, "Return Path Specified Label Switched Path (LSP) Ping", RFC 7110, January 2014.

Authors' Addresses

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Jeff Tantsura
Ericsson

Email: jeff.tantsura@ericsson.com

Ilya Varlashkin
Google

Email: Ilya@nobulus.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com