Abstract

draft-ietf-mpls-flow-ident describes the requirement for introducing flow identities within the MPLS architecture. This document describes a method of accomplishing this by using a technique called Synonymous Flow Labels in which labels which mimic the behaviour of other labels provide the identification service. These identifiers can be used to trigger per-flow operations on the on the packet at the receiving label switching router.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 28, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

[I-D.ietf-mpls-flow-ident] describes the requirement for introducing flow identities within the MPLS architecture. This document describes a method of accomplishing this by using a technique called Synonymous Flow Labels (SFL) (see (Section 2)) in which labels which mimic the behaviour of other labels provide the identification service. These identifiers can be used to trigger per-flow operations on the packet at the receiving label switching router.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP
3. Synonymous Flow Labels

An SFL is defined to be a label that causes exactly the same behaviour at the egress Label Switching Router (LSR) as the label it replaces, but in addition also causes an agreed action to take place on the packet. There are many possible additional actions such as the measurement of the number of received packets in a flow, triggering IPFIX inspection, triggering other types of Deep Packet Inspection, or identification of the packet source. In, for example, a Performance Monitoring (PM) application, the agreed action could be the recording of the receipt of the packet by incrementing a packet counter. This is a natural action in many MPLS implementations, and where supported this permits the implementation of high quality packet loss measurement without any change to the packet forwarding system.

Consider an MPLS application such as a pseudowire (PW), and consider that it is desired to use the approach specified in this document to make a packet loss measurement. By some method outside the scope of this text, two labels, synonymous with the PW labels are obtained from the egress terminating provider edge (T-PE). By alternating between these SFLs and using them in place of the PW label, the PW packets may be batched for counting without any impact on the PW forwarding behaviour (note that strictly only one SFL is needed in this application, but that is an optimization that is a matter for the implementor).

Now consider an MPLS application that is multi-point to point such as a VPN. Here it is necessary to identify a packet batch from a specific source. This is achieved by making the SFLs source specific, so that batches from one source are marked differently from batches from another source. The sources all operate independently and asynchronously from each other, independently co-ordinating with the destination. Each ingress is thus able to establish its own SFL to identify the sub-flow and thus enable PM per flow.

Finally we need to consider the case where there is no MPLS application label such as occurs when sending IP over an LSP. In this case introducing an SFL that was synonymous with the LSP label would introduce network wide forwarding state. This would not be acceptable for scaling reasons. We therefore have no choice but to introduce an additional label. Where penultimate hop popping (PHP) is in use, the semantics of this additional label can be similar to the LSP label. Where PHP is not in use, the semantics are similar to
an MPLS explicit NULL. In both of these cases the label has the additional semantics of the SFL.

Note that to achieve the goals set out in Section 1 SFLs need to be allocated from the platform label table.

4. User Service Traffic in the Data Plane

As noted in Section 3 it is necessary to consider two cases:

1. Applications label present
2. Single label stack

4.1. Applications Label Present

Figure 1 shows the case in which both an LSP label and an application label are present in the MPLS label stack. Traffic with no SFL function present runs over the "normal" stack, and SFL enabled flows run over the SFL stack with the SFL used to indicate the packet batch.

```
+-----------------+          +-----------------+
|                 |          |                 |
|      LSP        |          |      LSP        | <May be PHPed
|     Label       |          |     Label       |
+-----------------+          +-----------------+
|                 |          |                 |
|  Application    |          | Synonymous Flow |
|     Label       |          |     Label       |
+-----------------+          +-----------------+ <= Bottom of stack
|                 |          |                 |
|   Payload       |          |   Payload       |
+-----------------+          +-----------------+

"Normal" Label Stack         Label Stack with SFL

Figure 1: Use of Synonymous Labels In A Two Label MPLS Label Stack

At the egress LSR the LSP label is popped (if present). Then the SFL is processed in exactly the same way as the corresponding application label would have been processed.
4.1.1. Setting TTL and the Traffic Class Bits

The TTL and the Traffic Class bits [RFC5462] in the SFL LSE would normally be set to the same value as would have been set in the label that the SFL is synonymous with. However it is recognised that there may be an applications need to set the SFL to some other value. An example would be where it was desired to cause the SFL to trigger an action in the TTL expiry exception path as part of the label action.

4.2. Single Label Stack

Figure 2 shows the case in which only an LSP label is present in the MPLS label stack. Traffic with no SFL function present runs over the "normal" stack and SFL enabled flows run over the SFL stack with the SFL used to indicate the packet batch. However in this case it is necessary for the ingress LSR to first push the SFL and then to push the LSP label.

"Normal" Label Stack          Label Stack with SFL

Figure 2: Use of Synonymous Labels In A Single Label MPLS Label Stack

At the receiving LSR it is necessary to consider two cases:

1. Where the LSP label is still present
2. Where the LSP label is penultimate hop popped

If the LSP label is present, it processed exactly as it would normally processed and then it is popped. This reveals the SFL which in the case of [RFC6374] measurements is simply counted and then discarded. In this respect the processing of the SFL is synonymous...
with an Explicit NULL. As the SFL is the bottom of stack, the IP packet that follows is processed as normal.

If the LSP label is not present due to PHP action in the upstream LSR, two almost equivalent processing actions can take place. Either the SFL can be treated as an LSP label that was not PHPed and the additional associated SFL action is taken when the label is processed. Alternatively, it can be treated as an explicit NULL with associated SFL actions. From the perspective of the measurement system described in this document the behaviour of two approaches are indistinguishable and thus either may be implemented.

4.2.1. Setting TTL and the Traffic Class Bits

The TTL and the Traffic Class considerations described in Section 4.1.1 apply.

4.3. Aggregation of SFL Actions

There are cases where it is desirable to aggregate an SFL action against a number of labels. For example where it is desirable to have one counter record the number of packets received over a group of application labels, or where the number of labels used by a single application is large, and consequently the increase in the number of allocated labels needed to support the SFL actions consequently becomes too large to be viable. In these circumstances it would be necessary to introduce an additional label in the stack to act as an aggregate instruction. This is not strictly a synonymous action in that the SFL is not replacing a existing label, but is somewhat similar to the single label case shown in Section 4.2, and the same signalling, management and configuration tools would be applicable.
The Aggregate SFL is shown in the label stack depicted in Figure 3 as preceding the application label, however the choice of position before, or after, the application label will be application specific. In the case described in Section 4.1, by definition the SFL has the full application context. In this case the positioning will depend on whether the SFL action needs the full context of the application to perform its action and whether the complexity of the application will be increased by finding an SFL following the application label.

5. Equal Cost Multipath Considerations

The introduction to an SFL to an existing flow may cause that flow to take a different path through the network under conditions of Equal Cost Multipath (ECMP). This in turn may invalidate the certain uses of the SFL such as performance measurement applications. Where this is a problem there are two solutions worthy of consideration:

1. The operator can elect to always run with the SFL in place in the MPLS label stack.

2. The operator can elect to use [RFC6790] Entropy Labels in a network that fully supports this type of ECMP. If this approach is adopted, the intervening MPLS network MUST NOT load balance on any packet field other than the entropy label. Note that this is stricter than the text in Section 4.2 of [RFC6790]. In networks
in which the ECMP decision is independent of both the value of any other label in the label stack, and the MPLS payload, the path of the flow with the SFL will be congruent with the path without the SFL.

6. Privacy Considerations

Recent IETF concerns on pervasive monitoring are described in [RFC7258]. The inclusion of originating and/or flow information in a packet provides more identity information and hence potentially degrades the privacy of the communication. Whilst the inclusion of the additional granularity does allow greater insight into the flow characteristics it does not specifically identify which node originated the packet other than by inspection of the network at the point of ingress, or inspection of the control protocol packets. This privacy threat may be mitigated by encrypting the control protocol packets, regularly changing the synonymous labels and by concurrently using a number of such labels. Minimizing the scope of the identity indication can be useful in minimizing the observability of the flow characteristics.

7. Security Considerations

The issue noted in Section 6 is a security consideration. There are no other new security issues associated with the MPLS dataplane. Any control protocol used to request SFLs will need to ensure the legitimacy of the request.

8. IANA Considerations

This draft makes no IANA requests.

9. References

9.1. Normative References


9.2. Informative References

[I-D.ietf-mpls-flow-ident]


Authors’ Addresses

Stewart Bryant
Huawei

Email: stewart.bryant@gmail.com

Mach Chen
Huawei

Email: mach.chen@huawei.com

Zhenbin Li
Huawei

Email: lizhenbin@huawei.com
George Swallow
Cisco Systems
Email: swallow@cisco.com

Siva Sivabalan
Cisco Systems
Email: msiva@cisco.com

Gregory Mirsky
Ericsson
Email: gregory.mirsky@ericsson.com
MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

MPLS Segment Routing in IP Networks
draft-bryant-mpls-unified-ip-sr-03

Abstract

Segment routing is a source routed forwarding method that allows packets to be steered through a network on paths other than the shortest path derived from the routing protocol. The approach uses information encoded in the packet header to partially or completely specify the route the packet takes through the network, and does not make use of a signaling protocol to pre-install paths in the network.

Two different encapsulations have been defined to enable segment routing in an MPLS network or in an IPv6 network. While acknowledging that there is a strong need to support segment routing in both environments, this document defines a mechanism to carry MPLS segment routing packets encapsulated in UDP. The resulting approach is applicable to both IPv4 and IPv6 networks without the need for any changes to the IP or segment routing specifications.

This document makes no changes to the segment routing architecture and builds on existing protocol mechanisms such as the encapsulation of MPLS within UDP defined in RFC 7510.

No new procedures are introduced, but existing mechanisms are combined to achieve the desired result.

Requirements Language

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction ........................................... 3
2. The MPLS-SR-over-UDP Encoding Stack .................... 4
3. The Segment Routing Instruction Stack .................. 5
   3.1. TTL ........................................ 6
4. UDP/IP Encapsulation .................................... 6
5. Elements of Procedure ................................... 6
   5.1. Domain Ingress Nodes .............................. 7
   5.2. Legacy Transit Nodes ............................... 8
   5.3. On-Path Pass-Through SR Nodes ..................... 8
   5.4. SR Transit Nodes ................................ 9
   5.5. Penultimate SR Transit Nodes ...................... 9
   5.6. Domain Egress Nodes ................................ 10
6. A Note on Segment Routing Paths and Penultimate Hop Popping .......................... 11
7. Modes of Deployment ..................................... 11
   7.1. Interconnection of SR Domains ..................... 11
   7.2. SR Within an IP Network ............................ 12
8. Control Plane .......................................... 13
9. OAM .................................................. 14
10. Security Considerations ................................. 14
1. Introduction

Segment routing (SR) [I-D.ietf-spring-segment-routing] is a source routed forwarding method that allows packets to be steered through a network on paths other than the shortest path derived from the routing protocol. SR also allows the packets to be steered through a set of packet processing functions along that path. SR uses information encoded in the packet header to partially or completely specify the route the packet takes through the network and does not make use of a signaling protocol to pre-install paths in the network.

The approach to segment routing in IPv6 networks is known as SRv6 and is described in [I-D.ietf-6man-segment-routing-header]. The mechanism described encodes the segment routing instruction list as an ordered list of 128-bit IPv6 addresses that is carried in a new IPv6 extension header: the Source Routing Header (SRH).

MPLS Segment Routing (MPLS-SR) [I-D.ietf-spring-segment-routing-mpls] encodes the route the packet takes through the network and the instructions to be applied to the packet as it transits the network by imposing a stack of MPLS label stack entries on the packet.

This document describes a method for running SR in IPv4 or IPv6 networks by using an MPLS-SR label stack carried in UDP. No change is made to the MPLS-SR encoding mechanism as described in [I-D.ietf-spring-segment-routing-mpls] where a sequence of 32 bit units, one for each instruction, called the Segment Routing Instruction Stack (SRIS) is used. Each basic unit is encoded as an MPLS label stack entry and the segment routing instructions (i.e., the Segment Identifiers, SIDs) are encoded in the 20 bit MPLS Label fields.

In summary, the processing described in this document is a combination of normal MPLS-over-UDP behavior as described in [RFC7510], MPLS-SR lookup and label-pop behavior as described in [I-D.ietf-spring-segment-routing-mpls], and normal IP forwarding. No new procedures are introduced, but existing mechanisms are combined to achieve the desired result.
The method defined is a complementary way of running SR in an IP network that can be used alongside or interchangeably with that defined in [I-D.ietf-6man-segment-routing-header]. Implementers and deployers should consider the benefits and drawbacks of each method and select the approach most suited to their needs.

2. The MPLS-SR-over-UDP Encoding Stack

The MPLS-SR-over-UDP encoding stack is shown in Figure 1.

```
+---------------------+             +---------------------+
|                     |             |                     |
|      IP Header      |             |      Payload        |
|                     |             |                     |
+---------------------+             +---------------------+
|                     |             |                     |
|     UDP Header      |             |                     |
|                     |             |                     |
+---------------------+             +---------------------+
|                     |             |                     |
| Segment Routing     |             |                     |
| Instruction Stack   |             |                     |
|                     |             |                     |
+---------------------+             +---------------------+
```

**Figure 1: Packet Encapsulation**

The payload may be of any type that, with an appropriate convergence layer, can be carried over a packet network. It is anticipated that the most common packet types will be IPv4, IPv6, native MPLS, and pseudowires [RFC3985].

Preceding the Payload is the Segment Routing Instruction Stack (SRIS) that carries the sequence of instructions to be executed on the packet as it traverses the network. This is the Segment Identifier (SID) stack that is the ordered list of segments described in [I-D.ietf-spring-segment-routing].

Preceding the SRIS is a UDP header. The UDP header is included to:

- Introduce entropy to allow equal-cost multi-path load balancing (ECMP) [RFC2992] in the IP layer [RFC7510].
- Provide a protocol multiplexing layer as an alternative to using a new IP type/next header.
- Allow transit through firewalls and other middleboxes.
- Provide disaggregation.

Preceding the UDP header is the IP header which may be IPv4 or IPv6.

3. The Segment Routing Instruction Stack

The Segment Routing Instruction Stack (SRIS) consists of a sequence of Segment Identifiers (SIDs) as described in [I-D.ietf-spring-segment-routing] encoded as an MPLS label stack as described in [I-D.ietf-spring-segment-routing-mpls].

The top SRIS entry is the next instruction to be executed. When the node to which this instruction is directed has processed the instruction it is removed (popped) from the SRIS, and the next instruction is processed.

Each instruction is encoded in a single Label Stack Entry (LSE) as shown in Figure 2. The structure of the LSE is unchanged from [RFC3032].

```
+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+-------------
defined in [RFC3032] and modified in [RFC5462]. The S bit has bottom
of stack semantics defined in [RFC3032]. TTL is discussed in
Section 3.1.

3.1. TTL

The setting of the TTL is application specific, but the following
operational consideration should be born in mind. In SR the size of
the label stack may be increased within a single routing domain by
various operations such as the pushing of a Binding SID.
Furthermore, in SR packets are not necessarily constrained to travel
on the shortest path within a routing domain. Therefore,
consideration has to be given to the possibility that there may be a
forwarding loop. To mitigate against this it is RECOMMENDED that the
TTL is decremented at each hop as the packet passes through the SR
network regardless of any other changes to the network layer
encapsulation.

Further discussion of the use of TTL during tunnelling can be found
in [RFC4023].

4. UDP/IP Encapsulation

[RFC7510] specifies the values to be used in the UDP Source Port,
Destination Port, and Checksum fields.

An administrative domain, or set of administrative domains that are
sufficiently well managed and monitored to be able to safely use IP
segment routing is likely to comply with the requirements called out
in [RFC7510] to permit operation with a zero UDP checksum over IP.
However each operator needs to validate the decision on whether or
not to use a UDP checksum for themselves.

The [RFC7510] UDP header may be carried over IPv4 or over IPv6.

The IP source address is the address of the encapsulating device.
The IP destination address is implied by the instruction at the top
of the instruction stack.

If IPv4 is in use, fragmentation is not permitted.

5. Elements of Procedure

Nodes that are SR capable can process MPLS-SR packets. Not all of
the nodes in an SR domain are SR capable. Some nodes may be "legacy
routers" that cannot handle SR packets but can forward IP packets.
An SR capable node may advertise its capabilities using the IGP as
described in Section 8. There are six types of node in an SR domain:
Domain ingress nodes that receive packets and encapsulate them for transmission across the domain. Those packets may be any payload protocol including native IP packets or packets that are already MPLS encapsulated.

Legacy transit nodes that are IP routers but that are not SR capable (i.e., are not able to perform segment routing).

Transit nodes that are SR capable but that are not identified by a SID in the SID stack.

Transit nodes that are SR capable and need to perform SR routing because they are identified by a SID in the SID stack.

The penultimate SR capable node on the path that processes the last SID on the stack on behalf of the domain egress node.

The domain egress node that forwards the payload packet for ultimate delivery.

The following sub-sections describe the processing behavior in each case.

In summary, the processing is a combination of normal MPLS-over-UDP behavior as described in [RFC7510], MPLS-SR lookup and label-pop behavior as described in [I-D.ietf-spring-segment-routing-mpls], and normal IP forwarding. No new procedures are introduced, but existing mechanisms are combined to achieve the desired result.

The descriptions in the following sections represent the functional behavior. Optimizations on this behavior may be possible in implementations.

5.1. Domain Ingress Nodes

Domain ingress nodes receive packets from outside the domain and encapsulate them to be forwarded across the domain. Received packets may already be MPLS-SR packets (in the case of connecting two MPLS-SR networks across a native IP network), or may be native IP or MPLS packets.

In the latter case, the packet is classified by the domain ingress node and an MPLS-SR stack is imposed. In the former case the MPLS-SR stack is already in the packet. The top entry in the stack is popped from the stack and retained for use below.

The packet is then encapsulated in UDP with the destination port set to 6635 to indicate "MPLS-UDP" or to 6636 to indicate "MPLS-UDP-DTLS"
as described in [RFC7510]. The source UDP port is set randomly or to provide entropy as described in [RFC7510].

The packet is then encapsulated in IP for transmission across the network. The IP source address is set to the domain ingress node, and the destination address is set to the address corresponding to the label that was previously popped from the stack.

This processing is equivalent to sending the packet out of a virtual interface that corresponds to a virtual link between the ingress node and the next hop SR node realized by a UDP tunnel.

The packet is then sent into the IP network and is routed according to the local FIB and applying hashing to resolve any ECMP choices.

5.2. Legacy Transit Nodes

A legacy transit node is an IP router that has no SR capabilities. When such a router receives an MPLS-SR-in-UDP packet it will carry out normal TTL processing and if the packet is still live it will forward it as it would any other UDP-in-IP packet. The packet will be routed toward the destination indicated in the packet header using the local FIB and applying hashing to resolve any ECMP choices.

If the packet is mistakenly addressed to the legacy router, the UDP tunnel will be terminated and the packet will be discarded either because the MPLS-in-UDP port is not supported or because the uncovered top label has not been allocated. This is, however, a misconnection and should not occur unless there is a routing error.

5.3. On-Path Pass-Through SR Nodes

Just because a node is SR capable and receives an MPLS-SR-in-UDP packet does not mean that it performs SR processing on the packet. Only routers identified by SIDs in the SR stack need to do such processing.

Routers that are not addressed by the destination address in the IP header simply treat the packet as a normal UDP-in-IP packet carrying out normal TTL processing and if the packet is still live routing the packet according to the local FIB and applying hashing to resolve any ECMP choices.

This is important because it means that the SR stack can be kept relatively small and the packet can be steered through the network using shortest path first routing between selected SR nodes.
5.4. SR Transit Nodes

An SR capable node that is addressed by the top most SID in the stack when that is not the last SID in the stack (i.e., the S bit is not set) is an SR transit node. When an SR transit node receives an MPLS-SR-in-UDP packet that is addressed to it, it acts as follows:

- Perform TTL processing as normal for an IP packet.
- Determine that the packet is addressed to the local node.
- Find that the payload is UDP and that the destination port indicates MPLS-in-UDP.
- Strip the IP and UDP headers.
- Pop the top label from the SID stack and retain it for use below.
- Encapsulate the packet in UDP with the destination port set to 6635 (or 6636 for DTLS) and the source port set for entropy. The entropy value SHOULD be retained from the received UDP header or MAY be freshly generated since this is a new UDP tunnel.
- Encapsulate the packet in IP with the IP source address set to this transit router, and the destination address set to the address corresponding to the next SID in the stack.
- Send the packet into the IP network routing the packet according to the local FIB and applying hashing to resolve any ECMP choices.

5.5. Penultimate SR Transit Nodes

The penultimate SR transit node is an SR transit node as described in Section 5.4 where the SID for the node is directly followed by the final SID (i.e., that of domain egress node). When a penultimate SR transit node receives an MPLS-SR-in-UDP packet that is addressed to it, it acts according to whether penultimate hop popping (PHP) is supported for the final SID. That information could be indicated using the control plane as described in Section 8. It is worth making some additional observations about PHP in SR: these are collected in Section 6.

If PHP is allowed the penultimate SR transit node acts as follows:

- Perform TTL processing as normal for an IP packet.
- Determine that the packet is addressed to the local node.
Find that the payload is UDP and that the destination port indicates MPLS-in-UDP.

Strip the IP and UDP headers.

Pop the top label from the SID stack and retain it for use below.

Pop the next label from the SID stack.

Encapsulate the packet in UDP with the destination port set to 6635 (or 6636 for DTLS) and the source port set for entropy. The entropy value SHOULD be retained from the received UDP header or MAY be freshly generated since this is a new UDP tunnel.

Encapsulate the packet in IP with the IP source address set to this transit router, and the destination address set to the domain egress node IP address corresponding to the label that was previously popped from the stack.

Send the packet into the IP network routing the packet according to the local FIB and applying hashing to resolve any ECMP choices.

If PHP is not supported, the penultimate SR transit node just acts as a normal SR transit node just as described in Section 5.4. However, the penultimate SR transit node may be required to replace the final SID with an MPLS-SR label stack entry carrying an explicit null label value (0 for IPv4 and 2 for IPv6) before forwarding the packet. This requirement may also be indicated by the control plane as described in Section 8.

5.6. Domain Egress Nodes

The domain egress acts as follows:

Perform TTL processing as normal for an IP packet.

Determine that the packet is addressed to the local node.

Find that the payload is UDP and that the destination port indicates MPLS-in-UDP.

Strip the IP and UDP headers.

Pop the outermost SID if present (i.e., if PHP was not performed as described in Section 5.5.)
o Pop the explicit null label if it is present in the label stack as requested by the domain egress and communicated in the control plane as described in Section 8.

o Forward the payload packet according to its type and the local routing/forwarding mechanisms.

6. A Note on Segment Routing Paths and Penultimate Hop Popping

End-to-end SR paths are comprised of multiple segments. The end point of each segment is identified by a SID in the SID stack.

In normal SR processing a penultimate hop is the router that performs SR routing immediately prior to the end of segment router. Penultimate hop popping (PHP) is processing that applies at the penultimate router in a segment.

With MPLS-SR-in-UDP encapsulation, each SR segment is achieved using an MPLS-in-UDP tunnel that runs the full length of the segment. The SR SID stack on a packet is only examined at the head and tail of this segment. Thus, each segment is effectively one hop long in the SR overlay network and if there is any PHP processing it takes place at the head-end of the segment.

However, in order to simplify processing at each MPLS-SR-in-UDP end point, it is RECOMMENDED that PHP processing is only used for the final segment in an SR path as described in Section 5.5.

7. Modes of Deployment

As previously noted, the procedures described in this document may be used to connect islands of SR functionality across an IP backbone, or can provide SR function within a native IP network. This section briefly expounds upon those two deployment modes.

7.1. Interconnection of SR Domains

Figure 3 shows two SR domains interconnected by an IP network. The procedures described in this document are deployed at border routers R1 and R2 and packets are carried across the backbone network in a UDP tunnel.

R1 acts as the domain ingress as described in Section 5.1. It takes the MPLS-SR packet from the SR domain, pops the top label and uses it to identify its peer border router R2. R1 then encapsulates the packet in UDP in IP and sends it toward R2.
Routers within the IP network simply forward the packet using normal IP routing.

R2 acts as a domain egress router as described in Section 5.6. It receives a packet that is addressed to it, strips the IP and UDP headers, and acts on the payload SR label stack to continue to route the packet.

7.2. SR Within an IP Network

Figure 4 shows the procedures defined in this document to provide SR function across an IP network.

R1 receives a native packet and classifies it, determining that it should be sent on the SR path R2-R3-R4-R5. It imposes a label stack accordingly and then acts as a domain ingress as described in Section 5.1. It pops the label for R2, and encapsulates the packet in UDP in IP, sets the IP source to R1 and the IP destination to R2, and sends the packet into the IP network.

Routers Ra and Rb are transit routers that simply forward the packets using normal IP forwarding. They may be legacy transit routers (see Section 5.2) or on-path pass-through SR nodes (see Section 5.3).

R2 is an SR transit nodes as described in Section 5.4. It receives a packet addressed to it, strips the IP and UDP headers, and processes the SR label stack. It pops the top label and uses it to identify the next SR hop which is R3. R2 then encapsulates the packet in UDP in IP setting the IP source to R2 and the IP destination to R3.

Rc, Rd, and Re are transit routers and perform as Ra and Rb.
R3 is an SR transit node and performs as R2.

R4 is a penultimate SR transit node as described in Section 5.5. It receives a packet addressed to it, strips the IP and UDP headers, and processes the SR label stack. It pops the top label and uses it to identify the next SR hop which is R5.

R5 is the domain egress as described in Section 5.6. It receives a packet addressed to it, strips the IP and UDP headers.

---|---|---|---|---|---|---|---<br>**Ingress** | **Ra** | **Rb** | **R2** | **R3** | **R4** | **R5** | **Egress**<br>**---**|---|---|---|---|---|---|---<br>**R1** | --- | --- | --- | --- | --- | --- | ---<br>**---**|---|---|---|---|---|---|---<br>**Figure 4: SR Within an IP Network**

8. Control Plane

This document is concerned with forwarding plane issues, and a description of applicable control plane mechanisms is out of scope. This section is provided only as a collection of references. No changes to the control plane mechanisms for MPLS-SR are needed or proposed.

A routers that is able to support SR can advertise the fact in the IGP as follows:

- In IS-IS, by using the SR-Capabilities TLV as defined in [I-D.ietf-isis-segment-routing-extensions]
- In OSPF/OSPFv3 by using the Router Information LSA as defined in [I-D.ietf-ospf-segment-routing-extensions] and [I-D.ietf-ospf-ospfv3-segment-routing-extensions].

Nodes can advertise SIDs using the mechanisms defined in [I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions], or [I-D.ietf-ospf-ospfv3-segment-routing-extensions].
Support for PHP can be indicated in a SID advertisement using flags in the advertisements as follows:

- For IS-IS, the N (no-PHP) flag in the Prefix-SID sub-TLV indicates whether PHP is not to be used.
- For OSPF/OSPFv3, the NP (no-PHP) flag in the Prefix SID Sub-TLV indicates whether PHP is not to be used.

The requirement to use an explicit null SID if PHP is not in use can be indicated in SID advertisement using the Explicit-Null Flag (E-Flag). If set, the penultimate SR transit node replaces the final SID with a SID containing an Explicit-NULL value (0 for IPv4 and 2 for IPv6) before forwarding the packet.

The method of advertising the tunnel encapsulation capability of a router using IS-IS or OSPF are specified in [I-D.ietf-isis-encapsulation-cap] and [I-D.ietf-ospf-encapsulation-cap] respectively. No changes to those procedures are needed in support of this work.

9. OAM

OAM at the payload layer follows the normal OAM procedures for the payload. To the payload the whole SR network looks like a tunnel.

OAM in the IP domain follows the normal IP procedures. This can only be carried out between on the IP hops between pairs of SR nodes.

OAM between instruction processing entities i.e., at the SR layer uses the procedures documented for MPLS.

10. Security Considerations

The security consideration of [I-D.ietf-spring-ipv6-use-cases] and [RFC7510] apply. DTLS [RFC6347] SHOULD be used where security is needed on an MPLS-SR-over-UDP segment.

It is difficult for an attacker to pass a raw MPLS encoded packet into a network and operators have considerable experience at excluding such packets at the network boundaries.

It is easy for an ingress node to detect any attempt to smuggle IP packet into the network since it would see that the UDP destination port was set to MPLS. SR packets not having a destination address terminating in the network would be transparently carried and would pose no security risk to the network under consideration.
11. IANA Considerations

This document makes no IANA requests.

12. Acknowledgements

This draft was partly inspired by
[I-D.xu-mpls-unified-source-routing-instruction], and we acknowledge
the following authors of version -02 of that draft: Robert Raszuk,
Uma Chunduri, Luis M. Contreras, Luay Jalil, Hamid Assarpour, Gunter
Van De Velde, Jeff Tantsura, and Shaowen Ma.

Thanks to Joel Halpern, Bruno Decraene, Loa Andersson, Ron Bonica,
Eric Rosen, Robert Raszuk, Wim Henderickx, Jim Guichard, and Gunter
Van De Velde for their insightful comments on this draft.

13. Contributors

  o Mach Chen, Huawei Technologies, mach.chen@huawei.com

14. References

14.1. Normative References

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B.,
Litkowski, S., and R. Shakir, "Segment Routing
Architecture", draft-ietf-spring-segment-routing-13 (work
in progress), October 2017.

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B.,
Litkowski, S., and R. Shakir, "Segment Routing with MPLS
data plane", draft-ietf-spring-segment-routing-mpls-10
(work in progress), June 2017.

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

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y.,
Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack
Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001,
14.2. Informative References

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B.,
daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d.,
Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi,
T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk,
"IPv6 Segment Routing Header (SRH)", draft-ietf-6man-

[I-D.ietf-isis-encapsulation-cap]
Xu, X., Decraene, B., Raszuk, R., Chunduri, U., Contreras,
L., and L. Jalil, "Advertising Tunnelling Capability in
IS-IS", draft-ietf-isis-encapsulation-cap-01 (work in
progress), April 2017.

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

[I-D.ietf-ospf-encapsulation-cap]
Xu, X., Decraene, B., Raszuk, R., Contreras, L., and L.
Jalil, "The Tunnel Encapsulations OSPF Router
Information", draft-ietf-ospf-encapsulation-cap-09 (work
in progress), October 2017.
[I-D.ietf-ospf-ospfv3-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H.,
Shakir, R., Henderickx, W., and J. Tantsura, "OSPFv3 Extensions for Segment Routing",
draft-ietf-ospf-ospfv3-segment-routing-extensions-10 (work in progress), September 2017.

[I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H.,
Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing",

[I-D.ietf-spring-ipv6-use-cases]
Brzozowski, J., Leddy, J., Filsfils, C., Maglione, R., and
M. Townsley, "IPv6 SPRING Use Cases",
draft-ietf-spring-ipv6-use-cases-11 (work in progress), June 2017.

[I-D.xu-mpls-unified-source-routing-instruction]
Xu, X., Bashandy, A., Assarpour, H., Ma, S., Henderickx,
W., and j. jefftant@gmail.com, "Unified Source Routing Instructions using MPLS Label Stack",

RFC 2992, DOI 10.17487/RFC2992, November 2000,

RFC 3985, DOI 10.17487/RFC3985, March 2005,

"Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)",
RFC 4023, DOI 10.17487/RFC4023, March 2005,

Authors' Addresses

Stewart Bryant (editor)
Huawei

Email: stewart.bryant@gmail.com
Adrian Farrel (editor)
Juniper Networks
Email: afarrel@juniper.net

John Drake
Juniper Networks
Email: jdrake@juniper.net

Jeff Tantsura
Individual
Email: jefftant.ietf@gmail.com
Abstract

Bidirectional Forwarding Detection (BFD) is expected to be able to monitor a wide variety of encapsulations of paths between systems. When a BFD session monitors an explicitly routed unidirectional path there may be a need to direct egress BFD peer to use a specific path for the 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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on April 1, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
1.  Introduction

[RFC5880], [RFC5881], and [RFC5883] established the BFD protocol for IP networks. [RFC5884] and [RFC7726] set rules for using BFD asynchronous mode over IP/MPLS LSPs. These standards do not define means to control the path selection at the egress BFD peer to send BFD control packets towards the ingress BFD system.

For the case when BFD is used to detect defects of the traffic engineered LSP the path the BFD control packets transmitted by the egress BFD system toward the ingress may be disjoint from the LSP in the forward direction. The fact that BFD control packets are not guaranteed to follow the same links and nodes in both forward and reverse directions may be one of the factors contributing to producing false positive defect notifications, i.e., false alarms, at the ingress BFD peer. Ensuring that both directions of the BFD session use co-routed paths may, in some environments, improve the determinism of the failure detection and localization.

This document defines the BFD Reverse Path TLV as an extension to LSP Ping [RFC8029] and proposes that it is to be used to instruct the egress BFD peer to use an explicit path for its BFD control packets associated with a particular BFD session. The TLV will be allocated...
from the TLV and sub-TLV registry defined in [RFC8029]. As a special case, forward and reverse directions of the BFD session can form a bi-directional co-routed associated channel.

1.1. Conventions used in this document

1.1.1. Requirements Language

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

2. Problem Statement

When BFD is used to monitor explicitly routed unidirectional path, e.g., MPLS-TE LSP, BFD 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 follow the shortest path route and that might lead to the problem in detecting failures on an explicit unidirectional path as described below:

- a failure detection by ingress node on the reverse path may not be interpreted as bi-directional failure unambiguously.

To address this scenario, the egress BFD peer would be instructed to use a specific path for BFD control packets.

3. Control of the Reverse BFD Path

To bootstrap a BFD session over an MPLS LSP, LSP ping, defined in [RFC8029], MUST be used with BFD Discriminator TLV [RFC5884]. 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 the reverse path for the BFD session that is specified by the value in BFD Discriminator TLV.

3.1. BFD Reverse Path TLV

The BFD Reverse Path TLV is an optional TLV within the LSP ping [RFC8029]. 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 [RFC8029].
The BFD Reverse Path TLV carries information about the path onto which the egress BFD peer of the BFD session referenced by the BFD Discriminator TLV MUST transmit BFD control packets. The format of the BFD Reverse Path TLV is as presented in Figure 1.

```
+-------------------+---------------+
| BFD Reverse Path TLV Type | Length        |
|--------------------------+---------------|
+-------------------+---------------|
```

Figure 1: BFD Reverse Path TLV

BFD Reverse Path TLV Type is two octets in length and has a value of TBD1 (to be assigned by IANA as requested in Section 5).

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

Reverse Path field contains a sub-TLV. Any non-multicast Target FEC Stack sub-TLV (already defined, or to be defined in the future) for TLV Types 1, 16, and 21 of MPLS LSP Ping Parameters registry MAY be used in this field. Multicast Target FEC Stack sub-TLVs, i.e., p2mp and mp2mp, SHOULD NOT be included in Reverse Path field. If the egress LSR finds multicast Target Stack sub-TLV, it MUST send echo reply with the received Reverse Path TLV, BFD Discriminator TLV and set the Return Code to "Inappropriate Target FEC Stack sub-TLV present" Section 3.3. None, one or more sub-TLVs MAY be included in the BFD Reverse Path TLV. If no sub-TLVs are found in the BFD Reverse Path TLV, the egress BFD peer MUST revert to using the local policy based decision as described in Section 7 [RFC5884], i.e., routed over IP network.

If the egress LSR cannot find the path specified in the Reverse Path TLV it MUST send Echo Reply with the received BFD Discriminator TLV, Reverse Path TLV and set the Return Code to "Failed to establish the BFD session. The specified reverse path was not found" Section 3.3. An implementation MAY provide configuration options to define action at the egress BFD peer. For example, if the egress LSR cannot find the path specified in the Reverse Path TLV it MAY establish the BFD session over IP network as defined in [RFC5884].

3.2. Static and RSVP-TE sub-TLVs

When an explicit path on an MPLS data plane is set either as Static or RSVP-TE LSP, corresponding sub-TLVs, defined in [RFC7110], MAY be used to identify the explicit reverse path for the BFD session. If any of defined in [RFC7110] sub-TLVs used in BFD Reverse Path TLV, then the periodic verification of the control plane against the data plane, as recommended in Section 4 [RFC5884], MUST use the Return Path TLV, as per [RFC7110], with that sub-TLV. By using the LSP Ping with Return Path TLV an operator will be able to verify that the forward LSP and the reverse LSP are mapped to the same FECs as BFD session both at the ingress and the egress systems. Selection and control of the rate of LSP Ping with Return Path TLV follows the [RFC5884] that states: "The rate of generation of these LSP Ping Echo request messages SHOULD be significantly less than the rate of generation of the BFD Control packets. An implementation MAY provide configuration options to control the rate of generation of the periodic LSP Ping Echo request messages."

3.3. Return Codes

This document defines the following Return Codes for MPLS LSP Echo Reply:

- "Inappropriate Target FEC Stack sub-TLV present", (TBD3). When multicast Target FEC Stack sub-TLV found in the received Echo Request by the egress BFD peer, an Echo Reply with the return code set to "Inappropriate Target FEC Stack sub-TLV present" MUST be sent to the ingress BFD peer Section 3.1.

- "Failed to establish the BFD session. The specified reverse path was not found", (TBD4). When a specified reverse path is not available at the egress BFD peer, an Echo Reply with the return code set to "Failed to establish the BFD session. The specified reverse path was not found" MUST be sent back to the ingress BFD peer Section 3.1.

4. Use Case Scenario

In the network presented in Figure 2 node A monitors two tunnels to node H: A-B-C-D-G-H and A-B-E-F-G-H. To bootstrap a BFD session to monitor the first tunnel, node A MUST include a BFD Discriminator TLV with Discriminator value (e.g., foobar-1) and MAY include a BFD Reverse Path TLV that references H-G-D-C-B-A tunnel. To bootstrap a BFD session to monitor the second tunnel, node A MUST include a BFD Discriminator TLV with a different Discriminator value (e.g., foobar-2) [RFC7726] and MAY include a BFD Reverse Path TLV that references H-G-F-E-B-A tunnel.
If an operator needs node H to monitor a path to node A, e.g. H-G-D-C-B-A tunnel, then by looking up the list of known Reverse Paths it MAY find and use the existing BFD session.

5. IANA Considerations

5.1. BFD Reverse Path 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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TBD1)</td>
<td>BFD Reverse Path TLV</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 1: New BFD Reverse Type TLV

5.2. Return Code

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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TBD3)</td>
<td>Inappropriate Target FEC Stack sub-TLV present.</td>
<td>This document</td>
</tr>
<tr>
<td>(TBD4)</td>
<td>Failed to establish the BFD session. The specified reverse path was not found.</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 2: New Return Code
6. Security Considerations

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

7. Normative References


Appendix A. Acknowledgments

Authors greatly appreciate thorough review and the most helpful comments from Eric Gray and Carlos Pignataro.

Authors’ Addresses

Greg Mirsky
ZTE

Email: gregimirsky@gmail.com

Jeff Tantsura
Nuage Networks

Email: jefftant.ietf@gmail.com

Ilya Varlashkin
Google

Email: Ilya@nobulus.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com
Abstract

This document describes a YANG data model for Multi-Protocol Label Switching (MPLS) Label Distribution Protocol (LDP). This model also serves as the base model that is augmented to define Multipoint LDP (mLDP) model.

Status of This Memo

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

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

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

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

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

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

Table of Contents

1. Introduction ................................................. 3
   1.1. Base and Extended .................................. 3
2. Specification of Requirements ................................. 4
3. Overview .................................................. 4
4. Consolidated Tree .......................................... 7
5. Configuration .............................................. 16
   5.1. Configuration Tree .................................... 19
      5.1.1. Base ........................................... 19
      5.1.2. Extended .................................... 20
   5.2. Configuration Hierarchy ............................... 23
      5.2.1. Per-VRF parameters ............................ 23
6. Operational State ........................................... 25
   6.1. Operational Tree ..................................... 25
      6.1.1. Base .......................................... 25
      6.1.2. Extended .................................... 29
   6.2. States .............................................. 31
      6.2.1. Adjacency state ................................ 31
      6.2.2. Peer state ................................... 32
      6.2.3. Bindings state ................................ 33
      6.2.4. Capabilities state ............................ 35
7. Notifications ................................................. 35
8. Actions ..................................................... 36
9. YANG Specification ......................................... 37
   9.1. Base ................................................ 37
   9.2. Extended ........................................... 65
10. Security Considerations ....................................... 85
11. IANA Considerations ......................................... 85
12. Acknowledgments ........................................... 85
13. References ................................................ 87
   13.1. Normative References ................................ 85
   13.2. Informative References ............................. 87
1. Introduction

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

This document introduces a YANG data model for MPLS Label Distribution Protocol (LDP) [RFC5036]. This model also covers LDP IPv6 [RFC7552] and LDP capabilities [RFC5561].

The data model is defined for following constructs that are used for managing the protocol:

- Configuration
- Operational State
- Executables (Actions)
- Notifications

This document is organized to define the data model for each of the above constructs in the sequence as listed above.

1.1. Base and Extended

The configuration and state items are divided into following two broad categories:

- Base
- Extended

The "base" category contains the basic and fundamental features that are covered in LDP base specification [RFC5036] and constitute the minimum requirements for a typical base LDP deployment. Whereas, the "extended" category contains all other non-base features. All the items in a base category are mandatory and hence no "if-feature" is allowed under the "base" category model. The base and extended categories are defined in their own modules as described later.

The example of base feature includes the configuration of LDP lsr-id, enabling LDP interfaces, setting password for LDP session etc.,
whereas the examples of extended feature include inbound/outbound label policies, igp sync, downstream-on-demand etc. This is worth highlighting that LDP IPv6 [RFC7552] is also categorized as an extended feature.

While "base" model support will suffice for small deployments, it is expected that large deployments will require not only the "base" module support from the vendors but also the support for "extended" model for some extended feature(s) of interest.

2. Specification of Requirements

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

In this document, the word "IP" is used to refer to both IPv4 and IPv6, unless otherwise explicitly stated. For example, "IP address family" means and be read as "IPv4 and/or IPv6 address family"

3. Overview

This document defines two new modules for LDP YANG support:

- "ietf-mpls-ldp" module that models the base LDP features and augments /rt:routing/rt:control-plane-protocols defined in [RFC8022]
- extended "ietf-mpls-ldp-extended" module that models the extended LDP features and augments the base LDP

It is to be noted that mLDP data model [I-D.ietf-mpls-mldp-yang] augments LDP base and extended models to model the base and extended mLDP features respectively.

There are four main containers in our module(s):

- Read-Write parameters for configuration (Discussed in Section 5)
- Read-only parameters for operational state (Discussed in Section 6)
- Notifications for events (Discussed in Section 7)
- RPCs for executing commands to perform some action (Discussed in Section 8)
The modeling in this document complies with the Network Management Datastore Architecture (NMDA) [I-D.ietf-netmod-revised-datastores]. The operational state data is combined with the associated configuration data in the same hierarchy [I-D.ietf-netmod-rfc6087bis]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all "config true" (rw) and "config false" (ro) nodes defined in the schema.

Following diagram depicts high level LDP yang tree organization and hierarchy:

```
+-- rw routing
   +-- rw control-plane-protocols
      +-- rw mpls-ldp
      +-- rw ...
      ++-- rw ...
      ++-- ro ...
      ++-- ...
      ++-- ro ...
      ++-- ...
      ++-- ...
      ++-- rw ldp-ext: ....
      ++-- rw ...
      ++-- ro ...
      ++-- ...
      ++-- ro ...
      ++-- ...

rpcs:
   +-- x mpls-ldp-some_action
   +-- x ...

notifications:
   +---- n mpls-ldp-some_event
   +---- n ...
```

Figure 1

Before going into data model details, it is important to take note of the following points:
This module aims to address only the core LDP parameters as per RFC specification, as well as some widely deployed non-RFC features (such as label policies, session authentication etc). Any vendor specific feature should be defined in a vendor-specific augmentation of this model.

Multi-topology LDP [RFC7307] is beyond the scope of this document.

This module does not cover any applications running on top of LDP, nor does it cover any OAM procedures for LDP.

This model is a VPN Forwarding and Routing (VRF)-centric model. It is important to note that [RFC4364] defines VRF tables and default forwarding tables as different, however from a yang modelling perspective this introduces unnecessary complications, hence we are treating the default forwarding table as just another VRF.

A "network-instance", as defined in [I-D.ietf-rtgwg-ni-model], refers to a VRF instance (both default and non-default) within the scope of this model.

This model supports two address-families, namely "ipv4" and "ipv6".

This model assumes platform-wide label space (i.e. label space Id of zero). However, when Upstream Label assignment [RFC6389] is in use, an upstream assigned label is looked up in a Context-Specific label space as defined in [RFC5331].

The label and peer policies (including filters) are defined using a prefix-list. When used for a peer policy, the prefix refers to the LSR Id of the peer. The prefix-list is referenced from routing-policy model as defined in [I-D.ietf-rtgwg-policy-model].

This model uses the terms LDP "neighbor"/"adjacency", "session", and "peer" with the following semantics:

* Neighbor/Adjacency: An LDP enabled LSR that is discovered through LDP discovery mechanisms.

* Session: An LDP neighbor with whom a TCP connection has been established.

* Peer: An LDP session which has successfully progressed beyond its initialization phase and is either already exchanging the bindings or is ready to do so.
It is to be noted that LDP Graceful Restart mechanisms defined in [RFC3478] allow keeping the exchanged bindings for some time after a session goes down with a peer. We call such a state belonging to a "stale" peer -- i.e. keeping peer bindings from a peer with whom currently there is either no connection established or connection is established but GR session is in recovery state. When used in this document, the above terms will refer strictly to the semantics and definitions defined for them.

A graphical tree representation of full LDP YANG data model is presented in Figure 2, whereas LDP configuration (base and extended), state (base and extended), notification, and rpc are graphically represented in Figure 5, Figure 6, Figure 8, Figure 9, Figure 15, and Figure 16 respectively. The actual base and extended model definition in YANG is captured in Section 9.

While presenting the YANG tree view and actual .yang specification, this document assumes readers' familiarity with the concepts of YANG modeling, its presentation and its compilation.

4. Consolidated Tree

Following is a consolidated tree representation of configuration, state, notification, and rpc items under LDP base and extended.

module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
  +++rw mpls-ldp!
    +++rw global
      +++rw capability
        +++rw ldp-ext:end-of-lib {capability-end-of-lib}?
        | +++rw ldp-ext:enable? boolean
        +++rw ldp-ext:typed-wildcard-fec
          {capability-typed-wildcard-fec}?
          | +++rw ldp-ext:enable? boolean
        +++rw ldp-ext:upstream-label-assignment
          {capability-upstream-label-assignment}?
          | +++rw ldp-ext:enable? boolean
      +++rw graceful-restart
        +++rw enable? boolean
        +++rw reconnect-time? uint16
        +++rw recovery-time? uint16
        +++rw forwarding-holdtime? uint16
        +++rw ldp-ext:helper-enable? boolean
        | | {graceful-restart-helper-mode}?
        | +++rw lsr-id? rt-types:router-id
      +++rw address-families
++-rw ipv4!
  ++-rw enable?  boolean
  ++-ro label-distribution-controlmode?  enumeration
  ++-ro bindings
    +++-ro address* [address]
      +++-ro address  inet:ipv4-address
      +++-ro advertisement-type?  advertised-received
      +++-ro peer
        +++-ro lsr-id?  leafref
        +++-ro label-space-id?  leafref
    +++-ro fec-label* [fec]
      +++-ro fec  inet:ipv4-prefix
      +++-ro peer*
        [lsr-id label-space-id advertisement-type]
        +++-ro lsr-id  leafref
        +++-ro label-space-id  leafref
        +++-ro advertisement-type  advertised-received
        +++-ro label?  rt-types:mpls-label
        +++-ro used-in-forwarding?  boolean
  +++-rw ldp-ext:label-policy
    +++-rw ldp-ext:advertise
      +++-rw ldp-ext:egress-explicit-null
      |  +++-rw ldp-ext:enable?  boolean
      |  +++-rw ldp-ext:prefix-list?  prefix-list-ref
      |  +++-rw ldp-ext:accept
      |    +++-rw ldp-ext:prefix-list?  prefix-list-ref
    +++-rw ldp-ext:assign
      {policy-label-assignment-config}?
      |  +++-rw ldp-ext:independent-mode
      |    +++-rw ldp-ext:prefix-list?  prefix-list-ref
      |  +++-rw ldp-ext:ordered-mode
      |    {policy-ordered-label-config}?
      |    +++-rw ldp-ext:egress-prefix-list?  prefix-list-ref
    +++-rw ldp-ext:transport-address?  inet:ipv4-address
  +++-rw ldp-ext:ipv6!
    +++-rw ldp-ext:enable?  boolean
    +++-rw ldp-ext:label-policy
      +++-rw ldp-ext:advertise
        +++-rw ldp-ext:egress-explicit-null
        |  +++-rw ldp-ext:enable?  boolean
        |  +++-rw ldp-ext:prefix-list?  prefix-list-ref
      +++-rw ldp-ext:accept
        |  +++-rw ldp-ext:prefix-list?  prefix-list-ref
      +++-rw ldp-ext:assign
        {policy-label-assignment-config}?
---rw ldp-ext:independent-mode
  | ---rw ldp-ext:prefix-list?  prefix-list-ref
---rw ldp-ext:ordered-mode
  (policy-ordered-label-config)?
    | ---rw ldp-ext:egress-prefix-list?  prefix-list-ref
---rw ldp-ext:transport-address
  | inet:ipv6-address
---ro ldp-ext:label-distribution-controlmode?
    | enumeration
---ro ldp-ext:bindings
  +---ro ldp-ext:address* [address]
    |   | ---ro ldp-ext:address
    |   |     | inet:ipv6-address
    |   | ---ro ldp-ext:advertisement-type?
    |   |     | advertised-received
    |   | ---ro ldp-ext:peer
    |     | ---ro ldp-ext:lsr-id?  leafref
    |     | ---ro ldp-ext:label-space-id?  leafref
    | ---ro ldp-ext:fec-label* [fec]
      | ---ro ldp-ext:fec  inet:ipv6-prefix
      | ---ro ldp-ext:peer*
        | [lsr-id label-space-id advertisement-type]
      | ---ro ldp-ext:lsr-id  leafref
      | ---ro ldp-ext:label-space-id  leafref
      | ---ro ldp-ext:advertisement-type
        | advertised-received
      | ---ro ldp-ext:label?
        | rt-types:mpls-label
      | ---ro ldp-ext:used-in-forwarding?  boolean
---rw ldp-ext:forwarding-nexthop
  | {forwarding-nexthop-config}?  
    | ---rw ldp-ext:interfaces
      | ---rw ldp-ext:interface* [name]
        | ---rw ldp-ext:name  if:interface-ref
      | ---rw ldp-ext:address-family* [afi]
        | ---rw ldp-ext:afi  ldp:ldp-address-family
        | ---rw ldp-ext:ldp-disable?  boolean
      | ---rw ldp-ext:igp-synchronization-delay?  uint16
---rw discovery
  | ---rw interfaces
    | ---rw hello-holdtime?  uint16
    | ---rw hello-interval?  uint16
    | ---rw interface* [name]
      | ---rw name
        | if:interface-ref
      | ---ro next-hello?  uint16
      | ---rw address-families
---rw ipv4!
  ---rw enable?
  ---ro hello-adjacencies
   |  ---ro hello-adjacency* [adjacent-address]
    |   ---ro adjacent-address inet:ipv4-address
    |   ---ro flag* identityref
   ---ro hello-holdtime
    |   ---ro adjacent? uint16
    |   ---ro negotiated? uint16
    |   ---ro remaining? uint16
   ---ro next-hello? uint16
  ---ro statistics
   |   ---ro discontinuity-time
    |      yang:date-and-time
    |   ---ro hello-received? yang:counter64
    |   ---ro hello-dropped? yang:counter64
  ---ro peer
   |   ---ro lsr-id? leafref
   |   ---ro label-space-id? leafref
  ---rw ldp-ext:transport-address? union
---rw ldp-ext:ipv6!
  ---rw ldp-ext:enable?
  ---ro ldp-ext:hello-adjacencies
   |  ---ro ldp-ext:hello-adjacency* [adjacent-address]
    |   ---ro ldp-ext:adjacent-address inet:ipv6-address
    |   ---ro ldp-ext:flag* identityref
   ---ro ldp-ext:hello-holdtime
    |   ---ro ldp-ext:adjacent? uint16
    |   ---ro ldp-ext:negotiated? uint16
    |   ---ro ldp-ext:remaining? uint16
   ---ro ldp-ext:next-hello? uint16
  ---ro ldp-ext:statistics
   |   ---ro ldp-ext:discontinuity-time
    |      yang:date-and-time
    |   ---ro ldp-ext:hello-received? yang:counter64
    |   ---ro ldp-ext:hello-dropped? yang:counter64
  ---ro ldp-ext:peer
   |   ---ro ldp-ext:lsr-id? leafref
   |   ---ro ldp-ext:label-space-id? leafref
  ---rw ldp-ext:transport-address? union
---rw ldp-ext:hello-holdtime? uint16
---rw ldp-ext:hello-interval? uint16
---rw ldp-ext:igp-synchronization-delay? uint16
 {per-interface-timer-config}?
++rw targeted
++rw hello-holdtime? uint16
++rw hello-interval? uint16
++rw hello-accept
   ++rw enable? boolean
   ++rw ldp-ext:neighbor-list? neighbor-list-ref
      [policy-targeted-discovery-config]?
++rw address-families
++rw ipv4!
   ++ro hello-adjacencies
      ++ro hello-adjacency*
         [local-address adjacent-address]
         ++ro local-address inet:ipv4-address
         ++ro adjacent-address inet:ipv4-address
         ++ro flag* identityref
         ++ro hello-holdtime
            ++ro adjacent? uint16
            ++ro negotiated? uint16
            ++ro remaining? uint16
            ++ro next-hello? uint16
         ++ro statistics
            ++ro discontinuity-time
               | yang:date-and-time
            ++ro hello-received?yang:counter64
            ++ro hello-dropped?yang:counter64
         ++ro peer
            ++ro lsr-id? leafref
            ++ro label-space-id? leafref
   ++rw target* [adjacent-address]
      ++rw adjacent-address inet:ipv4-address
      ++rw enable? boolean
      ++rw local-address? inet:ipv4-address
++rw ldp-ext:ipv6!
   ++ro ldp-ext:hello-adjacencies
      ++ro ldp-ext:hello-adjacency*
         [local-address adjacent-address]
         ++ro ldp-ext:local-address inet:ipv6-address
         ++ro ldp-ext:adjacent-address inet:ipv6-address
         ++ro ldp-ext:flag* identityref
         ++ro ldp-ext:hello-holdtime
            ++ro ldp-ext:adjacent? uint16
            ++ro ldp-ext:negotiated? uint16
            ++ro ldp-ext:remaining? uint16
            ++ro ldp-ext:next-hello? uint16
         ++ro ldp-ext:statistics
            ++ro ldp-ext:discontinuity-time
yang:date-and-time

---ro ldp-ext:hello-received?
    yang:counter64
---ro ldp-ext:hello-dropped?
    yang:counter64
---ro ldp-ext:peer
    ---ro ldp-ext:lsr-id?        leafref
    ---ro ldp-ext:label-space-id? leafref
    ++rw ldp-ext:target* [adjacent-address]
    ++rw ldp-ext:adjacent-address     inet:ipv6-address
    ++rw ldp-ext:enable?             boolean
    ++rw ldp-ext:local-address?     inet:ipv6-address

++rw peers
++rw authentication
    ++rw (auth-type-selection)?
        ++:(auth-key)
            ++rw md5-key?        string
            ++:(ldp-ext:auth-key-chain) {key-chain}?
                ++rw ldp-ext:key-chain?    key-chain:key-chain-ref
    ++rw session-ka-holdtime?        uint16
    ++rw session-ka-interval?         uint16
    ++rw peer* [lsr-id label-space-id]
        ++rw lsr-id           rt-types:router-id
        ++rw label-space-id   uint16
    ++rw authentication
        ++rw (auth-type-selection)?
            ++:(auth-key)
                ++rw md5-key?        string
                ++:(ldp-ext:auth-key-chain) {key-chain}?
                    ++rw ldp-ext:key-chain?    key-chain:key-chain-ref
    ++rw capability
    ++rw address-families
        ++rw ipv4!
            ++ro hello-adjacencies
                ++ro hello-adjacency* [local-address adjacent-address]
                    ++ro local-address     inet:ipv4-address
                    ++ro adjacent-address     inet:ipv4-address
                    ++ro flag*               identityref
                    ++ro hello-holdtime
                        ++ro adjacent?    uint16
                        ++ro negotiated?    uint16
                        ++ro remaining?    uint16
                        ++ro next-hello?    uint16
                    ++ro statistics
                        ++ro discontinuity-time
                            ++ro yang:date-and-time
                            +--ro hello-received?     yang:counter64
++-ro typed-wildcard-fec
  |  +++-ro enable? boolean
  +++-ro upstream-label-assignment
    +++-ro enable? boolean
++-ro session-holdtime
  +++-ro peer? uint16
  +++-ro negotiated? uint16
  +++-ro remaining? uint16
++-ro session-state? enumeration
++-ro tcp-connection
  +++-ro local-address? inet:ip-address
  +++-ro local-port? inet:port-number
  +++-ro remote-address? inet:ip-address
  +++-ro remote-port? inet:port-number
++-ro up-time? string
++-ro statistics
  +++-ro discontinuity-time yang:date-and-time
++-ro received
    +++-ro total-octets? yang:counter64
    +++-ro total-messages? yang:counter64
    +++-ro address? yang:counter64
    +++-ro address-withdraw? yang:counter64
    +++-ro initialization? yang:counter64
    +++-ro keepalive? yang:counter64
    +++-ro label-abort-request? yang:counter64
    +++-ro label-mapping? yang:counter64
    +++-ro label-release? yang:counter64
    +++-ro label-request? yang:counter64
    +++-ro label-withdraw? yang:counter64
    +++-ro notification? yang:counter64
++-ro sent
    +++-ro total-octets? yang:counter64
    +++-ro total-messages? yang:counter64
    +++-ro address? yang:counter64
    +++-ro address-withdraw? yang:counter64
    +++-ro initialization? yang:counter64
    +++-ro keepalive? yang:counter64
    +++-ro label-abort-request? yang:counter64
    +++-ro label-mapping? yang:counter64
    +++-ro label-release? yang:counter64
    +++-ro label-request? yang:counter64
    +++-ro label-withdraw? yang:counter64
    +++-ro notification? yang:counter64
    +++-ro total-addresses? uint32
    +++-ro total-labels? uint32
    +++-rw ldp-ext:admin-down? boolean
      {per-peer-admin-down}?
+--rw ldp-ext:graceful-restart
  |  +--rw ldp-ext:enable?  boolean
  |  +--rw ldp-ext:reconnect-time?  uint16
  |  +--rw ldp-ext:recovery-time?  uint16
  +--rw ldp-ext:session-ka-holdtime?  uint16
  +--rw ldp-ext:session-ka-interval?  uint16
++-rw ldp-ext:session-downstream-on-demand
    (session-downstream-on-demand-config)?
    +--rw ldp-ext:enable?  boolean
    +--rw ldp-ext:peer-list?  peer-list-ref
    ++-rw ldp-ext:dual-stack-transport-perference
        {dual-stack-transport-perference}?
    +--rw ldp-ext:max-wait?  uint16
    +-rw ldp-ext:prefer-ipv4!
        +--rw ldp-ext:peer-list?  peer-list-ref

rpcs:
  +---x mpls-ldp-clear-peer
    +---w input
      |    +---w lsr-id?  leafref
      |    +---w label-space-id?  leafref
    +---x mpls-ldp-clear-hello-adjacency
      +---w input
        |        +---w hello-adjacency
        |           +---w (hello-adjacency-type)?
        |           |              +--:(targeted)
        |           |                  |              +---w targeted!
        |           |                  |                    +---w target-address?  inet:ip-address
        |           |                  |                       +--:(link)
        |           |                  |                     +---w link!
        |           |                  |                           +---w next-hop-interface?  leafref
        |           |                  |                                   +---w next-hop-address?  inet:ip-address
    +---x mpls-ldp-clear-peer-statistics
      +---w input
        |    +---w lsr-id?  leafref
        |    +---w label-space-id?  leafref

notifications:
  +---n mpls-ldp-peer-event
    +--ro event-type?  oper-status-event-type
    |    +--ro peer
    |    |    +---ro lsr-id?  leafref
    |    |    +---ro label-space-id?  leafref
    +---n mpls-ldp-hello-adjacency-event
      +--ro event-type?  oper-status-event-type
      |    +--ro (hello-adjacency-type)?
      |    |    +--:(targeted)
      |    |    |    +--ro targeted
      |    |    |          +--ro target-address?  inet:ip-address
5. Configuration

This specification defines the configuration parameters for base LDP as specified in [RFC5036] and LDP IPv6 [RFC7552]. Moreover, it incorporates provisions to enable LDP Capabilities [RFC5561], and defines some of the most significant and commonly used capabilities such as Typed Wildcard FEC [RFC5918], End-of-LIB [RFC5919], and LDP Upstream Label Assignment [RFC6389].

This model augments /rt:routing/rt:control-plane-protocols that is defined in [RFC8022] and follows NMDA as mentioned earlier.

Following is the high-level configuration organization for base LDP:
Following is the high-level configuration organization for extended LDP:

```
 augment /rt:routing/rt:control-plane-protocols:
    +-- mpls-ldp
       +-- global
           +-- ...
           +-- ...
           +-- address-families
               +-- ipv4
                 +-- ...
                 +-- ...
               +-- capability
                 +-- ...
                 +-- ...
       +-- discovery
           +-- interfaces
               +-- ...
               +-- ...
               +-- interface* [interface]
                 +-- ...
                 +-- address-families
                   +-- ipv4
                     +-- ...
                     +-- ...
               +-- targeted
                 +-- ...
                 +-- address-families
                   +-- ipv4
                     +-- target* [adjacent-address]
                       +-- ...
                       +-- ...
       +-- peers
           +-- ...
           +-- ...
           +-- peer* [lsr-id label-space-id]
             +-- ...
             +-- ...
```
++- address-families
  |  ++- ipv4
  |    |  |  ++- ...
  |    |  |  ++- ...
  |    |  |  ++- label-policy
  |    |  |    ++- ...
  |    |  |    ++- ...
  |    ++- ipv6
  |       |  |  ++- ...
  |       |  |  ++- ...
  |       |  |  ++- label-policy
  |       |      ++- ...
  |       |      ++- ...
  ++- capability
    |  ++- ...
    +-- ...
++- discovery
  |  ++- interfaces
  |    |  ++- ...
  |    |    ++- interface* [interface]
  |    |    |  ++- ...
  |    |    |  ++- address-families
  |    |    |    ++- ipv4
  |    |    |    |  ++- ...
  |    |    |    |  ++- ...
  |    |    |    ++- ipv6
  |    |    |      ++- ...
  |    |    |      ++- ...
  |    |    +-- targeteted
  |    |      ++- ...
  |    |      ++- address-families
  |    |      |  ++- ipv6
  |    |      |    ++- target* [adjacent-address]
  |    |      |      ++- ...
  |    |      +-- forwarding-nexthop
  |    |        |  ++- ...
  |    |        |  ++- ...
  |    |        ++- peers
  |    |              ++- ...
  |    |              ++- ...
  |    |              ++- peer*
  |    |              |  ++- ...
  |    |              |  ++- ...
  |    |              |  ++- label-policy
  |    |              |      ++- ...
  |    |              |      ++- ...
  |    |              ++- address-families
Given the configuration hierarchy, the model allows inheritance such that an item in a child tree is able to derive value from a similar or related item in one of the parent. For instance, hello holdtime can be configured per-VRF or per-VRF-interface, thus allowing inheritance as well flexibility to override with a different value at any child level.

5.1. Configuration Tree

5.1.1. Base

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

```
module: ietf-mpls-ldp
    augment /rt: routing/rt: control-plane-protocols:
      +--rw mpls-ldp!
      +--rw global
        +--rw graceful-restart
            +--rw enable? boolean
            +--rw reconnect-time? uint16
            +--rw recovery-time? uint16
            +--rw forwarding-holdtime? uint16
        +--rw lsr-id? rt-types: router-id
        +--rw address-families
            +--rw ipv4!
                +--rw enable? boolean
        +--rw discovery
            +--rw interfaces
                +--rw hello-holdtime? uint16
                +--rw hello-interval? uint16
                +--rw interface* [name] if: interface-ref
                    +--rw address-families
                        +--rw ipv4!
                            +--rw enable? boolean
            +--rw targeted
                +--rw hello-holdtime? uint16
```
5.1.2. Extended

Following is a simplified graphical representation of the data model for LDP extended configuration

module: ietf-mpls-ldp
  augment /rt:routing/rt:control-plane-protocols:
  +--rw mpls-ldp!
    +--rw global
    |    +--rw capability
    |    |    +--rw ldp-ext:end-of-lib {capability-end-of-lib}?
    |    |    |    +--rw ldp-ext:enable?  boolean
    |    |    +--rw ldp-ext:typed-wildcard-fec {capability-typed-wildcard-fec}?
    |    |    |    +--rw ldp-ext:enable?  boolean
    |    |    +--rw ldp-ext:upstream-label-assignment {capability-upstream-label-assignment}?
    |    |    |    +--rw ldp-ext:enable?  boolean
    |    +--rw graceful-restart
    |    |    +--rw ldp-ext:helper-enable?  boolean {graceful-restart-helper-mode}?
    +--rw address-families
---rw ipv4!
  ---rw ldp-ext:label-policy
    ---rw ldp-ext:advertise
      ---rw ldp-ext:egress-explicit-null
      | ---rw ldp-ext:enable?   boolean
      ---rw ldp-ext:prefix-list?   prefix-list-ref
    ---rw ldp-ext:accept
      ---rw ldp-ext:prefix-list?   prefix-list-ref
    ---rw ldp-ext:advertise (policy-label-assignment-config)?
      ---rw ldp-ext:independent-mode
      | ---rw ldp-ext:prefix-list?   prefix-list-ref
      ---rw ldp-ext:ordered-mode (policy-ordered-label-config)?
        +---rw ldp-ext:egress-prefix-list?   prefix-list-ref
    ---rw ldp-ext:transport-address?   union
  ---rw ldp-ext:ipv6!
    ---rw ldp-ext:enable?   boolean
    ---rw ldp-ext:label-policy
      ---rw ldp-ext:advertise
        ---rw ldp-ext:egress-explicit-null
        | ---rw ldp-ext:enable?   boolean
        ---rw ldp-ext:prefix-list?   prefix-list-ref
      ---rw ldp-ext:accept
        ---rw ldp-ext:prefix-list?   prefix-list-ref
      ---rw ldp-ext:assign (policy-label-assignment-config)?
        ---rw ldp-ext:independent-mode
        | ---rw ldp-ext:prefix-list?   prefix-list-ref
      ---rw ldp-ext:ordered-mode (policy-ordered-label-config)?
        +---rw ldp-ext:egress-prefix-list?   prefix-list-ref
    ---rw ldp-ext:transport-address?   union
  ---rw ldp-ext:forwarding-nexthop (forwarding-nexthop-config)?
    ---rw ldp-ext:interfaces
      ---rw ldp-ext:interface* [name]
        ---rw ldp-ext:name    if:interface-ref
      ---rw ldp-ext:address-family* [afi]
        ---rw ldp-ext:afi    ldp:ldp-address-family
      ---rw ldp-ext:ldp-disable?   boolean
    ---rw ldp-ext:igp-synchronization-delay?   uint16
---rw discovery
  ---rw interfaces
    ---rw interface* [name]
      ---rw name if:interface-ref
    ---rw address-families
      ---rw ipv4!
        ---rw ldp-ext:transport-address?   union
      ---rw ldp-ext:ipv6!
      ---rw ldp-ext:enable?   boolean
        ---rw ldp-ext:transport-address?   union
      ---rw ldp-ext:hello-holdtime?   uint16
Internet-Draft        YANG Data Model for MPLS LDP            March 2018

| | +--rw ldp-ext:hello-interval?             uint16
| +--rw ldp-ext:igp-synchronization-delay?  uint16 {per-interface}

-timer-config)?
| +--rw targeted
|   +--rw hello-accept
|     |   +--rw ldp-ext:neighbor-list? neighbor-list-ref {policy-targeted-disco

vry-config})?
| +--rw address-families
|   +--rw ldp-ext:ipv6!
|     +--rw ldp-ext:target* [adjacent-address]
|     |   +--rw ldp-ext:adjacent-address inet:ipv6-address
|     +--rw ldp-ext:enable?             boolean
|     +--rw ldp-ext:local-address?      inet:ipv6-address

| +--rw peers
|   +--rw authentication
|   |   +--rw (auth-type-selection)?
|   |     +--:(ldp-ext:auth-key-chain) {key-chain}?
|   |     +--rw ldp-ext:key-chain? key-chain:key-chain-ref

| +--rw peer* [lsr-id label-space-id]
|   +--rw lsr-id                         rt-types:router-id
|   +--rw label-space-id                 uint16
|   +--rw authentication
|   |   +--rw (auth-type-selection)?
|   |     +--:(ldp-ext:auth-key-chain) {key-chain}?
|   |     +--rw ldp-ext:key-chain? key-chain:key-chain-ref

| +--rw address-families
|   +--rw ipv4!
|     +--rw ldp-ext:label-policy
|     |   +--rw ldp-ext:advertise
|     |     |   +--rw ldp-ext:prefix-list? prefix-list-ref
|     |     +--rw ldp-ext:accept
|     |         +--rw ldp-ext:prefix-list? prefix-list-ref

|     +--rw ldp-ext:ipv6!
|     +--rw ldp-ext:label-policy
|     +--rw ldp-ext:advertise
|     |   +--rw ldp-ext:prefix-list? prefix-list-ref
|     +--rw ldp-ext:accept
|         +--rw ldp-ext:prefix-list? prefix-list-ref

| +--rw ldp-ext:admin-down?             boolean {per-peer-admin-down}?
| +--rw ldp-ext:graceful-restart
|   |   +--rw ldp-ext:enable?             boolean
|   +--rw ldp-ext:reconnect-time?       uint16
|   +--rw ldp-ext:recovery-time?        uint16
|   +--rw ldp-ext:session-ka-holdtime?  uint16
|   +--rw ldp-ext:session-ka-interval?  uint16

| +--rw ldp-ext:session-downstream-on-demand {session-downstream-on-demand}

nd-config)?
| +--rw ldp-ext:enable?             boolean
| +--rw ldp-ext:peer-list? peer-list-ref

| +--rw ldp-ext:dual-stack-transport-pereference {dual-stack-transport-pereference}?
|     +--rw ldp-ext:max-wait?             uint16
5.2. Configuration Hierarchy

The LDP configuration container is logically divided into following high-level config areas:

- Per-VRF parameters
  - Global parameters
  - Per-address-family parameters
  - LDP Capabilities parameters
  - Hello Discovery parameters
    - interfaces
      - Per-interface:
        - Global
        - Per-address-family
    - targeted
      - Per-target
  - Peer parameters
    - Global
    - Per-peer
    - Per-address-family
  - Forwarding parameters

Following subsections briefly explain these configuration areas.

5.2.1. Per-VRF parameters

LDP module resides under an network-instance and the scope of any LDP configuration defined under this tree is per network-instance (per-VRF). This configuration is further divided into sub categories as follows.

5.2.1.1. Per-VRF global parameters

There are configuration items that are available directly under a VRF instance and do not fall under any other sub tree. Example of such a parameter is LDP LSR id that is typically configured per VRF. To keep legacy LDP features and applications working in an LDP IPv4
networks with this model, this document recommends an operator to pick a routable IPv4 unicast address as an LSR Id.

5.2.1.2. Per-VRF Capabilities parameters

This container falls under global tree and holds the LDP capabilities that are to be enabled for certain features. By default, an LDP capability is disabled unless explicitly enabled. These capabilities are typically used to negotiate with LDP peer(s) the support/non-support related to a feature and its parameters. The scope of a capability enabled under this container applies to all LDP peers in the given VRF instance. There is also a peer level capability container that is provided to override a capability that is enabled/specified at VRF level.

5.2.1.3. Per-VRF Per-Address-Family parameters

Any LDP configuration parameter related to IP address family (AF) whose scope is VRF wide is configured under this tree. The examples of per-AF parameters include enabling LDP for an address family, prefix-list based label policies, and LDP transport address.

5.2.1.4. Per-VRF Hello Discovery parameters

This container is used to hold LDP configuration related to Hello and discovery process for both basic (link) and extended (targeted) discovery.

The "interfaces" is a container to configure parameters related to VRF interfaces. There are parameters that apply to all interfaces (such as hello timers), as well as parameters that can be configured per-interface. Hence, an interface list is defined under "interfaces" container. The model defines parameters to configure per-interface non AF related items, as well as per-interface per-AF items. The example of former is interface hello timers, and example of latter is enabling hellos for a given AF under an interface.

The "targeted" container under a VRF instance allows to configure LDP targeted discovery related parameters. Within this container, the "target" list provides a mean to configure multiple target addresses to perform extended discovery to a specific destination target, as well as to fine-tune the per-target parameters.

5.2.1.5. Per-VRF Peer parameters

This container is used to hold LDP configuration related to LDP sessions and peers under a VRF instance. This container allows to configure parameters that either apply on VRF’s all peers or a subset
(peer-list) of VRF peers. The example of such parameters include authentication password, session KA timers etc. Moreover, the model also allows per-peer parameter tuning by specifying a "peer" list under the "peers" container. A peer is uniquely identified using its LSR Id and hence LSR Id is the key for peer list.

Like per-interface parameters, some per-peer parameters are AF-agnostic (i.e. either non AF related or apply to both IP address families), and some that belong to an AF. The example of former is per-peer session password configuration, whereas the example of latter is prefix-list based label policies (inbound and outbound) that apply to a given peer.

5.2.1.6. Per-VRF Forwarding parameters

This container is used to hold configuration used to control LDP forwarding behavior under a VRF instance. One example of a configuration under this container is when a user wishes to enable neighbor discovery on an interface but wishes to disable use of the same interface as forwarding nexthop. This example configuration makes sense only when there are more than one LDP enabled interfaces towards the neighbor.

6. Operational State

Operational state of LDP can be queried and obtained from read-only state containers that fall under the same tree (/rt:routing/rt:control-plane-protocols/) as the configuration.

6.1. Operational Tree

6.1.1. Base

Following is a simplified graphical representation of the base data model for LDP operational state.

```markdown
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
  ---rw mpls-ldp!
    ---rw global
      ---rw address-families
        ---rw ipv4!
          ---ro label-distribution-controlmode? enumeration
          ---ro bindings
            ---ro address* [address]
              ---ro address inet:ipv4-address
              ---ro advertisement-type? advertised-received
```

++-ro peer
    +-ro lsr-id? leafref
    +-ro label-space-id? leafref
++-ro fec-label* [fec]
    +-ro fec inet:ipv4-prefix
    +-ro peer*
        [lsr-id label-space-id advertisement-type]
        +-ro lsr-id leafref
        +-ro label-space-id leafref
        +-ro advertisement-type advertised-received
        +-ro label? rt-types:mpls-label
        +-ro used-in-forwarding? boolean
++-rw discovery
++-rw interfaces
    +-rw interface* [name]
        +-rw name if:interface-ref
        +-ro next-hello? uint16
    +-rw address-families
        +-rw ipv4!
            +-ro hello-adjacencies
                +-ro hello-adjacency* [adjacent-address]
                    +-ro adjacent-address inet:ipv4-address
                    +-ro flag* identityref
                +-ro hello-holdtime
                    +-ro adjacent? uint16
                    +-ro negotiated? uint16
                    +-ro remaining? uint16
                +-ro next-hello? uint16
            +-ro statistics
                +-ro discontinuity-time
                    | yang:date-and-time
                    +-ro hello-received? yang:counter64
                    +-ro hello-dropped? yang:counter64
            +-ro peer
                +-ro lsr-id? leafref
                +-ro label-space-id? leafref
++-rw targeted
    +-rw address-families
        +-rw ipv4!
            +-ro hello-adjacencies
                +-ro hello-adjacency*
                    [local-address adjacent-address]
                    +-ro local-address inet:ipv4-address
                    +-ro adjacent-address inet:ipv4-address
                    +-ro flag* identityref
                +-ro hello-holdtime
                    +-ro adjacent? uint16
                    +-ro negotiated? uint16
| +--ro remaining?     uint16
| +--ro next-hello?    uint16
+--ro statistics
| +--ro discontinuity-time
|     yang:date-and-time
| +--ro hello-received?    yang:counter64
| +--ro hello-dropped?    yang:counter64
+--ro peer
| +--ro lsr-id?           leafref
| +--ro label-space-id?   leafref
+--rw peers
    +--rw peer* [lsr-id label-space-id]
       +--rw lsr-id                       rt-types:router-id
       +--rw label-space-id               uint16
    +--rw address-families
       +--rw ipv4!
           +--ro hello-adjacencies
             +--ro hello-adjacency*
                 [local-address adjacent-address]
                +--ro local-address           inet:ipv4-address
                +--ro adjacent-address        inet:ipv4-address
                +--ro flag*                  identityref
             +--ro hello-holdtime
                +--ro adjacent?     uint16
                +--ro negotiated?    uint16
                +--ro remaining?     uint16
                +--ro next-hello?    uint16
             +--ro statistics
                +--ro discontinuity-time
                |     yang:date-and-time
                | +--ro hello-received?    yang:counter64
                | +--ro hello-dropped?    yang:counter64
                +--ro interface?        if:interface-ref
    +--ro label-advertisement-mode
       +--ro local?        label-adv-mode
       +--ro peer?         label-adv-mode
       +--ro negotiated?   label-adv-mode
    +--ro next-keep-alive?          uint16
    +--ro received-peer-state
       +--ro graceful-restart
           +--ro enable?         boolean
           +--ro reconnect-time? uint16
           +--ro recovery-time?  uint16
       +--ro capability
           +--ro end-of-lib
           | +--ro enable?         boolean
           +--ro typed-wildcard-fec
           | +--ro enable?         boolean

---ro upstream-label-assignment
  ---ro enable? boolean
---ro session-holdtime
  ---ro peer? uint16
  ---ro negotiated? uint16
  ---ro remaining? uint16
---ro session-state? enumeration
---ro tcp-connection
  ---ro local-address? inet:ip-address
  ---ro local-port? inet:port-number
  ---ro remote-address? inet:ip-address
  ---ro remote-port? inet:port-number
---ro up-time? string
---ro statistics
  ---ro discontinuity-time yang:date-and-time
  ---ro received
    ---ro total-octets? yang:counter64
    ---ro total-messages? yang:counter64
    ---ro address? yang:counter64
    ---ro address-withdraw? yang:counter64
    ---ro initialization? yang:counter64
    ---ro keepalive? yang:counter64
    ---ro label-abort-request? yang:counter64
    ---ro label-mapping? yang:counter64
    ---ro label-release? yang:counter64
    ---ro label-request? yang:counter64
    ---ro label-withdraw? yang:counter64
    ---ro notification? yang:counter64
  ---ro sent
    ---ro total-octets? yang:counter64
    ---ro total-messages? yang:counter64
    ---ro address? yang:counter64
    ---ro address-withdraw? yang:counter64
    ---ro initialization? yang:counter64
    ---ro keepalive? yang:counter64
    ---ro label-abort-request? yang:counter64
    ---ro label-mapping? yang:counter64
    ---ro label-release? yang:counter64
    ---ro label-request? yang:counter64
    ---ro label-withdraw? yang:counter64
    ---ro notification? yang:counter64
  ---ro total-addresses? uint32
  ---ro total-labels? uint32
  ---ro total-fec-label-bindings? uint32
6.1.2. Extended

Following is a simplified graphical representation of the extended data model for LDP operational state.

```yang
module: ietf-mpls-ldp
  augment /rt:routing/rt:control-plane-protocols:
    +++rw mpls-ldp!
      +++rw global
        +++rw address-families
          +++rw ldp-ext:ipv6!
            +++ro ldp-ext:label-distribution-controlmode?
              | enumeration
            +++ro ldp-ext:bindings
              +++ro ldp-ext:address* [address]
                +++ro ldp-ext:address
                  | inet:ipv6-address
                +++ro ldp-ext:advertisement-type?
                  | advertised-received
                +++ro ldp-ext:peer
                  +++ro ldp-ext:lsr-id? leafref
                +++ro ldp-ext:label-space-id? leafref
            +++ro ldp-ext:fec-label* [fec]
              +++ro ldp-ext:fec inet:ipv6-prefix
                +++ro ldp-ext:peer* [lsr-id label-space-id advertisement-type]
                  +++ro ldp-ext:lsr-id leafref
                +++ro ldp-ext:label-space-id leafref
                +++ro ldp-ext:advertisement-type
                  | advertised-received
              +++ro ldp-ext:label?
                | rt-types:mpls-label
              +++ro ldp-ext:used-in-forwarding? boolean
        +++rw discovery
          +++rw interface* [name]
            +++rw name if:interface-ref
          +++rw address-families
            +++rw ldp-ext:ipv6!
              +++ro ldp-ext:hello-adjacencies
                +++ro ldp-ext:hello-adjacency* [adjacent-address]
                  +++ro ldp-ext:adjacent-address inet:ipv6-address
                +++ro ldp-ext:flag* identityref
                +++ro ldp-ext:hello-holdtime
                  | +++ro ldp-ext:adjacent? uint16
                +++ro ldp-ext:negotiated? uint16
                +++ro ldp-ext:remaining? uint16
              +++ro ldp-ext:next-hello? uint16
```
Figure 9
6.2. States

Following are main areas for which LDP operational state is defined:

Neighbor Adjacencies
Peer
Bindings (FEC-label and address)
Capabilities

6.2.1. Adjacency state

Neighbor adjacencies are per address-family hello adjacencies that are formed with neighbors as result of LDP basic or extended discovery. In terms of organization, there is a source of discovery (e.g. interface or target address) along with its associated parameters and one or more discovered neighbors along with neighbor discovery related parameters. For the basic discovery, there could be more than one discovered neighbor for a given source (interface), whereas there is at most one discovered neighbor for an extended discovery source (local-address and target-address). This is also to be noted that the reason for a targeted neighbor adjacency could be either an active source (locally configured targeted) or passive source (to allow any incoming extended/targeted hellos). A neighbor/adjacency record also contains session-state that helps highlight whether a given adjacency has progressed to subsequent session level or to eventual peer level.

Following captures high level tree hierarchy for neighbor adjacency state.
Peer related state is presented under peers tree. This is one of the core state that provides info on the session related parameters (mode, authentication, KA timeout etc.), TCP connection info, hello adjacencies for the peer, statistics related to messages and bindings, and capabilities exchange info.

Following captures high level tree hierarchy for peer state.
6.2.3. Bindings state

Binding state provides information on LDP FEC-label bindings as well as address binding for both inbound (received) as well as outbound (advertised) direction. FEC-label bindings are presented as a FEC-centric view, and address bindings are presented as an address-centric view:
FEC-Label bindings:
FEC 203.0.113.1/32:
  advertised: local-label 16000
  peer 192.0.2.1:0
  peer 192.0.2.2:0
  peer 192.0.2.3:0
  received:
    peer 192.0.2.1:0, label 16002, used-in-forwarding=Yes
    peer 192.0.2.2:0, label 17002, used-in-forwarding=No
FEC 203.0.113.2/32:
  . . .
FEC 198.51.100.0/24:
  . . .

Address bindings:
Addr 192.0.2.10:
  advertised
Addr 192.0.2.1:
  received, peer 192.0.2.1:0
Addr 192.0.2.2:
  received, peer 192.0.2.2:0
Addr 192.0.2.3:
  received, peer 192.0.2.3:0

Figure 12

Note that all local addresses are advertised to all peers and hence no need to provide per-peer information for local address advertisement. Furthermore, note that it is easy to derive a peer-centric view for the bindings from the information already provided in this model.

Following captures high level tree hierarchy for bindings state.
6.2.4. Capabilities state

LDP capabilities state comprise two types of information - global information (such as timer etc.), and per-peer information.

Following captures high level tree hierarchy for LDP capabilities state.

```
---rw mpls-ldp!
  +--rw peers
    +--rw peer* [lsr-id label-space-id]
      +--rw lsr-id yang:dotted-quad
      +--rw label-space-id
      +--ro received-peer-state
        +--ro capability
          +--ro . . .
          +--ro . . .
```

Figure 14

7. Notifications

This model defines a list of notifications to inform client of important events detected during the protocol operation. These events include events related to changes in the operational state of an LDP peer, hello adjacency, and FEC etc. It is to be noted that an
LDP FEC is treated as operational (up) as long as it has at least 1 NHLFE with outgoing label.

Following is a simplified graphical representation of the data model for LDP notifications.

```
module: ietf-mpls-ldp
notifications:
  +++-n mpls-ldp-peer-event
    +--ro event-type?   oper-status-event-type
    +--ro peer
        +--ro lsr-id?           leafref
        +--ro label-space-id?   leafref
  +++-n mpls-ldp-hello-adjacency-event
    +--ro event-type?   oper-status-event-type
    +--ro (hello-adjacency-type)?
        +--:(targeted)
            +--ro targeted
            +--ro target-address?   inet:ip-address
        +--:(link)
            +--ro link
                +--ro next-hop-interface?   if:interface-ref
                +--ro next-hop-address?     inet:ip-address
  +++-n mpls-ldp-fec-event
    +--ro event-type?   oper-status-event-type
    +--ro prefix?       inet:ip-prefix
```

Figure 15

8. Actions

This model defines a list of rpcs that allow performing an action or executing a command on the protocol. For example, it allows to clear (reset) LDP peers, hello-adjacencies, and statistics. The model makes an effort to provide different level of control so that a user is able to either clear all, or clear all for a given type, or clear a specific entity.

Following is a simplified graphical representation of the data model for LDP actions.
module: ietf-mpls-ldp
rpcs:
  +---x mpls-ldp-clear-peer
        +---w input
        |      +---w lsr-id?           leafref
        |      +---w label-space-id?   leafref
  +---x mpls-ldp-clear-hello-adjacency
        +---w input
        |      +---w hello-adjacency
        |      |      +---w (hello-adjacency-type)?
        |      |        +--:(targeted)
        |      |        |      +---w targeted!
        |      |        |      +---w target-address?   inet:ip-address
        |      +--:(link)
        |      |      +---w link!
        |      |      +---w next-hop-interface? leafref
        |      |      +---w next-hop-address?     inet:ip-address
  +---x mpls-ldp-clear-peer-statistics
        +---w input
        |      +---w lsr-id?           leafref
        |      +---w label-space-id?   leafref

Figure 16

9.  YANG Specification

Following are the actual YANG definition (module) for LDP constructs defined earlier in the document.

9.1.  Base

<CODE BEGINS> file "ietf-mpls-ldp@2018-02-28.yang"

module ietf-mpls-ldp {
    namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp";
    prefix "ldp";

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

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

Raza, et al. Expires September 6, 2018 [Page 37]
import ietf-routing {
    prefix "rt";
}

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

import ietf-interfaces {
    prefix "if";
}

import ietf-ip {
    prefix "ip";
}

organization
    "IETF MPLS Working Group";
contact
    "WG Web:  <http://tools.ietf.org/wg/teas/>
    WG List:  <mailto:teas@ietf.org>
    Editor:  Kamran Raza
              <mailto:skraza@cisco.com>
    Editor:  Rajiv Asati
              <mailto:rajiva@cisco.com>
    Editor:  Xufeng Liu
              <mailto:Xufeng_Liu@jabil.com>
    Editor:  Santosh Esale
              <mailto:sesale@juniper.net>
    Editor:  Xia Chen
              <mailto:jescia.chenxia@huawei.com>
    Editor:  Himanshu Shah
              <mailto:hshah@ciena.com>";

description
    "This YANG module defines the essential components for the
    management of Multi-Protocol Label Switching (MPLS) Label
    Distribution Protocol (LDP). It is also the base model to
    be augmented for Multipoint LDP (mLDP).

    Copyright (c) 2018 IETF Trust and the persons identified as
    authors of the code. All rights reserved."
Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info).

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

revision 2018-02-28 {
  description
    "Initial revision."
  reference
    "RFC XXXX: YANG Data Model for MPLS LDP."
}

/*
 * Typedefs
 */
typedef ldp-address-family {
  type identityref {
    base rt:address-family;
  }
  description
    "LDP address family type."
}

typedef duration32-inf {
  type union {
    type uint32;
    type enumeration {
      enum "infinite" {
        description "The duration is infinite."
      }
    }
  }
  units seconds;
  description
    "Duration represented as 32 bit seconds with infinite."
}

typedef advertised-received {
  type enumeration {
    enum advertised {
      description "Advertised information."
    }
    enum received {

typedef downstream-upstream {
    type enumeration {
        enum downstream {
            description "Downstream information.";
        }
        enum upstream {
            description "Upstream information.";
        }
    }
    description "Received or advertised.";
}

typedef label-adv-mode {
    type enumeration {
        enum downstream-unsolicited {
            description "Downstream Unsolicited.";
        }
        enum downstream-on-demand {
            description "Downstream on Demand.";
        }
    }
    description "Label Advertisement Mode.";
}

typedef oper-status-event-type {
    type enumeration {
        enum up {
            value 1;
            description "Operational status changed to up.";
        }
        enum down {
            value 2;
            description "Operational status changed to down.";
        }
    }
    description "Operational status event type for notifications.";
}
/* Identities */

identity adjacency-flag-base {
    description "Base type for adjacency flags."
}

identity adjacency-flag-active {
    base adjacency-flag-base;
    description "This adjacency is configured and actively created."
}

identity adjacency-flag-passive {
    base adjacency-flag-base;
    description "This adjacency is not configured and passively accepted."
}

/* */

* Groupings */

grouping adjacency-state-attributes {
    description "The operational state attributes of an LDP hello adjacency, which can be used for basic and extended discoveris, in IPv4 and IPv6 address families."

    leaf-list flag {
        type identityref {
            base adjacency-flag-base;
        }
        description "On or more flags to indicate whether the adjacency is actively created, passively accepted, or both."
    }

    container hello-holdtime {
        description "Containing hello holdtime state information."
        leaf adjacent {
            type uint16;
            units seconds;
            description "The holdtime value learned from the adjacent LSR."
        }
        leaf negotiated {
            type uint16;
        }
    }

Raza, et al. Expires September 6, 2018 [Page 41]
units seconds;
  description
    "The holdtime negotiated between this LSR and the adjacent
    LSR.";
}
leaf remaining {
  type uint16;
  units seconds;
  description
    "The time remaining until the holdtime timer expires.";
}

leaf next-hello {
  type uint16;
  units seconds;
  description
    "The time when the next Hello message will be sent.";
}

container statistics {
  description
    "Statistics objects.";
  leaf discontinuity-time {
    type yang:date-and-time;
    mandatory true;
    description
      "The time on the most recent occasion at which any one or
      more of this interface’s counters suffered a
discontinuity.  If no such discontinuities have occurred
since the last re-initialization of the local management
subsystem, then this node contains the time the local
management subsystem re-initialized itself.";
  }
  leaf hello-received {
    type yang:counter64;
    description
      "The number of Hello messages received.";
  }
  leaf hello-dropped {
    type yang:counter64;
    description
      "The number of Hello messages dropped.";
  }
} // statistics
} // adjacency-state-attributes
grouping basic-discovery-timers {
  description
  "The timer attributes for basic discovery, used in the
  per-interface setting and in the all-interface setting."

  leaf hello-holdtime {
    type uint16 {
      range 15..3600;
    }
    units seconds;
    default 15;
    description
    "The time interval for which a LDP link Hello adjacency
    is maintained in the absence of link Hello messages from
    the LDP neighbor";
  }
  leaf hello-interval {
    type uint16 { 
      range 5..1200;
    }
    units seconds;
    default 5;
    description
    "The interval between consecutive LDP link Hello messages
    used in basic LDP discovery";
  }
} // basic-discovery-timers

grouping binding-address-state-attributes {
  description
  "Operational state attributes of an address binding, used in
  IPv4 and IPv6 address families.";

  leaf advertisement-type {
    type advertised-received;
    description
    "Received or advertised.";
  }
  container peer {
    when ".../advertisement-type = 'received'" {
      description
      "Applicable for received address.";
    }
    description
    "LDP peer from which this address is received.";
    uses ldp-peer-ref;
  }
} // binding-address-state-attributes
grouping binding-label-state-attributes {
    description
        "Operational state attributes for a FEC-label binding, used in
        IPv4 and IPv6 address families.";

    list peer {
        key "lsr-id label-space-id advertisement-type";
        description
            "List of advertised and received peers.";
        uses ldp-peer-ref {
            description
                "The LDP peer from which this binding is received, or to
                which this binding is advertised.
                The peer is identified by its LDP ID, which consists of
                the LSR ID and the Label Space ID.";
        }
        leaf advertisement-type {
            type advertised-received;
            description
                "Received or advertised.";
        }
        leaf label {
            type rt-types:mpls-label;
            description
                "Advertised (outbound) or received (inbound)
                label.";
        }
        leaf used-in-forwarding {
            type boolean;
            description
                "'true' if the label is used in forwarding.";
        }
    } // peer
} // binding-label-state-attributes

grouping graceful-restart-attributes-per-peer {
    description
        "Per peer graceful restart attributes.
        On the local side, these attributes are configuration and
        operational state data. One the peer side, these attributes
        are operational state data received from the peer.";

    container graceful-restart {
        description
            "Attributes for graceful restart.";
        leaf enable {
            type boolean;
            default false;
description
   "Enable or disable graceful restart."
}
leaf reconnect-time {
    type uint16 {
        range 10..1800;
    }
    units seconds;
    default 120;
    description
    "Specifies the time interval that the remote LDP peer
     must wait for the local LDP peer to reconnect after the
     remote peer detects the LDP communication failure."
}
leaf recovery-time {
    type uint16 {
        range 30..3600;
    }
    units seconds;
    default 120;
    description
    "Specifies the time interval, in seconds, that the remote
     LDP peer preserves its MPLS forwarding state after
     receiving the Initialization message from the restarted
     local LDP peer."
}
} // graceful-restart
} // graceful-restart-attributes-per-peer

grouping ldp-interface-ref {
    description
    "Defining a reference to LDP interface."

    leaf name {
        type if:interface-ref;
        must "/if:interfaces/if:interface[if:name=current()]/ip:ipv4"
          + " or "
          + "(/if:interfaces/if:interface[if:name=current()]/ip:ipv6)"
        {
            description "Interface is IPv4 or IPv6."
        }
        description
        "The name of an LDP interface."
    }
}

grouping ldp-peer-ref {
    description
"An absolute reference to an LDP peer, by the LDP ID, which consists of the LSR ID and the Label Space ID."

leaf lsr-id {
  type leafref {
  }
  description
  "The LSR ID of the peer, as a portion of the peer LDP ID.";
}

leaf label-space-id {
  type leafref {
  }
  description
  "The Label Space ID of the peer, as a portion of the peer LDP ID.";
}

} // ldp-peer-ref

grouping ldp-peer-ref-container {
  description
  "A container containing an absolute reference to an LDP peer."
}

    container peer {
      description
      "Reference to an LDP peer, by the LDP ID, which consists of the LSR ID and the Label Space ID.";
      uses ldp-peer-ref;
    } // peer
  } // ldp-peer-ref

    grouping peer-attributes {
      description
      "Peer configuration attributes, used in the per-peer setting can in the all-peer setting."
    }

    leaf session-ka-holdtime {
      type uint16 {
        range 45..3600;
      }
      units seconds;
      default 180;
      description
      "The time interval after which an inactive LDP session terminates and the corresponding TCP session closes.";
    }
Inactivity is defined as not receiving LDP packets from the peer.

leaf session-ka-interval {
  type uint16 {
    range 15..1200;
  }
  units seconds;
  default 60;
  description
  "The interval between successive transmissions of keepalive packets. Keepalive packets are only sent in the absence of other LDP packets transmitted over the LDP session."
}

} // peer-attributes

grouping peer-authentication {
  description
  "Peer authentication container, used in the per-peer setting can in the all-peer setting."
  container authentication {
    description
    "Containing authentication information.";
    choice auth-type-selection {
      description
      "Options for expressing authentication setting.";
      case auth-key {
        leaf md5-key {
          type string;
          description
          "MD5 Key string.";
        }
      }
    }
  }
  } // authentication
} // peer-authentication

grouping peer-state-derived {
  description
  "The peer state information derived from the LDP protocol operations."
  container label-advertisement-mode {
    config false;
    description "Label advertisement mode state."
    leaf local {
      type label-adv-mode;
    }
  }
} // peer-state-derived
leaf peer {
    type label-adv-mode;
    description
        "Peer Label Advertisement Mode.";
}
leaf negotiated {
    type label-adv-mode;
    description
        "Negotiated Label Advertisement Mode.";
}
leaf next-keep-alive {
    type uint16;
    units seconds;
    config false;
    description "Time to send the next KeepAlive message.";
}
container received-peer-state {
    config false;
    description
        "Operational state information learned from the peer.";
    uses graceful-restart-attributes-per-peer;
    container capability {
        description "Configure capability.";
        container end-of-lib {
            description
                "Configure end-of-lib capability.";
            leaf enable {
                type boolean;
                description
                    "Enable end-of-lib capability.";
            }
        }
        container typed-wildcard-fec {
            description
                "Configure typed-wildcard-fec capability.";
            leaf enable {
                type boolean;
                description
                    "Enable typed-wildcard-fec capability.";
            }
        }
    }
}
container upstream-label-assignment {
    description "Configure upstream label assignment capability.";
    leaf enable {
        type boolean;
        description "Enable upstream label assignment.";
    }
}
}
} // capability
} // received-peer-state

container session-holdtime {
    config false;
    description "Session holdtime state.";
    leaf peer {
        type uint16;
        units seconds;
        description "Peer holdtime.";
    }
    leaf negotiated {
        type uint16;
        units seconds;
        description "Negotiated holdtime.";
    }
    leaf remaining {
        type uint16;
        units seconds;
        description "Remaining holdtime.";
    }
}
} // session-holdtime

leaf session-state {
    type enumeration {
        enum non-existent {
            description "NON EXISTENT state. Transport disconnected.";
        }
        enum initialized {
            description "INITIALIZED state.";
        }
        enum openrec {
            description "OPENREC state.";
        }
        enum opensent {
            description "OPENSENT state.";
        }
        enum operational {
            description "OPERATIONAL state.";
        }
    }
}
config false;
description
  "Representing the operational status of the LDP session."
reference
  "RFC5036, Sec. 2.5.4.";
}

container tcp-connection {
  config false;
description
  "TCP connection state.";
  leaf local-address {
    type inet:ip-address;
description "Local address.";
  }
  leaf local-port {
    type inet:port-number;
description "Local port number.";
  }
  leaf remote-address {
    type inet:ip-address;
description "Remote address.";
  }
  leaf remote-port {
    type inet:port-number;
description "Remote port number.";
  }
}

leaf up-time {
  type string;
  config false;
description "Up time. The interval format in ISO 8601.";
}

container statistics {
  config false;
description
  "Statistics objects.";

  leaf discontinuity-time {
    type yang:date-and-time;
    mandatory true;
description
    "The time on the most recent occasion at which any one or
    more of this interface’s counters suffered a
    discontinuity. If no such discontinuities have occurred
since the last re-initialization of the local management subsystem, then this node contains the time the local management subsystem re-initialized itself.;

} container received {
    description "Inbound statistics.";
    uses statistics-peer-received-sent;
}

container sent {
    description "Outbound statistics.";
    uses statistics-peer-received-sent;
}

leaf total-addresses {
    type uint32;
    description "The number of learned addresses."
}

leaf total-labels {
    type uint32;
    description "The number of learned labels."
}

leaf total-fec-label-bindings {
    type uint32;
    description "The number of learned label-address bindings."
}

} // statistics

} // peer-state-derived

grouping statistics-peer-received-sent {
    description "Inbound and outbound statistic counters."
    leaf total-octets {
        type yang:counter64;
        description "The total number of octets sent or received."
    }
    leaf total-messages {
        type yang:counter64;
        description "The number of messages sent or received."
    }
    leaf address {
        type yang:counter64;
        description
leaf address-withdraw {
    type yang:counter64;
    description
        "The number of address-withdraw messages sent or received.";
}

leaf initialization {
    type yang:counter64;
    description
        "The number of initialization messages sent or received.";
}

leaf keepalive {
    type yang:counter64;
    description
        "The number of keepalive messages sent or received.";
}

leaf label-abort-request {
    type yang:counter64;
    description
        "The number of label-abort-request messages sent or received.";
}

leaf label-mapping {
    type yang:counter64;
    description
        "The number of label-mapping messages sent or received.";
}

leaf label-release {
    type yang:counter64;
    description
        "The number of label-release messages sent or received.";
}

leaf label-request {
    type yang:counter64;
    description
        "The number of label-request messages sent or received.";
}

leaf label-withdraw {
    type yang:counter64;
    description
        "The number of label-withdraw messages sent or received.";
}

leaf notification {
    type yang:counter64;
    description
        "The number of messages sent or received.";
}
augment "/rt:routing/rt:control-plane-protocols" {
    description "LDP augmentation.";

carrier mpls-ldp {
    presence "Enables the LDP protocol.";
    description "Containing configuration and operational data for the LDP protocol.";

carrier global {
    description "Global attributes for LDP.";

carrier capability {
    description "Containing the LDP capability data. The container is used for augmentations.";
    reference "RFC5036: Sec. 1.5.";
} // capability

carrier graceful-restart {
    description "Attributes for graceful restart.";
    leaf enable {
        type boolean;
        default false;
        description "Enable or disable graceful restart.";
    }
    leaf reconnect-time {
        type uint16 {
            range 10..1800;
        }
        units seconds;
        default 120;
        description "Specifies the time interval that the remote LDP peer must wait for the local LDP peer to reconnect after the remote peer detects the LDP communication failure.";
    }
} // graceful-restart
leaf recovery-time {
    type uint16 {
        range 30..3600;
    }
    units seconds;
    default 120;
    description
    "Specifies the time interval, in seconds, that the remote LDP peer preserves its MPLS forwarding state after receiving the Initialization message from the restarted local LDP peer."
}

leaf forwarding-holdtime {
    type uint16 {
        range 30..3600;
    }
    units seconds;
    default 180;
    description
    "Specifies the time interval, in seconds, before the termination of the recovery phase."
}

leaf lsr-id {
    type rt-types:router-id;
    description
    "Specify the value to act as the LDP LSR ID. If this attribute is not specified, LDP uses the router ID as determined by the system."
}

container address-families {
    description
    "Per address family configuration and operational state. The address family can be either IPv4 or IPv6."
    container ipv4 {
        presence
        "Present if IPv4 is enabled, unless the 'enable' leaf is set to 'false'"
        description
        "Containing data related to the IPv4 address family."

        leaf enable {
            type boolean;
            default true;
            description
            "The 'enable' leaf controls the enablement state of IPv4. When set to true, IPv4 is enabled; when set to false, IPv4 is disabled."
        }
    }
}

// graceful-restart
leaf label-distribution-controlmode {
  type enumeration {
    enum independent {
      description "Independent label distribution control.";
    }
    enum ordered {
      description "Ordered label distribution control.";
    }
  }
  config false;
  description "Label distribution control mode.";
  reference "RFC5036: LDP Specification. Sec 2.6.";
}

// ipv4 bindings
container bindings {
  config false;
  description "LDP address and label binding information.";
  list address {
    key "address";
    description "List of address bindings learned by LDP.";
    leaf address {
      type inet:ipv4-address;
      description "The IPv4 address learned from an Address message received from or advertised to a peer.";
    }
    uses binding-address-state-attributes;
  } // binding-address
  list fec-label {
    key "fec";
    description "List of FEC-label bindings learned by LDP.";
    leaf fec {
      type inet:ipv4-prefix;
      description "The prefix FEC value in the FEC-label binding, learned in a Label Mapping message received from"
container discovery {
  description
  "Neighbor discovery configuration and operational state.";
}
container interfaces {
  description
  "A list of interfaces for LDP Basic Discovery.";
  reference
  "RFC5036: LDP Specification. Sec 2.4.1.";
  uses basic-discovery-timers;
  list interface {
    key "name";
    description
    "List of LDP interfaces used for LDP Basic Discovery.";
    uses ldp-interface-ref;
    leaf next-hello {
      type uint16;
      units seconds;
      config false;
      description "Time to send the next hello message.";
    }
  }
  container address-families {
    description
    "Container for address families.";
    container ipv4 {
      presence
      "Present if IPv4 is enabled, unless the 'enable'
      leaf is set to 'false'";
      description
      "IPv4 address family.";
      leaf enable {
        type boolean;
        default true;
        description
        "Enable the address family on the interface.";
      }
    }
  }
}
// ipv4
container hello-adjacencies {
  config false;
  description
  "Containing a list of hello adjacencies.";

  list hello-adjacency {
    key "adjacent-address";
    config false;
    description "List of hello adjacencies.";

    leaf adjacent-address {
      type inet:ipv4-address;
      description
      "Neighbor address of the hello adjacency.";
    }

    uses adjacency-state-attributes;
    uses ldp-peer-ref-container;
  } // hello-adjacency
} // hello-adjacencies
} // ipv4
} // address-families
} // list interface
} // interfaces

container targeted {
  description
  "A list of targeted neighbors for extended discovery.";

  leaf hello-holdtime {
    type uint16 {
      range 15..3600;
    }
    units seconds;
    default 45;
    description
    "The time interval for which LDP targeted Hello adjacency is maintained in the absence of targeted Hello messages from an LDP neighbor.";
  }

  leaf hello-interval {
    type uint16 {
      range 5..3600;
    }
  }
}
units seconds;
default 15;
description
"The interval between consecutive LDP targeted Hello
messages used in extended LDP discovery.";
}

container hello-accept {
description
"LDP policy to control the acceptance of extended
neighbor discovery Hello messages.";
leaf enable {
type boolean;
default false;
description
"'true' to accept; 'false' to deny.";
}
}

container address-families {
description
"Container for address families.";
container ipv4 {
presence
"Present if IPv4 is enabled.";
description
"IPv4 address family.";
container hello-adjacencies {
config false;
description
"Containing a list of hello adjacencies.";
list hello-adjacency {
key "local-address adjacent-address";
description "List of hello adjacencies.";
leaf local-address {
type inet:ipv4-address;
description
"Local address of the hello adjacency.";
}
leaf adjacent-address {
type inet:ipv4-address;
description
"Neighbor address of the hello adjacency.";
}
uses adjacency-state-attributes;
uses ldp-peer-ref-container;
} // hello-adjacency
} // hello-adjacencies

list target {
  key "adjacent-address";
  description
  "Targeted discovery params.";

  leaf adjacent-address {
    type inet:ipv4-address;
    description
    "Configures a remote LDP neighbor and enables
    extended LDP discovery of the specified
    neighbor.";
  }

  leaf enable {
    type boolean;
    default true;
    description
    "Enable the target.";
  }

  leaf local-address {
    type inet:ipv4-address;
    description
    "The local address used as the source address to
    send targeted Hello messages.
    If the value is not specified, the
    transport-address is used as the source
    address.";
  }
} // target
} // ipv4
} // address-families
} // targeted
} // discovery

container peers {
  description
  "Peers configuration attributes.";

  uses peer-authentication;
  uses peer-attributes;

  list peer {
    key "lsr-id label-space-id";
description
"List of peers."

leaf lsr-id {
type rt-types:router-id;
description
"The LSR ID of the peer, to identify the globally
unique LSR. This is the first four octets of the LDP
ID. This leaf is used together with the leaf
'label-space-id' to form the LDP ID.";
reference
"RFC5036. Sec 2.2.2."
}

leaf label-space-id {
type uint16;
description
"The Label Space ID of the peer, to identify a specific
label space within the LSR. This is the last two
octets of the LDP ID. This leaf is used together with
the leaf 'lsr-id' to form the LDP ID.";
reference
"RFC5036. Sec 2.2.2."
}

uses peer-authentication;
container capability {
description
"Per peer capability"
}

container address-families {
description
"Per-vrf per-af params.";
container ipv4 {
presence
"Present if IPv4 is enabled.";
description
"IPv4 address family."

container hello-adjacencies {
config false;
description
"Containing a list of hello adjacencies."

list hello-adjacency {
key "local-address adjacent-address";
description "List of hello adjacencies.";
}
leaf local-address {
  type inet:ipv4-address;
  description "Local address of the hello adjacency.";
}
leaf adjacent-address {
  type inet:ipv4-address;
  description "Neighbor address of the hello adjacency.";
}

uses adjacency-state-attributes;

leaf interface {
  type if:interface-ref;
  description "Interface for this adjacency.";
}

} // hello-adjacency
} // hello-adjacencies
} // ipv4
} // address-families

uses peer-state-derived;

} // list peer
} // peers
} // container mpls-ldp

/*
* RPCs
*/
.rpc mpls-ldp-clear-peer {
  description "Clears the session to the peer.";
  input {
    uses ldp-peer-ref {
      description "The LDP peer to be cleared. If this is not provided
      then all peers are cleared.
      The peer is identified by its LDP ID, which consists of
      the LSR ID and the Label Space ID.";
    }
  }
}

.rpc mpls-ldp-clear-hello-adjacency {
  description "Clears the hello adjacency";
}
input {
    container hello-adjacency {
        description "Link adjacency or targeted adjacency. If this is not provided then all hello adjacencies are cleared";
        choice hello-adjacency-type {
            description "Adjacency type.";
            case targeted {
                container targeted {
                    presence "Present to clear targeted adjacencies.";
                    description "Clear targeted adjacencies.";
                    leaf target-address {
                        type inet:ip-address;
                        description "The target address. If this is not provided then all targeted adjacencies are cleared";
                    }
                } // targeted
            }
            case link {
                container link {
                    presence "Present to clear link adjacencies.";
                    description "Clear link adjacencies.";
                    leaf next-hop-interface {
                        type leafref {
                            path "/rt:routing/rt:control-plane-protocols/" + "mpls-ldp/discovery/interfaces/interface/name";
                        }
                        description "Interface connecting to next-hop. If this is not provided then all link adjacencies are cleared.";
                    }
                    leaf next-hop-address {
                        type inet:ip-address;
                        must "./next-hop-interface" {
                            description "Applicable when interface is specified.";
                        }
                        description "IP address of next-hop. If this is not provided then adjacencies to all next-hops on the given interface are cleared.";
                    } // next-hop-address
                } // link
            }
        }
    }
}
rpc mpls-ldp-clear-peer-statistics {
    description
        "Clears protocol statistics (e.g. sent and received
        counters).";
    input {
        uses ldp-peer-ref {
            description
                "The LDP peer whose statistics are to be cleared.
                If this is not provided then all peers’ statistics are
                cleared.
                The peer is identified by its LDP ID, which consists of
                the LSR ID and the Label Space ID."
        }
    }
}

;/* Notifications */
notification mpls-ldp-peer-event {
    description
        "Notification event for a change of LDP peer operational
        status.";
    leaf event-type {
        type oper-status-event-type;
        description "Event type.";
    }
    uses ldp-peer-ref-container;
}

notification mpls-ldp-hello-adjacency-event {
    description
        "Notification event for a change of LDP adjacency operational
        status.";
    leaf event-type {
        type oper-status-event-type;
        description "Event type.";
    }
    choice hello-adjacency-type {
        description
            "Interface or targeted adjacency.";
        case targeted {
            container targeted {
                }
description "Targeted adjacency through LDP extended discovery.";
leaf target-address {
    type inet:ip-address;
    description "The target adjacent address learned.";
}
} // targeted

} // link

notification mpls-ldp-fec-event {
    description "Notification event for a change of FEC status.";
    leaf event-type {
        type oper-status-event-type;
        description "Event type.";
    }
    leaf prefix {
        type inet:ip-prefix;
        description "The address prefix element of the FEC whose status has changed.";
    }
}
9.2. Extended

<CODE BEGINS> file "ietf-mpls-ldp-extended@2018-02-28.yang"

module ietf-mpls-ldp-extended {
    prefix "ldp-ext";

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

    import ietf-routing {
        prefix "rt";
    }

    import ietf-key-chain {
        prefix "key-chain";
    }

    import ietf-mpls-ldp {
        prefix "ldp";
    }

    import ietf-interfaces {
        prefix "if";
    }

    organization
        "IETF MPLS Working Group";
    contact
        "WG Web: <http://tools.ietf.org/wg/teas/>
        WG List: <mailto:teas@ietf.org>
        Editor: Kamran Raza
        <mailto:skraza@cisco.com>
        Editor: Rajiv Asati

Raza, et al. Expires September 6, 2018 [Page 65]
description

"This YANG module defines the extended components for the management of Multi-Protocol Label Switching (MPLS) Label Distribution Protocol (LDP). It is also the model to be augmented for extended Multipoint LDP (mLDP).

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

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

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

revision 2018-02-28 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for MPLS LDP.";
}

/*
 * Features
 */
feature dual-stack-transport-pereference {
  description
    "This feature indicates that the system allows to configure the transport connection pereference in a dual-stack setup.";
}
feature capability-end-of-lib {
  description
    "This feature indicates that the system allows to configure
    LDP end-of-lib capability.";
}

feature capability-typed-wildcard-fec {
  description
    "This feature indicates that the system allows to configure
    LDP typed-wildcard-fec capability.";
}

feature capability-upstream-label-assignment {
  description
    "This feature indicates that the system allows to configure
    LDP upstream label assignment capability.";
}

feature forwarding-nexthop-config {
  description
    "This feature indicates that the system allows to configure
    forwarding nexthop on interfaces.";
}

feature graceful-restart-helper-mode {
  description
    "This feature indicates that the system supports graceful
    restart helper mode.";
}

feature key-chain {
  description
    "This feature indicates that the system supports keychain for
    authentication.";
}

feature per-interface-timer-config {
  description
    "This feature indicates that the system allows to configure
    interface hello timers at the per-interface level.";
}

feature per-peer-admin-down {
  description
    "This feature indicates that the system allows to
    administratively disable a peer.";
}
feature per-peer-graceful-restart-config {
    description
    "This feature indicates that the system allows to configure graceful restart at the per-peer level.";
}

feature per-peer-session-attributes-config {
    description
    "This feature indicates that the system allows to configure session attributes at the per-peer level.";
}

feature policy-label-assignment-config {
    description
    "This feature indicates that the system allows to configure policies to assign labels according to certain prefixes.";
}

feature policy-ordered-label-config {
    description
    "This feature indicates that the system allows to configure ordered label policies.";
}

feature policy-targeted-discovery-config {
    description
    "This feature indicates that the system allows to configure policies to control the acceptance of targeted neighbor discovery hello messages.";
}

feature session-downstream-on-demand-config {
    description
    "This feature indicates that the system allows to configure session downstream-on-demand";
}

/*
 * Typedefs
 */
typedef neighbor-list-ref {
    type string;
    description
    "A type for a reference to a neighbor address list. The string value is the name identifier for uniquely identifying the referenced address list, which contains a list of addresses that a routing policy can applied. The definition of such an address list is outside the scope of this document.";
}
typedef prefix-list-ref {
    type string;
    description
    "A type for a reference to a prefix list. The string value is the name identifier for uniquely identifying the referenced prefix set, which contains a list of prefixes that a routing policy can applied. The definition of such a prefix set is outside the scope of this document.";
}

typedef peer-list-ref {
    type string;
    description
    "A type for a reference to a peer address list. The string value is the name identifier for uniquely identifying the referenced address list, which contains a list of addresses that a routing policy can applied. The definition of such an address list is outside the scope of this document.";
}

/*
 * Identities
 */

/*
 * Groupings
 */

grouping address-family-ipv4-augment {
    description "Augmentation to address family IPv4.";
    uses policy-container;

    leaf transport-address {
        type inet:ipv4-address;
        description
        "The transport address advertised in LDP Hello messages. If this value is not specified, the LDP LSR ID is used as the transport address.";
        reference
        "RFC5036. Sec. 3.5.2.";
    }
} // address-family-ipv4-augment

grouping authentication-keychain-augment {
    description "Augmentation to authentication to add keychain.";
leaf key-chain {
    type key-chain:key-chain-ref;
    description
        "key-chain name.
        If not specified, no key chain is used.";
}
} // authentication-keychain-augment

grouping capability-augment {
    description "Augmentation to capability.";

    container end-of-lib {
        if-feature capability-end-of-lib;
        description
            "Configure end-of-lib capability.";
        leaf enable {
            type boolean;
            default false;
            description
                "Enable end-of-lib capability.";
        }
    }

    container typed-wildcard-fec {
        if-feature capability-typed-wildcard-fec;
        description
            "Configure typed-wildcard-fec capability.";
        leaf enable {
            type boolean;
            default false;
            description
                "Enable typed-wildcard-fec capability.";
        }
    }

    container upstream-label-assignment {
        if-feature capability-upstream-label-assignment;
        description
            "Configure upstream label assignment capability.";
        leaf enable {
            type boolean;
            default false;
            description
                "Enable upstream label assignment.";
        }
    }
} // capability-augment

grouping global-augment {
    description "Augmentation to global attributes.";
leaf igp-synchronization-delay {
  type uint16 {
    range "0 | 3..300";
  }
  units seconds;
  default 0;
  description "Sets the interval that the LDP waits before notifying the Interior Gateway Protocol (IGP) that label exchange is completed so that IGP can start advertising the normal metric for the link. If the value is not specified, there is no delay.";
}
} // global-augment

grouping global-forwarding-nexthop-augment {
  description "Augmentation to global forwarding nexthop interfaces.";
  container forwarding-nexthop {
    if-feature forwarding-nexthop-config;
    description "Configuration for forwarding nexthop."
  }
  container interfaces {
    description "A list of interfaces on which forwarding is disabled."
    list interface {
      key "name";
      description "List of LDP interfaces used for LDP Basic Discovery.";
      uses ldp:ldp-interface-ref;
      list address-family {
        key "afi";
        description "Per-vrf per-af params.";
        leaf afi {
          type ldp:ldp-address-family;
          description "Address family type value."
        }
        leaf ldp-disable {
          type boolean;
          default false;
          description "'true' to disable LDP forwarding on the interface."
        }
      }
    }
  }
}
grouping graceful-restart-augment {
  description "Augmentation to graceful restart."

  leaf helper-enable {
    if-feature graceful-restart-helper-mode;
    type boolean;
    default false;
    description "Enable or disable graceful restart helper mode.;"
  }
}

grouping interface-address-family-ipv4-augment {
  description "Augmentation to interface address family IPv4.";

  leaf transport-address {
    type union {
      type enumeration {
        enum "use-global-transport-address" {
          description "Use the transport address set at the global level
          common for all interfaces for this address family.";
        }
        enum "use-interface-address" {
          description "Use interface address as the transport address.";
        }
      }
      type inet:ipv4-address;
    }
    default "use-global-transport-address";
    description "IP address to be advertised as the LDP transport address.";
  }
}

grouping interface-address-family-ipv6-augment {
  description "Augmentation to interface address family IPv6.";

  leaf transport-address {
    type union {
      type enumeration {

enum "use-global-transport-address" {
    description
    "Use the transport address set at the global level common for all interfaces for this address family.";
}
enum "use-interface-address" {
    description
    "Use interface address as the transport address.";
}
type inet:ipv6-address;
default "use-global-transport-address";
description
"IP address to be advertised as the LDP transport address.";
} // interface-address-family-ipv6-augment

grouping interface-augment {
    description "Augmentation to interface.";
    uses ldp:basic-discovery-timers {
        if-feature per-interface-timer-config;
    }
    leaf igp-synchronization-delay {
        if-feature per-interface-timer-config;
        type uint16 {
            range "0 | 3..300";
        }
        units seconds;
        default 0;
        description
        "Sets the interval that the LDP waits before notifying the Interior Gateway Protocol (IGP) that label exchange is completed so that IGP can start advertising the normal metric for the link. If the value is not specified, there is no delay.";
    }
} // interface-augment

grouping peer-af-policy-container {
    description
    "LDP policy attribute container under peer address-family.";
    container label-policy {
        description
        "Label policy attributes.";
        container advertise {
            description
            "Advertise";
        }
    }
} // peer-af-policy-container
"Label advertising policies."
leaf prefix-list {
  type prefix-list-ref;
  description
  "Applies the prefix list to filter outgoing label
   advertisements.
   If the value is not specified, no prefix filter
   is applied.";
}
}
} // accept
} // label-policy
} // peer-af-policy-container

grouping peer-augment {
  description "Augmentation to each peer list entry.";

  leaf admin-down {
    if-feature per-peer-admin-down;
    type boolean;
    default false;
    description
    "'true' to disable the peer."
  }

  uses ldp:graceful-restart-attributes-per-peer {
    if-feature per-peer-graceful-restart-config;
  }

  uses ldp:peer-attributes {
    if-feature per-peer-session-attributes-config;
  }
} // peer-augment

grouping peers-augment {
  description "Augmentation to peers container.";

container session-downstream-on-demand {
  if-feature session-downstream-on-demand-config;
  description
    "Session downstream-on-demand attributes."
  leaf enable {
    type boolean;
    default false;
    description
      "'true' if session downstream-on-demand is enabled.";
  }
  leaf peer-list {
    type peer-list-ref;
    description
      "The name of a peer ACL, to be applied to the
downstream-on-demand sessions.
If this value is not specified, no filter is applied to
any downstream-on-demand sessions.";
  }
}

container dual-stack-transport-pereference {
  if-feature dual-stack-transport-pereference;
  description
    "The settings of peers to establish TCP connection in a
dual-stack setup.";
  leaf max-wait {
    type uint16 {
      range "0..60";
    }
    default 30;
    description
      "The maximum wait time in seconds for preferred transport
connection establishment. 0 indicates no preference.";
  }
  container prefer-ipv4 {
    presence
      "Present if IPv4 is preferred for transport connection
establishment, subject to the 'peer-list' in this
container.";
    description
      "Uses IPv4 as the preferred address family for transport
connection establishment, subject to the 'peer-list' in
this container.
If this container is not present, as a default, IPv6 is
the preferred address family for transport connection
establishment.";
    leaf peer-list {
      type peer-list-ref;
      description

"The name of a peer ACL, to be applied to the IPv4 transport connections. If this value is not specified, no filter is applied, and the IPv4 is prefered for all peers.";

}
}
}
// peers-augment

grouping policy-container {
  description "LDP policy attributes.";
  container label-policy {
    description "Label policy attributes.";
    container advertise {
      description "Label advertising policies.";
      container egress-explicit-null {
        description "Enables an egress router to advertise an explicit null label (value 0) in place of an implicit null label (value 3) to the penultimate hop router.";
        leaf enable {
          type boolean;
          default false;
          description "'true' to enable explicit null.";
        }
      }
      leaf prefix-list {
        type prefix-list-ref;
        description "Applies the prefix list to filter outgoing label advertisements. If the value is not specified, no prefix filter is applied.";
      }
    }
    leaf prefix-list {
      type prefix-list-ref;
      description "Applies the prefix list to filter incoming label advertisements.
      } // advertise
  container accept {
    description "Label advertisement acceptance policies.";
    leaf prefix-list {
      type prefix-list-ref;
      description "Applies the prefix list to filter incoming label advertisements.";
If the value is not specified, no prefix filter is applied.

} // accept

container assign {
  if-feature policy-label-assignment-config;
  description
    "Label assignment policies";
  container independent-mode {
    description
      "Independent label policy attributes.";
    leaf prefix-list {
      type prefix-list-ref;
      description
        "Assign labels according to certain prefixes. If the value is not specified, no prefix filter is applied (labels are assigned to all learned routes).";
    }
  } // independent-mode

  container ordered-mode {
    if-feature policy-ordered-label-config;
    description
      "Ordered label policy attributes.";
    leaf egress-prefix-list {
      type prefix-list-ref;
      description
        "Assign labels according to certain prefixes for egress LSR.";
    }
  } // ordered-mode

} // assign

} // label-policy

} // policy-container

/*/ 
 * Configuration and state data nodes 
 */

// Forwarding nexthop augmentation to the global tree
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/
  + "ldp:global" {
    description "Graceful forwarding nexthop augmentation."
    uses global-forwarding-nexthop-augment;
  }

// global/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:global/ldp:address-families" {
  description "Global IPv6 augmentation.";
}

container ipv6 {
  presence
    "Present if IPv6 is enabled, unless the 'enable'
    leaf is set to 'false';"
  description
    "Containing data related to the IPv6 address family.";

  leaf enable {
    type boolean;
    default true;
    description
        "'true' to enable the address family."
  }

  uses policy-container;

  leaf transport-address {
    type inet:ipv6-address;
    mandatory true;
    description
        "The transport address advertised in LDP Hello messages.";
  }

  leaf label-distribution-controlmode {
    type enumeration {
      enum independent {
        description
            "Independent label distribution control.";
      }
      enum ordered {
        description
            "Ordered label distribution control.";
      }
    }
    config false;
    description
        "Label distribution control mode.";
    reference
        "RFC5036: LDP Specification. Sec 2.6.";
  }

  // ipv6 bindings
  container bindings {
    config false;
    description
"LDP address and label binding information.");
list address {
  key "address";
  description
  "List of address bindings learned by LDP.";
  leaf address {
    type inet:ipv6-address;
    description
    "The IPv6 address learned from an Address message received from or advertised to a peer.";
  }
  uses ldp:binding-address-state-attributes;
} // binding-address

list fec-label {
  key "fec";
  description
  "List of FEC-label bindings learned by LDP.";
  leaf fec {
    type inet:ipv6-prefix;
    description
    "The prefix FEC value in the FEC-label binding, learned in a Label Mapping message received from or advertised to a peer.";
  }
  uses ldp:binding-label-state-attributes;
} // fec-label
} // bindings
} // ipv6

// discovery/interfaces/interface/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/
+ "ldp:discovery/ldp:interfaces/ldp:interface/
+ "ldp:address-families" {
  description "Interface IPv6 augmentation.";

  container ipv6 {
    presence
    "Present if IPv6 is enabled, unless the ‘enable’ leaf is set to ‘false’";
    description
    "IPv6 address family.";

    leaf enable {
      type boolean;
      default true;
      description
      "IPv6 address family.";
    }
  }
}
"Enable the address family on the interface."
}

// ipv6
container hello-adjacencies {
  config false;
  description
  "Containing a list of hello adjacencies."

  list hello-adjacency {
    key "adjacent-address";
    config false;
    description "List of hello adjacencies."

    leaf adjacent-address {
      type inet:ipv6-address;
      description
      "Neighbor address of the hello adjacency."
    }

    uses ldp:adjacency-state-attributes;
    uses ldp:ldp-peer-ref-container;
  } // hello-adjacency
} // hello-adjacencies
} // ipv6

// discovery/targeted/address-families/ipv6
  description "Targeted discovery IPv6 augmentation.";

  container ipv6 {
    presence
    "Present if IPv6 is enabled.";
    description
    "IPv6 address family.";

    container hello-adjacencies {
      config false;
      description
      "Containing a list of hello adjacencies."

      list hello-adjacency {
        key "local-address adjacent-address";
        config false;
        description "List of hello adjacencies."
      }
    }
  }
}
leaf local-address {
    type inet:ipv6-address;
    description
        "Local address of the hello adjacency.";
}
leaf adjacent-address {
    type inet:ipv6-address;
    description
        "Neighbor address of the hello adjacency.";
}

uses ldp:adjacency-state-attributes;
uses ldp:ldp-peer-ref-container;
} // hello-adjacency
} // hello-adjacencies

list target {
    key "adjacent-address";
    description
        "Targeted discovery params.";
    leaf adjacent-address {
        type inet:ipv6-address;
        description
            "Configures a remote LDP neighbor and enables extended LDP discovery of the specified neighbor.";
    }
    leaf enable {
        type boolean;
        default true;
        description
            "Enable the target.";
    }
    leaf local-address {
        type inet:ipv6-address;
        description
            "The local address used as the source address to send targeted Hello messages.
            If the value is not specified, the transport-address is used as the source address.";
    }
} // target
} // ipv6

// /peers/peer/state/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:peers/ldp:peer/ldp:address-families" {
  description "Peer state IPv6 augmentation.";
}

container ipv6 {
  presence
    "Present if IPv6 is enabled.";
  description
    "IPv6 address family.";
}

container hello-adjacencies {
  config false;
  description
    "Containing a list of hello adjacencies.";
}

list hello-adjacency {
  key "local-address adjacent-address";
  description "List of hello adjacencies.";
}

  leaf local-address {
    type inet:ipv6-address;
    description "Local address of the hello adjacency.";
  }

  leaf adjacent-address {
    type inet:ipv6-address;
    description "Neighbor address of the hello adjacency.";
  }

uses ldp:adjacency-state-attributes;

leaf interface {
  type if:interface-ref;
  description "Interface for this adjacency.";
}

} // hello-adjacency
} // hello-adjacencies
} // ipv6

/*
 * Configuration data and operational state data nodes
 */
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:global" {
  description "Graceful restart augmentation.";
  uses global-augment;
}
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/" 
  + "ldp:global/ldp:capability" {
    description "Capability augmentation.";
    uses capability-augment;
  }

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/" 
  + "ldp:global/ldp:graceful-restart" {
    description "Graceful restart augmentation.";
    uses graceful-restart-augment;
  }

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/" 
  + "ldp:global/ldp:address-families/ldp:ipv4" {
    description "Address family IPv4 augmentation.";
    uses address-family-ipv4-augment;
  }

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/" 
  + "ldp:discovery/ldp:interfaces/ldp:interface" {
    description "Interface augmentation.";
    uses interface-augment;
  }

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/" 
  + "ldp:discovery/ldp:interfaces/ldp:interface/" 
  + "ldp:address-families/ldp:ipv4" {
    description "Interface address family IPv4 augmentation.";
    uses interface-address-family-ipv4-augment;
  }

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/" 
  + "ldp:discovery/ldp:interfaces/ldp:interface/" 
  + "ldp:address-families/ldp-ext:ipv6" {
    description "Interface address family IPv6 augmentation.";
    uses interface-address-family-ipv6-augment;
  }

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/" 
  + "ldp:discovery/ldp:targeted/ldp:hello-accept" {
    description "Targeted discovery augmentation.";
    leaf neighbor-list {
      if-feature policy-targeted-discovery-config;
      type neighbor-list-ref;
      description
      "The name of a neighbor ACL, to accept Hello messages from
      LDP peers as permitted by the neighbor-list policy.
      If this value is not specified, targeted Hello messages from
      
      Raza, et al.            Expires September 6, 2018              
      [Page 83]"
any source are accepted.
};
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/
+ "ldp:peers" {
  description "Peers augmentation."
  uses peers-augment;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/
+ "ldp:peers/ldp:authentication/ldp:auth-type-selection" {
  if-feature key-chain;
  description "Peers authentication augmentation."
  case auth-key-chain {
    uses authentication-keychain-augment;
  }
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/
+ "ldp:peers/ldp:peer" {
  description "Peer list entry augmentation."
  uses peer-augment;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/
+ "ldp:peers/ldp:peer/ldp:authentication/"
+ "ldp:auth-type-selection" {
  if-feature key-chain;
  description "Peer list entry authentication augmentation."
  case auth-key-chain {
    uses authentication-keychain-augment;
  }
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/
+ "ldp:peers/ldp:peer/ldp:address-families/ldp:ipv4" {
  description
    "Peer list entry IPv4 augmentation."
  uses peer-af-policy-container;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/
+ "ldp:peers/ldp:peer/ldp:address-families/ldp-ext:ipv6" {
  description
    "Peer list entry IPv6 augmentation."
    uses peer-af-policy-container;
}
10. Security Considerations

The configuration, state, action and notification data defined using YANG data models in this document are likely to be accessed via the protocols such as NETCONF [RFC6241] etc.

Hence, YANG implementations MUST comply with the security requirements specified in section 15 of [RFC6020]. Additionally, NETCONF implementations MUST comply with the security requirements specified in sections 2.2, 2.3 and 9 of [RFC6241] as well as section 3.7 of [RFC6536].

11. IANA Considerations

This document does not extend LDP base protocol specification and hence there are no IANA considerations.

Note to the RFC Editor: Please remove IANA section before the publication.

12. Acknowledgments

The authors would like to acknowledge Eddie Chami, Nagendra Kumar, Mannan Venkatesan, Pavan Beeram for their contribution to this document. We also acknowledge Ladislav Lhotka for his useful comments as the YANG Doctor.

13. References

13.1. Normative References

[I-D.ietf-netmod-revised-datastores]


13.2. Informative References


Appendix A.  Additional Contributors

Reshad Rahman
Cisco Systems Inc.
Email: rrahman@cisco.com

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

Authors’ Addresses

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

Rajiv Asati
Cisco Systems, Inc.
USA
Email: rajiva@cisco.com

Danial Johari
Cisco Systems, Inc.
CA
Email: dajohari@cisco.com

Xufeng Liu
Jabil
USA
Email: xufeng_liu@jabil.com

Jeff Tantsura
Email: jefftant.ietf@gmail.com

Santosh Esale
Juniper Networks
USA
Email: sesale@juniper.net
Xia Chen
Huawei Technologies
CN
Email: jescia.chenxia@huawei.com

Loa Andersson
Huawei Technologies
Email: loa@pi.nu

Himanshu Shah
Ciena Corporation
USA
Email: hshah@ciena.com

Matthew Bocci
Nokia
Email: matthew.bocci@nokia.com
Label Switched Path (LSP) Ping/Trace Multipath Support for Link Aggregation Group (LAG) Interfaces
draft-ietf-mpls-lsp-ping-lag-multipath-04

Abstract

This document defines an extension to the MPLS Label Switched Path (LSP) Ping and Traceroute as specified in RFC 8029. The extension allows the MPLS LSP Ping and Traceroute to discover and exercise specific paths of Layer 2 (L2) Equal-Cost Multipath (ECMP) over Link Aggregation Group (LAG) interfaces.

This document updates RFC8029.

Requirements Language

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

Status of This Memo

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

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

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

Copyright Notice

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

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

Table of Contents

1. Introduction ......................................................... 3
   1.1. Terminology ................................................. 3
   1.2. Background ................................................. 3
2. Overview ............................................................ 4
3. LSR Capability Discovery .......................................... 6
4. Mechanism to Discover L2 ECMP Multipath .......................... 7
   4.1. Initiator LSR Procedures ................................... 7
   4.2. Responder LSR Procedures ................................... 7
   4.3. Additional Initiator LSR Procedures ......................... 9
5. Mechanism to Validate L2 ECMP Traversal .......................... 10
   5.1. Incoming LAG Member Links Verification ....................... 11
      5.1.1. Initiator LSR Procedures ............................. 11
      5.1.2. Responder LSR Procedures ............................. 11
      5.1.3. Additional Initiator LSR Procedures ................. 12
   5.2. Individual End-to-End Path Verification ...................... 13
6. LSR Capability TLV ................................................. 14
7. LAG Description Indicator Flag: G ................................ 15
8. Local Interface Index Sub-TLV ..................................... 16
9. Remote Interface Index Sub-TLV .................................... 16
10. Detailed Interface and Label Stack TLV ........................... 17
   10.1. Sub-TLVs .................................................. 19
      10.1.1. Incoming Label Stack Sub-TLV ....................... 19
      10.1.2. Incoming Interface Index Sub-TLV ................... 19
11. Security Considerations ........................................... 20
12. IANA Considerations .............................................. 21
   12.1. LSR Capability TLV ....................................... 21
      12.1.1. LSR Capability Flags .............................. 21
   12.2. Local Interface Index Sub-TLV ............................. 21
      12.2.1. Interface Index Flags .............................. 22
1. Introduction

1.1. Terminology

The following acronyms/terms are used in this document:

- MPLS - Multiprotocol Label Switching.
- LSP - Label Switched Path.
- LSR - Label Switching Router.
- ECMP - Equal-Cost Multipath.
- LAG - Link Aggregation Group.
- Initiator LSR - LSR which sends MPLS echo request.
- Responder LSR - LSR which receives MPLS echo request and sends MPLS echo reply.

1.2. Background

The MPLS Label Switched Path (LSP) Ping and Traceroute as specified in [RFC8029] are powerful tools designed to diagnose all available layer 3 (L3) paths of LSPs, i.e., provides diagnostic coverage of L3 Equal-Cost Multipath (ECMP). In many MPLS networks, Link Aggregation Group (LAG) as defined in [IEEE802.1AX], which provide Layer 2 (L2) ECMP, are often used for various reasons. MPLS LSP Ping and Traceroute tools were not designed to discover and exercise specific paths of L2 ECMP. The result raises a limitation for following scenario when LSP X traverses over LAG Y:
Label switching of LSP X over one or more member links of LAG Y have succeeded.

Label switching of LSP X over one or more member links of LAG Y have failed.

MPLS echo request for LSP X over LAG Y is load balanced over a member link which is label switching successfully.

With the above scenario, MPLS LSP Ping and Traceroute will not be able to detect the label switching failure of problematic member link(s) of the LAG. In other words, lack of L2 ECMP diagnostic coverage can produce an outcome where MPLS LSP Ping and Traceroute can be blind to label switching failures over problematic LAG interface. It is, thus, desirable to extend the MPLS LSP Ping and Traceroute to have deterministic diagnostic coverage of LAG interfaces.

Creation of this document was motivated by issues encountered in live networks.

2. Overview

This document defines an extension to the MPLS LSP Ping and Traceroute to describe Multipath Information for LAG member links separately, thus allowing MPLS LSP Ping and Traceroute to discover and exercise specific paths of L2 ECMP over LAG interfaces. Reader is expected to be familiar with mechanics of Downstream Mapping described in Section 3.3 of [RFC8029] and Downstream Detailed Mapping TLV (DDMAP) described in Section 3.4 of [RFC8029].

MPLS echo request carries a DDMAP and an optional TLV to indicate that separate load balancing information for each L2 nexthop over LAG is desired in MPLS echo reply. Responder LSR places the same optional TLV in the MPLS echo reply to provide acknowledgement back to the initiator. It also adds, for each downstream LAG member, a load balance information (i.e. multipath information and interface index). The following figure and the texts provides an example using an LDP network. However the problem and the mechanism is applicable to all types of LSPs which can traverse over LAG interfaces.
When node A is initiating LSP Traceroute to node E, node B will return to node A load balance information for following entries.

1. Downstream C over Non-LAG (upper path).
2. First Downstream C over LAG (middle path).
3. Second Downstream C over LAG (middle path).
4. Downstream D over Non-LAG (lower path).

This document defines:

- In Section 3, a mechanism discover capabilities of responder LSRs;
- In Section 4, a mechanism to discover L2 ECMP multipath information;
- In Section 5, a mechanism to validate L2 ECMP traversal in some LAG provisioning models;
- In Section 6, the LSR Capability TLV;
- In Section 7, the LAG Description Indicator flag;
- In Section 8, the Local Interface Index Sub-TLV;
- In Section 9, the Remote Interface Index Sub-TLV;
- In Section 10, the Detailed Interface and Label Stack TLV;
- In Appendix A, issues with LAG having an L2 Switch.

Note that the mechanism described in this document does not impose any changes to scenarios where an LSP is pinned down to a particular
LAG member (i.e. the LAG is not treated as one logical interface by
the LSP).

Also note that many LAGs are built from p2p links, and thus router X
and router X+1 have the same number of LAG members. It is possible
to build LAGs asymmetrically by using Ethernet switches in the
middle. Appendix A lists some cases which this document does not
address; if an operator deploys LAGs in a manner similar to what’s
shown in Appendix A, the mechanisms in this document may not suit
them.

3. LSR Capability Discovery

The MPLS Ping operates by an initiator LSR sending an MPLS echo
request message and receiving back a corresponding MPLS echo reply
message from a responder LSR. The MPLS Traceroute operates in a
similar way except the initiator LSR potentially sends multiple MPLS
echo request messages with incrementing TTL values.

There has been many extensions to the MPLS Ping and Traceroute
mechanism over the years. Thus it is often useful, and sometimes
necessary, for the initiator LSR to deterministically disambiguate
the difference between:

- The responder LSR sent the MPLS echo reply message with contents C
  because it has feature X, Y and Z implemented.
- The responder LSR sent the MPLS echo reply message with contents C
  because it has subset of features X, Y and Z implemented but not
  all.
- The responder LSR sent the MPLS echo reply message with contents C
  because it does not have features X, Y and Z implemented.

To allow the initiator LSR to disambiguate the above differences,
this document defines the LSR Capability TLV (described in
Section 6). When the initiator LSR wishes to discover the
capabilities of the responder LSR, the initiator LSR includes the LSR
Capability TLV in the MPLS echo request message. When the responder
LSR receives an MPLS echo request message with the LSR Capability TLV
included, then the responder LSR MUST include the LSR Capability TLV
in the MPLS echo reply message with the LSR Capability TLV describing
features and extensions supported by the local LSR.

It is RECOMMENDED that implementations supporting the LAG Multipath
extensions defined in this document include the LSR Capability TLV in
MPLS echo request messages.
4.  Mechanism to Discover L2 ECMP Multipath

4.1.  Initiator LSR Procedures

The MPLS echo request carries a DDMAP with the "LAG Description Indicator flag" (G) set in the DS Flags to indicate that separate load balancing information for each L2 nexthop over LAG is desired in MPLS echo reply. The new "LAG Description Indicator flag" is described in Section 7.

4.2.  Responder LSR Procedures

This section describes the handling of the new TLVs by nodes which understand the "LAG Description Indicator flag". There are two cases - nodes which understand the "LAG Description Indicator flag" but which for some reason cannot describe LAG members separately, and nodes which both understand the "LAG Description Indicator flag" and are able to describe LAG members separately. Note that Section 6, Section 8 and Section 9 describe the new TLVs referenced by this section, and looking over the definition of the new TLVs first may make it easier to read this section.

A responder LSR that understand the "LAG Description Indicator flag" but is not capable of describing outgoing LAG member links separately uses the following procedures:

- If the received MPLS echo request message had the LSR Capability TLV, the responder LSR MUST include the LSR Capability TLV in the MPLS echo reply message.
- The responder LSR MUST clear the "Downstream LAG Info Accommodation flag" in the LSR Capability Flags field of the LSR Capability TLV. This will allow the initiator LSR to understand that the responder LSR cannot describe outgoing LAG member links separately in the DDMAP.

A responder LSR that understands the "LAG Description Indicator flag" and is capable of describing outgoing LAG member links separately uses the follow procedures, regardless of whether or not outgoing interfaces include LAG interfaces:

- If the received MPLS echo request message had the LSR Capability TLV, the responder LSR MUST include the LSR Capability TLV in the MPLS echo reply message.
- The responder LSR MUST set the "Downstream LAG Info Accommodation flag" in the LSR Capability Flags field of the LSR Capability TLV.
For each downstream that is a LAG interface:

* The responder LSR MUST add DDMAP in the MPLS echo reply.

* The responder LSR MUST set the "LAG Description Indicator flag" in the DS Flags field of the DDMAP.

* In the DDMAP, Local Interface Index Sub-TLV, Remote Interface Index Sub-TLV and Multipath Data Sub-TLV are to describe each LAG member link. All other fields of the DDMAP are to describe the LAG interface.

* For each LAG member link of this LAG interface:

  + The responder LSR MUST add a Local Interface Index Sub-TLV (described in Section 8) with the "LAG Member Link Indicator flag" set in the Interface Index Flags field, describing the interface index of this outgoing LAG member link (the local interface index is assigned by the local LSR).

  + The responder LSR MAY add a Remote Interface Index Sub-TLV (described in Section 9) with the "LAG Member Link Indicator flag" set in the Interface Index Flags field, describing the interface index of the incoming LAG member link on the downstream LSR (this interface index is assigned by the downstream LSR). How the local LSR obtains the interface index of the LAG member link on the downstream LSR is outside the scope of this document.

  + The responder LSR MUST add an Multipath Data Sub-TLV for this LAG member link, if received DDMAP requested multipath information.

Based on the procedures described above, every LAG member link will have a Local Interface Index Sub-TLV and a Multipath Data Sub-TLV entries in the DDMAP. The order of the Sub-TLVs in the DDMAP for a LAG member link MUST be Local Interface Index Sub-TLV immediately followed by Multipath Data Sub-TLV. A LAG member link may also have a corresponding Remote Interface Index Sub-TLV. When a Local Interface Index Sub-TLV, a Remote Interface Index-Sub-TLV and a Multipath Data Sub-TLV are placed in the DDMAP to describe a LAG member link, they MUST be placed in the order of Local Interface Index Sub-TLV, Remote Interface Index-Sub-TLV and Multipath Data Sub-TLV.

A responder LSR possessing a LAG interface with two member links would send the following DDMAP for this LAG interface:

When none of the received multipath information maps to a particular LAG member link, then the responder LSR MUST still place the Local Interface Index Sub-TLV and the Multipath Data Sub-TLV for that LAG member link in the DDMAP, with the Multipath Length field of the Multipath Data Sub-TLV being zero.

4.3. Additional Initiator LSR Procedures

The procedures above allow an initiator LSR to:

- Identify whether or not the responder LSR can describe outgoing LAG member links separately, by looking at the LSR Capability TLV.

- Utilize the value of the "LAG Description Indicator flag" in DS Flags to identify whether each received DDMAP describes a LAG interface or a non-LAG interface.

- Obtain multipath information which is expected to traverse the specific LAG member link described by corresponding interface index.

When an initiator LSR receives a DDMAP containing LAG member information from a downstream LSR with TTL=n, then the subsequent DDMAP sent by the initiator LSR to the downstream LSR with TTL=n+1 through a particular LAG member link MUST be updated with following procedures:
The Local Interface Index Sub-TLVs MUST be removed in the sending DDMAP.

If the Remote Interface Index Sub-TLVs were present and the initiator LSR is traversing over a specific LAG member link, then the Remote Interface Index Sub-TLV corresponding to the LAG member link being traversed SHOULD be included in the sending DDMAP. All other Remote Interface Index Sub-TLVs MUST be removed from the sending DDMAP.

The Multipath Data Sub-TLVs MUST be updated to include just one Multipath Data Sub-TLV. The initiator MAY keep just the Multipath Data Sub-TLV corresponding to the LAG member link being traversed, or combine the Multipath Data Sub-TLVs for all LAG member links into a single Multipath Data Sub-TLV when diagnosing further downstream LSRs.

All other fields of the DDMAP are to comply with procedures described in [RFC8029].

Using the DDMAP example described in the Figure 2, the DDMAP being sent by the initiator LSR through LAG member link #1 to the next downstream LSR should be:

```
+-----------------------------------------------+
| DDMAP fields describing LAG interface with DS Flags G set |
| [OPTIONAL] Remote Interface Index Sub-TLV of LAG member link #1 |
| Multipath Data Sub-TLV LAG member link #1 |
| Label Stack Sub-TLV |
```

Figure 3: Example of DDMAP in MPLS Echo Request

5. Mechanism to Validate L2 ECMP Traversal

Section 4 defines the responder LSR procedures to constructs a DDMAP for a downstream LAG, and also defines that inclusion of the Remote Interface Index Sub-TLVs describing the incoming LAG member links of the downstream LSR is optional. The reason why it is optional for the responder LSR to include the Remote Interface Index Sub-TLVs is that this information from the downstream LSR is often not available on the responder LSR. In such case, the traversal of LAG member links can be validated with procedures described in Section 5.1. If
LSRs can provide the Remote Interface Index Sub-TLVs in DDMAP objects, then the validation procedures described in Section 5.2 can be used.

5.1. Incoming LAG Member Links Verification

Without downstream LSRs returning remote Interface Index Sub-TLVs in the DDMAP, validation of the LAG member link traversal requires that initiator LSR traverses all available LAG member links and taking the results through a logic. This section provides the mechanism for the initiator LSR to obtain additional information from the downstream LSRs and describes the additional logic in the initiator LSR to validate the L2 ECMP traversal.

5.1.1. Initiator LSR Procedures

The MPLS echo request is sent with a DDMAP with the "Interface and Label Stack Object Request flag" and "LAG Description Indicator flag" set in the DS Flags to indicate the request for Detailed Interface and Label Stack TLV with additional LAG member link information (i.e. interface index) in the MPLS echo reply.

5.1.2. Responder LSR Procedures

A responder LSR that understands the "LAG Description Indicator flag" but is not capable of describing incoming LAG member link is to use following procedures:

- If the received MPLS echo request message had the LSR Capability TLV, the responder LSR MUST include the LSR Capability TLV in the MPLS echo reply message.
- The responder LSR MUST clear the "Upstream LAG Info Accommodation flag" in the LSR Capability Flags field of the LSR Capability TLV. This will allow the initiator LSR to understand that the responder LSR cannot describe incoming LAG member link.

A responder LSR that understands the "LAG Description Indicator flag" and is capable of describing incoming LAG member link MUST use the following procedures, regardless of whether or not incoming interface was a LAG interface:

- If the received MPLS echo request message had the LSR Capability TLV, the responder LSR MUST include the LSR Capability TLV in the MPLS echo reply message.
- The responder LSR MUST set the "Upstream LAG Info Accommodation flag" in the LSR Capability Flags field of the LSR Capability TLV.
When the received DDMAP had "Interface and Label Stack Object Request flag" set in the DS Flags field, the responder LSR MUST add the Detailed Interface and Label Stack TLV (described in Section 10) in the MPLS echo reply.

When the received DDMAP had "Interface and Label Stack Object Request flag" set in the DS Flags field and the incoming interface was a LAG, the responder LSR MUST add the Incoming Interface Index Sub-TLV (described in Section 10.1.2) in the Detailed Interface and Label Stack TLV. The "LAG Member Link Indicator flag" MUST be set in the Interface Index Flags field, and the Interface Index field set to the LAG member link which received the MPLS echo request.

These procedures allow initiator LSR to:

- Identify whether or not the responder LSR can describe the incoming LAG member link, by looking at the LSR Capability TLV.
- Utilize the Incoming Interface Index Sub-TLV in the Detailed Interface and Label Stack TLV to identify, if the incoming interface was a LAG, the identity of the incoming LAG member.

5.1.3. Additional Initiator LSR Procedures

Along with procedures described in Section 4, the procedures described in this section will allow an initiator LSR to know:

- The expected load balance information of every LAG member link, at LSR with TTL=n.
- With specific entropy, the expected interface index of the outgoing LAG member link at TTL=n.
- With specific entropy, the interface index of the incoming LAG member link at TTL=n+1.

Expectation is that there’s a relationship between the interface index of the outgoing LAG member link at TTL=n and the interface index of the incoming LAG member link at TTL=n+1 for all discovered entropies. In other words, set of entropies that load balances to outgoing LAG member link X at TTL=n should all reach the nexthop on same incoming LAG member link Y at TTL=n+1.

With additional logics, the initiator LSR can perform following checks in a scenario where the initiator knows that there is a LAG, with two LAG members, between TTL=n and TTL=n+1, and has the multipath information to traverse the two LAG members.
The initiator LSR sends two MPLS echo request messages to traverse
the two LAG members at TTL=n+1:

- Success case:
  - One MPLS echo request message reaches TTL=n+1 on an LAG member 1.
  - The other MPLS echo request message reaches TTL=n+1 on an LAG member 2.

The two MPLS echo request messages sent by the initiator LSR reach
two different LAG members at the immediate downstream LSR.

- Error case:
  - One MPLS echo request message reaches TTL=n+1 on an LAG member 1.
  - The other MPLS echo request message also reaches TTL=n+1 on an LAG member 1.

One or two MPLS echo request messages sent by the initiator LSR
does not reach the immediate downstream LSR, or the two MPLS echo
request messages reach a same LAG member at the immediate
downstream LSR.

Note that defined procedures will provide a deterministic result for
LAG interfaces that are back-to-back connected between routers (i.e.
no L2 switch in between). If there is a L2 switch between LSR at
TTL=n and LSR at TTL=n+1, there is no guarantee that traversal of
every LAG member link at TTL=n will result in reaching different
interface index at TTL=n+1. Issues resulting from LAG with L2 switch
in between are further described in Appendix A. LAG provisioning
models in operated network should be considered when analyzing the
output of LSP Traceroute exercising L2 ECMPs.

5.2. Individual End-to-End Path Verification

When the Remote Interface Index Sub-TLVs are available from an LSR
with TTL=n, then the validation of LAG member link traversal can be
performed by the downstream LSR of TTL=n+1. The initiator LSR
follows the procedures described in Section 4.3.

The DDMAP validation procedures by the downstream responder LSR are
then updated to include the comparison of the incoming LAG member
link (which MPLS echo request was received on) to the interface index
described in the Remote Interface Index Sub-TLV in the DDMAP.
Failure of this comparison results in the return code being set to "Downstream Mapping Mismatch (5)".

A responder LSR that is not able to perform the above additional DDMAP validation procedures is considered to lack the upstream LAG capability. Thus, if the received MPLS echo request contained the LSR Capability TLV, then the responder LSR MUST include the LSR Capability TLV in the MPLS echo reply and the LSR Capability TLV MUST have the "Upstream LAG Info Accomodation flag" cleared.

6. LSR Capability TLV

The LSR Capability object is a new TLV that MAY be included in the MPLS echo request message and the MPLS echo reply message. An MPLS echo request message and an MPLS echo reply message MUST NOT include more than one LSR Capability object. Presence of an LSR Capability object in an MPLS echo request message is a request that a responder LSR includes an LSR Capability object in the MPLS echo reply message, with the LSR Capability object describing features and extensions supported. When the received MPLS echo request message contains an LSR Capability object, an responder LSR MUST include the LSR Capability object in the MPLS echo reply message.

LSR Capability TLV Type is TBD1. Length is 4. The value field of the LSR Capability TLV has following format:

```
+-------+-------+-------+-------+
|  Type  |  Length| LSR Capability Flags |
+-------+-------+-----------------------+
```

Figure 4: LSR Capability TLV

LSR Capability Flags

The LSR Capability Flags field is a bit vector with following format:

```
+-------+-------+-------+-------+
|  Must Be Zero (Reserved)  | [U|D] |
+-------+-------+-----------------------+
```

Akiya, et al. Expires December 6, 2018
Two flags are defined: U and D. The remaining flags MUST be set to zero when sending and ignored on receipt. Both U and D flags MUST be cleared in MPLS echo request message when sending, and ignored on receipt. Neither, either or both U and D flags MAY be set in MPLS echo reply message.

Flag | Name and Meaning
---- | -----------------
    | Upstream LAG Info Accommodation
    | An LSR sets this flag when the node is capable of describing a LAG member link in the Incoming Interface Index Sub-TLV in the Detailed Interface and Label Stack TLV.
    | Downstream LAG Info Accommodation
    | An LSR sets this flag when the node is capable of describing LAG member links in the Local Interface Index Sub-TLV and the Multipath Data Sub-TLV in the Downstream Detailed Mapping TLV.

7. LAG Description Indicator Flag: G

One flag, G, is added in DS Flags field of the DDMAP TLV. The G flag of the DS Flags field in the MPLS echo request message indicates the request for detailed LAG information from the responder LSR. In the MPLS echo reply message, the G flag MUST be set if the DDMAP TLV describes a LAG interface. It MUST be cleared otherwise.

DS Flags

DS Flags G is added, in Bit Number TBD5, in DS Flags bit vector.

  0 1 2 3 4 5 6 7
  +---+---+---+---+---+---+---+---+---+
  | MBZ |G|MBZ|I|N|
  +---+---+---+---+---+

RFC-Editor-Note: Please update above figure to place the flag G in the bit number TBD5.
### Flag Name and Meaning

---

**G** LAG Description Indicator

When this flag is set in the MPLS echo request, responder is requested to respond with detailed LAG information. When this flag is set in the MPLS echo reply, the corresponding DDMAP describes a LAG interface.

#### 8. Local Interface Index Sub-TLV

The Local Interface Index object is a Sub-TLV that MAY be included in a DDMAP TLV. Zero or more Local Interface Index object MAY appear in a DDMAP TLV. The Local Interface Index Sub-TLV describes the index assigned by the local LSR to the egress interface.

The Local Interface Index Sub-TLV Type is TBD2. Length is 8, and the Value field has following format:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Local Interface Index                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Local Interface Index Sub-TLV

Local Interface Index

An Index assigned by the LSR to this interface.

#### 9. Remote Interface Index Sub-TLV

The Remote Interface Index object is a Sub-TLV that MAY be included in a DDMAP TLV. Zero or more Remote Interface Index object MAY appear in a DDMAP TLV. The Remote Interface Index Sub-TLV describes the index assigned by the downstream LSR to the ingress interface.

The Remote Interface Index Sub-TLV Type is TBD3. Length is 8, and the Value field has following format:
10. Detailed Interface and Label Stack TLV

The "Detailed Interface and Label Stack" object is a TLV that MAY be included in a MPLS echo reply message to report the interface on which the MPLS echo request message was received and the label stack that was on the packet when it was received. A responder LSR MUST NOT insert more than one instance of this TLV. This TLV allows the initiator LSR to obtain the exact interface and label stack information as it appears at the responder LSR.

Detailed Interface and Label Stack TLV Type is TBD4. Length is K + Sub-TLV Length (sum of Sub-TLVs). K is the sum of all fields of this TLV prior to Sub-TLVs, but the length of K depends on the Address Type. Details of this information is described below. The Value field has following format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Address Type  |             Must Be Zero                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   IP Address (4 or 16 octets)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Interface (4 or 16 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                                 |
|                       List of Sub-TLVs                        |
|                                                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Detailed Interface and Label Stack TLV

Remote Interface Index

An Index assigned by the downstream LSR to the ingress interface.
The Detailed Interface and Label Stack TLV format is derived from the Interface and Label Stack TLV format (from [RFC8029]). Two changes are introduced. First is that label stack, which is of variable length, is converted into a sub-TLV. Second is that a new sub-TLV is added to describe an interface index. The fields of Detailed Interface and Label Stack TLV have the same use and meaning as in [RFC8029]. A summary of the fields taken from the Interface and Label Stack TLV is as below:

**Address Type**

The Address Type indicates if the interface is numbered or unnumbered. It also determines the length of the IP Address and Interface fields. The resulting total for the initial part of the TLV is listed in the table below as "K Octets". The Address Type is set to one of the following values:

<table>
<thead>
<tr>
<th>Type #</th>
<th>Address Type</th>
<th>K Octets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IPv4 Numbered</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>IPv4 Unnumbered</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>IPv6 Numbered</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>IPv6 Unnumbered</td>
<td>28</td>
</tr>
</tbody>
</table>

**IP Address and Interface**

IPv4 addresses and interface indices are encoded in 4 octets; IPv6 addresses are encoded in 16 octets.

If the interface upon which the echo request message was received is numbered, then the Address Type MUST be set to IPv4 Numbered or IPv6 Numbered, the IP Address MUST be set to either the LSR’s Router ID or the interface address, and the Interface MUST be set to the interface address.

If the interface is unnumbered, the Address Type MUST be either IPv4 Unnumbered or IPv6 Unnumbered, the IP Address MUST be the LSR’s Router ID, and the Interface MUST be set to the index assigned to the interface.

Note: Usage of IPv6 Unnumbered has the same issue as [RFC8029], described in Section 3.4.2 of [RFC7439]. A solution should be considered an applied to both [RFC8029] and this document.
10.1. Sub-TLVs

This section defines the sub-TLVs that MAY be included as part of the Detailed Interface and Label Stack TLV.

<table>
<thead>
<tr>
<th>Sub-Type</th>
<th>Value Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Incoming Label stack</td>
</tr>
<tr>
<td>2</td>
<td>Incoming Interface Index</td>
</tr>
</tbody>
</table>

10.1.1. Incoming Label Stack Sub-TLV

The Incoming Label Stack sub-TLV contains the label stack as received by the LSR. If any TTL values have been changed by this LSR, they SHOULD be restored.

Incoming Label Stack Sub-TLV Type is 1. Length is variable, and the Value field has following format:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 8: Incoming Label Stack Sub-TLV

10.1.2. Incoming Interface Index Sub-TLV

The Incoming Interface Index object is a Sub-TLV that MAY be included in a Detailed Interface and Label Stack TLV. The Incoming Interface Index Sub-TLV describes the index assigned by this LSR to the interface which received the MPLS echo request message.

Incoming Interface Index Sub-TLV Type is 2. Length is 8, and the Value field has the same format as the Local Interface Index Sub-TLV described in Section 8, and has following format:
Interface Index Flags

Interface Index Flags field is a bit vector with following format.

<table>
<thead>
<tr>
<th>Must Be Zero (Reserved)</th>
<th>M</th>
</tr>
</thead>
</table>

One flag is defined: M. The remaining flags MUST be set to zero when sending and ignored on receipt.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Name and Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>LAG Member Link Indicator</td>
</tr>
</tbody>
</table>

When this flag is set, interface index described in this sub-TLV is a member of a LAG.

Incoming Interface Index

An Index assigned by the LSR to this interface.

11. Security Considerations

This document extends LSP Traceroute mechanism to discover and exercise L2 ECMP paths. As a result of supporting the code points and procedures described in this document, additional processing are required by initiator LSRs and responder LSRs, especially to compute and handle increasing number of multipath information. Due to additional processing, it is critical that proper security measures described in [RFC8029] are followed.
The LSP Traceroute allows an initiator LSR to discover the paths of tested LSPs, providing deep knowledge of the MPLS network. Exposing such information to a malicious user is considered dangerous. To prevent leakage of vital information to untrusted users, a responder LSR MUST only accept MPLS echo request messages from trusted sources via filtering source IP address field of received MPLS echo request messages.

12. IANA Considerations

12.1. LSR Capability TLV

The IANA is requested to assign new value TBD1 for LSR Capability TLV from the "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD1</td>
<td>LSR Capability TLV</td>
<td>this document</td>
</tr>
</tbody>
</table>

12.1.1. LSR Capability Flags

The IANA is requested to create and maintain a registry entitled "LSR Capability Flags" with following registration procedures:

Registry Name: LAG Interface Info Flags

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>D: Downstream LAG Info Accommodation</td>
<td>this document</td>
</tr>
<tr>
<td>30</td>
<td>U: Upstream LAG Info Accommodation</td>
<td>this document</td>
</tr>
<tr>
<td>0-29</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>

Assignments of LSR Capability Flags are via Standards Action [RFC8126].

12.2. Local Interface Index Sub-TLV

The IANA is requested to assign new value TBD2 (from the range 4-31743) for the Local Interface Index Sub-TLV from the "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry, "Sub-TLVs for TLV Types 20" sub-registry.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD2</td>
<td>Local Interface Index Sub-TLV</td>
<td>this document</td>
</tr>
</tbody>
</table>
12.2.1. Interface Index Flags

The IANA is requested to create and maintain a registry entitled "Interface Index Flags" with following registration procedures:

Registry Name: Interface Index Flags

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>M: LAG Member Link Indicator</td>
<td>this document</td>
</tr>
<tr>
<td>0-14</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>

Assignments of Interface Index Flags are via Standards Action [RFC8126].

Note that this registry is used by the Interface Index Flags field of following Sub-TLVs:

- The Local Interface Index Sub-TLV which may be present in the "Downstream Detailed Mapping" TLV.
- The Remote Interface Index Sub-TLV which may be present in the "Downstream Detailed Mapping" TLV.
- The Incoming Interface Index Sub-TLV which may be present in the "Detailed Interface and Label Stack" TLV.

12.3. Remote Interface Index Sub-TLV

The IANA is requested to assign new value TBD3 (from the range 32768-49161) for the Remote Interface Index Sub-TLV from the "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry, "Sub-TLVs for TLV Types 20" sub-registry.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD3</td>
<td>Remote Interface Index Sub-TLV</td>
<td>this document</td>
</tr>
</tbody>
</table>

12.4. Detailed Interface and Label Stack TLV

The IANA is requested to assign new value TBD4 for Detailed Interface and Label Stack TLV from the "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry ([IANA-MPLS-LSP-PING]).
12.4.1. Sub-TLVs for TLV Type TBD4

The IANA is requested to create and maintain a sub-registry entitled "Sub-TLVs for TLV Type TBD4" under "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry.

Initial values for this sub-registry, "Sub-TLVs for TLV Types TBD4", are described below.

<table>
<thead>
<tr>
<th>Sub-Type</th>
<th>Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Incoming Label Stack</td>
<td>this document</td>
</tr>
<tr>
<td>2</td>
<td>Incoming Interface Index</td>
<td>this document</td>
</tr>
<tr>
<td>3-16383</td>
<td>Unassigned (mandatory TLVs)</td>
<td></td>
</tr>
<tr>
<td>16384-31743</td>
<td>Experimental</td>
<td></td>
</tr>
<tr>
<td>32768-49161</td>
<td>Unassigned (optional TLVs)</td>
<td></td>
</tr>
<tr>
<td>49162-64511</td>
<td>Experimental</td>
<td></td>
</tr>
</tbody>
</table>

Assignments of Sub-Types in the mandatory and optional spaces are via Standards Action [RFC8126]. Assignments of Sub-Types in the experimental space is via Specification Required [RFC8126].

12.5. DS Flags

The IANA is requested to assign a new bit number from the "DS flags" sub-registry from the "Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry ([IANA-MPLS-LSP-PING]).

Note: the "DS flags" sub-registry is created by [RFC8029].

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD5 G</td>
<td>LAG Description Indicator</td>
<td>this document</td>
</tr>
</tbody>
</table>

13. Acknowledgements

The authors would like to thank Nagendra Kumar and Sam Aldrin for providing useful comments and suggestions. The authors would like to thank Loa Andersson for performing a detailed review and providing number of comments.
The authors also would like to extend sincere thanks to the MPLS RT review members who took time to review and provide comments. The members are Eric Osborne, Mach Chen and Yimin Shen. The suggestion by Mach Chen to generalize and create the LSR Capability TLV was tremendously helpful for this document and likely for future documents extending the MPLS LSP Ping and Traceroute mechanism. The suggestion by Yimin Shen to create two separate validation procedures had a big impact to the contents of this document.

14. References

14.1. Normative References


14.2. Informative References


Appendix A. LAG with L2 Switch Issues

Several flavors of "LAG with L2 switch" provisioning models are described in this section, with MPLS data plane ECMP traversal validation issues with each.

A.1. Equal Numbers of LAG Members

R1 ==== S1 ==== R2

The issue with this LAG provisioning model is that packets traversing a LAG member from R1 to S1 can get load balanced by S1 towards R2. Therefore, MPLS echo request messages traversing specific LAG member from R1 to S1 can actually reach R2 via any LAG members, and sender of MPLS echo request messages have no knowledge of this nor no way to control this traversal. In the worst case, MPLS echo request messages with specific entropies to exercise every LAG members from R1 to S1 can all reach R2 via same LAG member. Thus it is impossible for MPLS echo request sender to verify that packets intended to traverse specific LAG member from R1 to S1 did actually traverse that LAG member, and to deterministically exercise "receive" processing of every LAG member on R2.

A.2. Deviating Numbers of LAG Members

R1 ==== S1 ==== R2

There are deviating number of LAG members on the two sides of the L2 switch. The issue with this LAG provisioning model is the same as previous model, sender of MPLS echo request messages have no knowledge of L2 load balance algorithm nor entropy values to control the traversal.

A.3. LAG Only on Right

R1 ---- S1 ==== R2

The issue with this LAG provisioning model is that there is no way for MPLS echo request sender to deterministically exercise both LAG members from S1 to R2. And without such, "receive" processing of R2 on each LAG member cannot be verified.

A.4. LAG Only on Left

R1 ==== S1 ---- R2
MPLS echo request sender has knowledge of how to traverse both LAG members from R1 to S1. However, both types of packets will terminate on the non-LAG interface at R2. It becomes impossible for MPLS echo request sender to know that MPLS echo request messages intended to traverse a specific LAG member from R1 to S1 did indeed traverse that LAG member.

Authors’ Addresses

Nobo Akiya  
Big Switch Networks
Email: nobo.akiya.dev@gmail.com

George Swallow  
Cisco Systems
Email: swallow@cisco.com

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

Bruno Decraene  
Orange
Email: bruno.dehraene@orange.com

John E. Drake  
Juniper Networks
Email: jdrake@juniper.net

Mach (Guoyi) Chen  
Huawei
Email: mach.chen@huawei.com
Resilient MPLS Rings
draft-ietf-mpls-rmr-07

Abstract

This document describes the use of the MPLS control and data planes on ring topologies. It describes the special nature of rings, and proceeds to show how MPLS can be effectively used in such topologies. It describes how MPLS rings are configured, auto-discovered and signaled, as well as how the data plane works. Companion documents describe the details of discovery and signaling for specific protocols.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 5, 2018.

Copyright Notice

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

Rings are a very common topology in transport networks. A ring is the simplest topology offering link and node resilience. Rings are nearly ubiquitous in access and aggregation networks. As MPLS
increases its presence in such networks, and takes on a greater role in transport, it is imperative that MPLS handles rings well; this is not the case today.

This document describes the special nature of rings, and the special needs of MPLS on rings. It then shows how these needs can be met in several ways, some of which involve extensions to protocols such as IS-IS [RFC5305], OSPF [RFC3630], RSVP-TE [RFC3209] and LDP [RFC5036].

The intent of this document is to handle rings that "occur naturally". Many access and aggregation networks in metros have their start as a simple ring. They may then grow into more complex topologies, for example, by adding parallel links to the ring, or by adding "express" links. The goal here is to discover these rings (with some guidance), and run MPLS over them efficiently. The intent is not to construct rings in a mesh network, and use those for protection.

1.1. Definitions

A (directed) graph \( G = (V, E) \) consists of a set of vertices (or nodes) \( V \) and a set of edges (or links) \( E \). An edge is an ordered pair of nodes \((a, b)\), where \( a \) and \( b \) are in \( V \). (In this document, the terms node and link will be used instead of vertex and edge.)

A ring is a subgraph of \( G \). A ring consists of a subset of \( n \) nodes \( \{R_i, 0 \leq i < n\} \) of \( V \). The directed edges \((R_i, R_{i+1}) \) and \((R_{i+1}, R_i)\), \( 0 \leq i < n-1 \) must be a subset of \( E \) (note that index arithmetic is done modulo \( n \)). We define the direction from node \( R_i \) to \( R_{i+1} \) as "clockwise" (CW) and the reverse direction as "anticlockwise" (AC).

As there may be several rings in a graph, we number each ring with a distinct ring ID \( RID \).

```
R0 . . . R1
   .
R7       R2
Anti-Clockwise   .       Clockwise
v .      RID = 17     v
    .       .
R6       R3
   .
R5 . . . R4
```

Figure 1: Ring with 8 nodes

The following terminology is used for ring LSPs:
Ring ID (RID): A non-zero number that identifies a ring; this is unique in some scope of a Service Provider’s network. A node may belong to multiple rings.

Ring node: A member of a ring. Note that a device may belong to several rings.

Node index: A logical numbering of nodes in a ring, from zero up to one less than the ring size. Used purely for exposition in this document.

Ring master: The ring master initiates the ring identification process. Mastership is indicated in the IGP by a two-bit field.

Ring neighbors: Nodes whose indices differ by one (modulo ring size).

Ring links: Links that connect ring neighbors.

Express links: Links that connect non-neighboring ring nodes.

Ring direction: A two-bit field in the IGP indicating the direction of a link. The choices are:

- UN: 00 undefined link
- CW: 01 clockwise ring link
- AC: 10 anticlockwise ring link
- EX: 11 express link

Ring Identification: The process of discovering ring nodes, ring links, link directions, and express links.

The following notation is used for ring LSPs:

- \( R_k \): A ring node with index \( k \). \( R_k \) has AC neighbor \( R_{(k-1)} \) and CW neighbor \( R_{(k+1)} \).

- \( RL_k \): A (unicast) Ring LSP anchored on node \( R_k \).

- \( CL_{jk} \): A label allocated by \( R_j \) for \( RL_k \) in the CW direction.

- \( AL_{jk} \): A label allocated by \( R_j \) for \( RL_k \) in the AC direction.

- \( P_{jk} \ (Q_{jk}) \): A Path (Resv) message sent by \( R_j \) for \( RL_k \).
2. Motivation

A ring is the simplest topology that offers resilience. This is perhaps the main reason to lay out fiber in a ring. Thus, effective mechanisms for fast failover on rings are needed. Furthermore, there are large numbers of rings. Thus, configuration of rings needs to be as simple as possible. Finally, bandwidth management on access rings is very important, as bandwidth is generally quite constrained here.

The goals of this document are to present mechanisms for improved MPLS-based resilience in ring networks (using ideas that are reminiscent of Bidirectional Line Switched Rings), for automatic bring-up of LSPs, better bandwidth management and for auto-hierarchy. These goals can be achieved using extensions to existing IGP and MPLS signaling protocols, using central provisioning, or in other ways.

3. Theory of Operation

Say a ring has ring ID RID. The ring is provisioned by choosing one or more ring masters for the ring and assigning them the RID. Other nodes in the ring may also be assigned this RID, or may be configured as "promiscuous". Ring discovery then kicks in. When each ring node knows its CW and AC ring neighbors and its ring links, and all express links have been identified, ring identification is complete.

Once ring identification is complete, each node signals one or more ring LSPs RL_i. RL_i, anchored on node R_i, consists of two counter-rotating unicast LSPs that start and end at R_i. A ring LSP is "multipoint": any node R_j can use RL_i to send traffic to R_i; this can be in either the CW or AC directions, or both (i.e., load balanced). Both of these counter-rotating LSPs are "active"; the choice of direction to send traffic to R_i is determined by policy at the node where traffic is injected into the ring. The default is to send traffic along the shortest path. Bidirectional connectivity between nodes R_i and R_j is achieved by using two different ring LSPs: R_i uses RL_j to reach R_j, and R_j uses RL_i to reach R_i.

3.1. Provisioning

The goal here is to provision rings with the absolute minimum configuration. The exposition below aims to achieve that using auto-discovery via a link-state IGP (see Section 4). Of course, auto-discovery can be overriden by configuration. For example, a link that would otherwise be classified by auto-discovery as a ring link might be configured not to be used for ring LSPs.
3.2. Ring Nodes

Ring nodes have a loopback address, and run a link-state IGP and an MPLS signaling protocol. To provision a node as a ring node for ring RID, the node is simply assigned that RID. A node may be part of several rings, and thus may be assigned several ring IDs.

To simplify ring provisioning even further, a node N may be made "promiscuous" by being assigned an RID of 0. A promiscuous node listens to RIDs in its IGP neighbors' link-state updates. For every non-zero RID N hears from a neighbor, N joins the corresponding ring by taking on that RID. In many situations, the use of promiscuous mode means that only one or two nodes in a ring needs to be provisioned; everything else is auto-discovered.

A ring node indicates in its IGP updates the ring LSP signaling protocols it supports. This can be LDP and/or RSVP-TE. Ideally, each node should support both.

3.3. Ring Links and Directions

Ring links must be MPLS-capable. They are by default unnumbered, point-to-point (from the IGP point of view) and "auto-bundled". The last attribute means that parallel links between ring neighbors are considered as a single link, without the need for explicit configuration for bundling (such as a Link Aggregation Group). Note that each component may be advertised separately in the IGP; however, signaling messages and labels across one component link apply to all components. Parallel links between a pair of ring nodes is often the result of having multiple lambdas or fibers between those nodes. RMR is primarily intended for operation at the packet layer; however, parallel links at the lambda or fiber layer result in parallel links at the packet layer.

A ring link is not provisioned as belonging to the ring; it is discovered to belong to ring RID if both its adjacent nodes belong to RID. A ring link’s direction (CW or AC) is also discovered; this process is initiated by the ring’s ring master. Note that the above two attributes can be overridden by provisioning if needed; it is then up to the provisioning system to maintain consistency across the ring.

3.3.1. Express Links

Express links are discovered once ring nodes, ring links and directions have been established. As defined earlier, express links are links joining non-neighboring ring nodes; often, this may be the
result of optically bypassing ring nodes. The use of express links will be described in a future version of this document.

3.4. Ring LSPs

Ring LSPs are not provisioned. Once a ring node $R_i$ knows its RID, its ring links and directions, it kicks off ring LSP signaling automatically. $R_i$ allocates CW and AC labels for each ring LSP $RL_k$. $R_i$ also initiates the creation of $RL_i$. As the signaling propagates around the ring, CW and AC labels are exchanged. When $R_i$ receives CW and AC labels for $RL_k$ from its ring neighbors, primary and fast reroute (FRR) paths for $RL_k$ are installed at $R_i$. More details are given in Section 5.

For RSVP-TE LSPs, bandwidths may be signaled in both directions. However, these are not provisioned either; rather, one does "reverse call admission control". When a service needs to use an LSP, the ring node where the traffic enters the ring attempts to increase the bandwidth on the LSP to the egress. If successful, the service is admitted to the ring.

3.5. Installing Primary LFIB Entries

In setting up $RL_k$, a node $R_j$ sends out two labels: $CL_{jk}$ to $R_{j-1}$ and $AL_{jk}$ to $R_{j+1}$. $R_j$ also receives two labels: $CL_{j+1,k}$ from $R_{j+1}$, and $AL_{j-1,k}$ from $R_{j-1}$. $R_j$ can now set up the forwarding entries for $RL_k$. In the CW direction, $R_j$ swaps incoming label $CL_{jk}$ with $CL_{j+1,k}$ with next hop $R_{j+1}$; these allow $R_j$ to act as LSR for $RL_k$. $R_j$ also installs an LFIB entry to push $CL_{j+1,k}$ with next hop $R_{j+1}$ to act as ingress for $RL_k$. Similarly, in the AC direction, $R_j$ swaps incoming label $AL_{jk}$ with $AL_{j-1,k}$ with next hop $R_{j-1}$ (as LSR), and an entry to push $AL_{j-1,k}$ with next hop $R_{j-1}$ (as ingress).

Clearly, $R_k$ does not act as ingress for its own LSPs. However, $R_k$ can send OAM messages, for example, an MPLS ping or traceroute ([I-D.ietf-mpls-rfc4379bis]), using labels $CL_{k,k+1}$ and $AL_{k-1,k}$, to test the entire ring LSP anchored at $R_k$ in both directions. Furthermore, if these LSPs use UHP, then $R_k$ installs LFIB entries to pop $CL_{k,k}$ for packets received from $R_k-1$ and to pop $AL_{k,k}$ for packets received from $R_k+1$.

3.6. Installing FRR LFIB Entries

At the same time that $R_j$ sets up its primary CW and AC LFIB entries, it can also set up the protection forwarding entries for $RL_k$. In the CW direction, $R_j$ sets up an FRR LFIB entry to swap incoming label $CL_{jk}$ with $AL_{j-1,k}$ with next hop $R_{j-1}$. In the AC direction,
R_j sets up an FRR LFIB entry to swap incoming label AL_jk with CL_j+1,k with next hop R_j+1. Again, R_k does not install FRR LFIB entries in this manner.

3.7. Protection

In this scheme, there are no protection LSPs as such -- no node or link bypass LSPs, no standby LSPs, no detours, and no LFA-type protection. Protection is via the "other" direction around the ring, which is why ring LSPs are in counter-rotating pairs. Protection works in the same way for link, node and ring LSP failures.

If a node R_j detects a failure from R_j+1 -- either all links to R_j+1 fail, or R_j+1 itself fails, R_j switches traffic on all CW ring LSPs to the AC direction using the FRR LFIB entries. If the failure is specific to a single ring LSP, R_j switches traffic just for that LSP. In either case, this switchover can be very fast, as the FRR LFIB entries can be preprogrammed. Fast detection and fast switchover lead to minimal traffic loss.

R_j then sends an indication to R_j-1 that the CW direction is not working, so that R_j-1 can similarly switch traffic to the AC direction. For RSVP-TE, this indication can be a PathErr or a Notify; other signaling protocols have similar indications. These indications propagate AC until each traffic source on the ring AC of the failure uses the AC direction. Thus, within a short period, traffic will be flowing in the optimal path, given that there is a failure on the ring. This contrasts with (say) bypass protection, where until the ingress recomputes a new path, traffic will be suboptimal.

Note that the failure of a node or a link will not necessarily affect all ring LSPs. Thus, it is important to identify the affected LSPs (and switch them), but to leave the rest alone.

One point to note is that when a ring node, say R_j, fails, RL_j is clearly unusable. However, the above protection scheme will cause a traffic loop: R_j-1 detects a failure CW, and protects by sending CW traffic on RL_j back all the way to R_j+1, which in turn sends traffic to R_j-1, etc. There are three proposals to avoid this:

1. Each ring node acting as ingress sends traffic with a TTL of at most 2*n, where n is the number of nodes in the ring.

2. A ring node sends protected traffic (i.e., traffic switched from CW to AC or vice versa) with TTL just large enough to reach the egress.
3. A ring node sends protected traffic with a special purpose label below the ring LSP label. A protecting node first checks for the presence of this label; if present, it means that the traffic is looping and MUST be dropped.

It is recommended that (2) be implemented. The other methods are optional.

4. Autodiscovery

4.1. Overview

Auto-discovery proceeds in three phases. The first phase is the announcement phase. The second phase is the mastership phase. The third phase is the ring identification phase.

![Diagram of a ring with non-ring nodes and links]

The format of an RMR Node Type-Length-Value (TLV) is given below. It consists of information pertaining to the node and optionally, sub-TLVs. A Neighbor sub-TLV contains information pertaining to the node’s neighbors. Other sub-TLVs may be defined in the future. Details of the format specific to IS-IS and OSPF will be given in the corresponding IGP documents.

```
[RMR Node Type][RMR Node Length][RID][Node Flags][sub-TLVs]

Ring Node TLV Format

[RMR Nbr Type][RMR Nbr Length][Nbr Address][Nbr Flags]

Ring Neighbor Sub-TLV Format
```
Each node participating in an MPLS ring is assigned an RID; in the example, RID = 17. A node is also provisioned with a mastership value. Each node advertises a ring node TLV for each ring it is participating in, along with the associated flags. It then starts timer T1.

A node in promiscuous mode doesn’t advertise any ring node TLVs. However, when it hears a ring node TLV from an IGP neighbor, it joins that ring, and sends its own ring node TLV with that RID.

The announcement phase allows a ring node to discover other ring nodes in the same ring so that a ring master can be elected.

When timer T1 fires, a node enters the mastership phase. In this phase, each ring node N starts timer T2 and checks if it is master. If it is the node with the lowest loopback address of all nodes with the highest mastership values, N declares itself master by re-advertising its ring node TLV with the M bit set.

When timer T2 fires, each node examines the ring node TLVs from all other nodes in the ring to identify the ring master. There should be
exactly one; if not, each node restarts timer T2 and tries again. The nodes that set their M bit should be extra careful in advertising their M bit in subsequent tries.

4.4. Ring Identification Phase

When there is exactly one ring master M, M enters the Ring Identification Phase. M indicates that it has successfully completed this phase by advertising ring link TLVs. This is the trigger for M’s CW neighbor to enter the Ring Identification Phase. This phase passes CW until all ring nodes have completed ring identification.

In the Ring Identification Phase, a node X that has two or more IGP neighbors that belong to the ring picks one of them to be its CW ring neighbor. If X is the ring master, it also picks a node as its AC ring neighbor. If there are exactly two such nodes, this step is trivial. If not, X computes a ring that includes all nodes that have completed the Ring Identification Phase (as seen by their ring link TLVs) and further contains the maximal number of nodes that belong to the ring. Based on that, X picks a CW neighbor and inserts ring link TLVs with ring direction CW for each link to its CW neighbor; X also inserts a ring link TLV with direction AC for each link to its AC neighbor. Then, X determines its express links. These are links connected to ring nodes that are not ring neighbors. X advertises ring link TLVs for express links by setting the link direction to "express link".

4.5. Ring Changes

The main changes to a ring are:

- ring link addition;
- ring link deletion;
- ring node addition; and
- ring node deletion.

The main goal of handling ring changes is (as much as possible) not to perturb existing ring operation. Thus, if the ring master hasn’t changed, all of the above changes should be local to the point of change. Link adds just update the IGP; signaling should take advantage of the new capacity as soon as it learns. Link deletions in the case of parallel links also show up as a change in capacity (until the last link in the bundle is removed.)
The removal of the last ring link between two nodes, or the removal of a ring node is an event that triggers protection switching. In a simple ring, the result is a broken ring. However, if a ring has express links, then it may be able to converge to a smaller ring with protection. Details of this process will be given in a future version.

The addition of a new ring node can also be handled incrementally. Again, the details of this process will be given in a future version.

5. Ring Signaling

A future version of this document will specify protocol-independent details about ring LSP signaling.

6. Ring OAM

Each ring node should advertise in its ring node TLV the OAM protocols it supports. Each ring node is expected to run a link-level OAM over each ring link. This should be an OAM protocol that both neighbors agree on. The default hello time is 3.3 milliseconds.

Each ring node also sends OAM messages over each direction of its ring LSP. This is a multi-hop OAM to check LSP liveness; typically, BFD would be used for this. The node chooses the hello interval; the default is once a second.

7. Advanced Topics

7.1. Half-rings

In some cases, a ring H may be incomplete, either because H is permanently missing a link (not just because of a failure), or because the link required to complete H is in a different IGP area. Either way, the ring discovery algorithm will fail. We call such a ring a "half-ring". Half-rings are sufficiently common that finding a way to deal with them effectively is a useful problem to solve. This topic will not be addressed in this document; that task is left for a future document.

7.2. Hub Node Resilience

Let’s call the node(s) that connect a ring to the rest of the network "hub node(s)" (usually, there are a pair of hub nodes.) Suppose a ring has two hub nodes H1 and H2. Suppose further that a non-hub ring node X wants to send traffic to some node Z outside the ring. This could be done, say, by having targeted LDP (T-LDP) sessions from H1 and H2 to X advertising LDP reachability to Z via H1 (H2); there
would be a two-label stack from X to reach Z. Say that to reach Z, X prefers H1; thus, traffic from X to Z will first go to H1 via a ring LSP, then to Z via LDP.

If H1 fails, traffic from X to Z will drop until the T-LDP session from H1 to Z fails, the IGP reconverges, and H2’s label to Z is chosen. Thereafter, traffic will go from X to H2 via a ring LSP, then to Z via LDP. However, this convergence could take a long time. Since this is a very common and important situation, it is again a useful problem to solve. However, this topic too will not be addressed in this document; that task is left for a future document.

8. Security Considerations

It is not anticipated that either the notion of MPLS rings or the extensions to various protocols to support them will cause new security loopholes. As this document is updated, this section will also be updated.

9. Acknowledgments

Many thanks to Pierre Bichon whose exemplar of self-organizing networks and whose urging for ever simpler provisioning led to the notion of promiscuous nodes.

10. IANA Considerations

There are no requests as yet to IANA for this document.

11. References

11.1. Normative References


11.2. Informative References


Authors’ Addresses

Kireeti Kompella
Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, CA  94089
USA
Email: kireeti.kompella@gmail.com

Luis M. Contreras
Telefonica
Ronda de la Comunicacion
Sur-3 building, 3rd floor
Madrid  28050
Spain
Email: luismiguel.contrerasmurillo@telefonica.com
URI:   http://people.tid.es/LuisM.Contreras/
Abstract

This document contains the specification for the MPLS Static Label Switched Paths (LSPs) YANG model. The model allows for the provisioning of static LSP(s) on LER(s) and LSR(s) devices along a LSP path without the dependency on any signaling protocol. The MPLS Static LSP model augments the MPLS base YANG model with specific data to configure and manage MPLS Static LSP(s).

Status of This Memo

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

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

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

This Internet-Draft will expire on August 19, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document.
1. Introduction

This document describes a YANG data model for configuring and managing the Static LSPs feature. The model allows the configuration of LER and LSR devices with the necessary MPLS cross-connects or bindings to realize an end-to-end LSP service.

A static LSP is established by manually specifying incoming and outgoing MPLS label(s) and necessary forwarding information on each of the traversed Label Edge Router (LER) and Label Switched Router (LSR) devices (ingress, transit, or egress nodes) of the forwarding path.

For example, on an ingress LER device, the model is used to associate a specific Forwarding Equivalence Class (FEC) of packets - e.g. matching a specific IP prefix in a Virtual Routing or Forwarding (VRF) instance - to an MPLS outgoing label imposition, next-hop(s) and respective outgoing interface(s) to forward the packet. On an LSR device, the model is used to create a binding that swaps the incoming label with an outgoing label and forwards the packet on one or multiple egress path(s). On an egress LER, it is used to create a binding that decapsulates the incoming MPLS label and performs forwarding based on the inner MPLS label (if present) or IP forwarding in the packet.

The MPLS Static LSP YANG model is defined in module "ietf-mpls-static" and augments the MPLS Base YANG model defined in module "ietf-mpls" in [I-D.saad-mpls-static-yang].
1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

The following terms are defined in [RFC6020]:

- augment,
- configuration data,
- data model,
- data node,
- feature,
- mandatory node,
- module,
- schema tree,
- state data,
- RPC operation.

1.2. Model Organization

The base MPLS Static LSP model covers the core features with the minimal set of configuration parameters needed to manage and operate MPLS Static LSPs.

Additional MPLS Static LSP parameters as well as optional feature(s) are grouped in a separate MPLS Static LSP extended model. The relationship between the MPLS base and other MPLS modules are shown in Figure 1.
1.3. MPLS Static LSPs Model Tree Diagram

The MPLS Static and extendend LSP tree diagram is shown in Figure 2.

```
module: ietf-mpls-static
augment /rt:route/mpls:mpls:
  +++-rw static-lsps
    +++-rw static-lsp* [name]
      +++-rw name             -> ../config/name
      +++-rw config
        +++-rw name?          string
        +++-rw operation?     mpls-operations-type
      +++-ro state
        +++-ro name?          string
        +++-ro operation?     mpls-operations-type
      +++-rw (out-segment)?
        +++-(simple-path)
          +++-rw simple-path
        ++--rw config
          ++--rw next-hop?     inet:ip-address
          ++--rw outgoing-label? rt-types:mpls-label
          ++--rw outgoing-interface? if:interface-ref
```

Figure 1: Relationship between MPLS modules
Figure 2: MPLS Static LSP tree diagram
1.4. MPLS Static LSP YANG Module(s)

The MPLS Static LSP module is shown in Figure 3.

```yaml
<CODE BEGINS> file "ietf-mpls-static@2017-07-02.yang"
module ietf-mpls-static {
    namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-static";
    prefix "mpls-static";
    import ietf-mpls {
        prefix mpls;
    }
    import ietf-routing {
        prefix "rt";
    }
    import ietf-routing-types {
        prefix "rt-types";
    }
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-interfaces {
        prefix "if";
    }
    /* Import TE generic types */
    import ietf-te {
        prefix te;
    }
    organization "IETF MPLS Working Group";
    contact
        "WG Web:   <http://tools.ietf.org/wg/mpls/>
        WG List: <mailto:mpls@ietf.org>
        WG Chair: Loa Andersson
                   <mailto:loa@pi.nu>
        WG Chair: Ross Callon
                   <mailto:rcallon@juniper.net>
```
description

"This YANG module augments the 'ietf-routing' module with basic configuration and operational state data for MPLS static";

revision "2017-07-02" {

description

"Latest revision:
  - Addressed MPLS-RT review comments";

reference "RFC 3031: A YANG Data Model for Static MPLS LSPs";
}

typedef static-lsp-ref {

type leafref {

  path "/rt:routing/mpls:mpls/mpls-static:static-lsps/" +
      "mpls-static:static-lsp/mpls-static:name";
}
typedef mpls-operations-type {
  type enumeration {
    enum impose-and-forward {
      description "Operation impose outgoing label(s) and forward to next-hop";
    }
    enum pop-and-forward {
      description "Operation pop incoming label and forward to next-hop";
    }
    enum pop-impose-and-forward {
      description "Operation pop incoming label, impose one or more outgoing label(s) and forward to next-hop";
    }
    enum swap-and-forward {
      description "Operation swap incoming label, with outgoing label and forward to next-hop";
    }
    enum pop-and-lookup {
      description "Operation pop incoming label and perform a lookup";
    }
  }
  description "MPLS operations types";
}

grouping path-basic_config {
  description "common definitions for statics";
  leaf next-hop {
    type inet:ip-address;
    description "next hop IP address for the LSP";
  }
  leaf outgoing-label {
    type rt-types:mpls-label;
    description "label value to push at the current hop for the LSP";
  }
}
leaf outgoing-interface {
  type if:interface-ref;
  description
    "The outgoing interface";
}

grouping path-outgoing-labels_config {
  description "Path outgoing labels grouping";
  leaf index {
    type uint8 {
      range "0..255";
    }
    description
      "Index of the label. Index 0 indicates top of the label stack";
  }
  leaf label {
    type rt-types:mpls-label;
    description
      "The outgoing MPLS labels to impose";
  }
}

grouping path-outgoing-labels {
  description "Path outgoing labels grouping";
  container outgoing-labels {
    description "List of outgoing labels";
    list outgoing-labels {
      key "index";
      description "Outgoing label list";
      leaf index {
        type leafref {
          path ".../config/index";
        }
        description
          "Index of the label. Index 0 indicates top of the label stack";
      }
      container config {
        description
          "Configuration intended parameters";
        uses path-outgoing-labels_config;
      }
      container state {
        config false;
      }
  }
}
grouping path-properties_config {
  description
  "MPLS path properties";
  leaf path-index {
    type uint16;
    description
    "Path identifier";
  }
  leaf backup-path-index {
    type uint16;
    description
    "Backup path identifier";
  }
  leaf next-hop {
    type inet:ip-address;
    description
    "The address of the next-hop";
  }
  leaf outgoing-interface {
    type if:interface-ref;
    description
    "The outgoing interface";
  }
  leaf loadshare {
    type uint16;
    description
    "This value is used to compute a loadshare to perform un-equal
    load balancing when multiple outgoing path(s) are specified. A
    share is computed as a ratio of this number to the total under
    all configured path(s).";
  }
  leaf role {
    type enumeration {
      enum PRIMARY {
        description
      }
    }
  }
}
"Path as primary traffic carrying";
}
enum BACKUP {
  description "Path acts as backup";
}
enum PRIMARY_AND_BACKUP {
  description "Path acts as primary and backup simultaneously";
}

description "The MPLS path role";
}

grouping static-lsp-paths {
  description "Static LSP path grouping";
  choice out-segment {
    description "The MPLS out-segment type choice";
    case simple-path {
      container simple-path {
        description "Simple path container";
        container config {
          description "Holds the intended configuration";
          uses path-basic_config;
        }
        container state {
          config false;
          description "Holds the state and inuse configuration";
          uses path-basic_config;
        }
      }
    }
    case multiple-paths {
      container paths {
        description "List of outgoing paths";
        list path {
          key path-index;
          description "The list of MPLS paths associated with the FEC";
          leaf path-index {
            type leafref {
              path "..//config/path-index";
            }
            description "Index of the path";
          }
        }
      }
    }
  }
}
container config {
  description
   "Holds the intended configuration";
  uses path-properties_config;
}

container state {
  config false;
  description
   "Holds the state and inuse configuration";
  uses path-properties_config;
}

uses path-outgoing-labels;
}
}


grouping in-segment_config {
  description "In-segment grouping";
  choice type {
    description
     "Basic FEC choice";
    case ip-prefix {
      leaf ip-prefix {
        type inet:ip-prefix;
        description "An IP prefix";
      }
    }
    case mpls-label {
      leaf incoming-label {
        type rt-types:mpls-label;
        description "label value on the incoming packet";
      }
    }
    case tunnel {
      leaf tunnel {
        type te:tunnel-ref;
        description "TE tunnel FEC mapping";
      }
    }
  }
  leaf incoming-interface {
    type if:interface-ref;
    description
     "Optional incoming interface if FEC is restricted to traffic incoming on a specific interface";
  }
}
grouping in-segment {
  description "In-segment grouping";
  container in-segment {
    description "MPLS incoming segment";
    container config {
      description "Holds the intended configuration";
      uses in-segment_config;
    }
    container state {
      config false;
      description "Holds the state and inuse configuration";
      uses in-segment_config;
    }
  }
}

grouping static-lsp-top_config {
  description "Static LSP configuration grouping";
  leaf name {
    type string;
    description "name to identify the LSP";
  }
  leaf operation {
    type mpls-operations-type;
    description "The MPLS operation to be executed on the incoming packet";
  }
}

grouping static-lsp-top {
  description "common definitions for static LSPs";
  container config {
    description "Holds the intended configuration";
    uses static-lsp-top_config;
  }
  container state {
    config false;
    description "Holds the state and inuse configuration";
    uses static-lsp-top_config;
  }
}
Figure 3: MPLS Static LSP YANG module

The extended MPLS Static LSP module is shown in Figure 4.

<CODE BEGINS> file "ietf-mpls-static-extended@2017-07-02.yang"
module ietf-mpls-static-extended {
    prefix "mpls-static-ext";
    import ietf-mpls {
        prefix "mpls";
    }
    import ietf-routing {
        prefix "rt";
    }
    import ietf-mpls-static {
        prefix "mpls-static";
    }
    organization "IETF MPLS Working Group";
}
<CODE ENDS>


contact

"WG Web:  <http://tools.ietf.org/wg/mpls/>

WG List:   <mailto:mpls@ietf.org>

WG Chair: Loa Andersson
          <mailto:loa@pi.nu>

WG Chair: Ross Callon
          <mailto:rcallon@juniper.net>

WG Chair: George Swallow
          <mailto:swallow.ietf@gmail.com>

Editor:   Tarek Saad
          <mailto:tsaad@cisco.com>

Editor:   Kamran Raza
          <mailto:skraza@cisco.com>

Editor:   Rakesh Gandhi
          <mailto:rgandhi@cisco.com>

Editor:   Xufeng Liu
          <mailto: xufeng.liu.ietf@gmail.com>

Editor:   Vishnu Pavan Beeram
          <mailto: vbeeram@juniper.net>

Editor:   Himanshu Shah
          <mailto: hshah@ciena.com>

Editor:   Igor Bryskin
          <mailto: Igor.Bryskin@huawei.com>

Editor:   Xia Chen
          <mailto: jescia.chenxia@huawei.com>

Editor:   Raqib Jones
          <mailto: raqib@Brocade.com>

Editor:   Bin Wen
          <mailto: Bin_Wen@cable.comcast.com">

description

"This module contains the Extended MPLS YANG data model.";

revision 2017-03-10 {

description "Latest revision of MPLS extended yang module.";
reference "RFC2205";
}

/* RSVP features */
feature bandwidth {
  description
    "Indicates support for static LSP bandwidth allocation";
}

grouping static-lsp-extended_config {
  description
    "Configuration parameters for MPLS extended parameters";
  leaf bandwidth {
    type uint32;
    description
      "bandwidth in Mbps, e.g., using offline calculation";
  }
  leaf lsp-priority-setup {
    type uint8 {
      range "0..7";
    }
    description "LSP setup priority";
  }
  leaf lsp-priority-hold {
    type uint8 {
      range "0..7";
    }
    description "LSP hold priority";
  }
}

grouping bidir-static-lsp_config {
  description "common definitions for static LSPs";
  leaf forward-lsp {
    type mpls-static:static-lsp-ref;
    description
      "Reference to a configured static forward LSP";
  }
  leaf reverse-lsp {
    type mpls-static:static-lsp-ref;
    description
      "Reference to a configured static reverse LSP";
  }
}

grouping bidir-static-lsp {


Internet-Draft    MPLS Static LSPs YANG Data Model    February 2018

description "grouping for top level list of static LSPs";
container config {
    description
        "Holds the intended configuration";
    uses bidir-static-lsp_config;
}
container state {
    config false;
    description
        "Holds the state and inuse configuration";
    uses bidir-static-lsp_config;
}

augment "/rt:routing/mpls:mpls/mpls-static:static-lsps" {
    description
        "RSVP signaling all interfaces configuration extensions";
    uses static-lsp-extended_config;
}

augment "/rt:routing/mpls:mpls" {
    description "Augmentations for MPLS Static LSPs";
    container bidir-static-lsps {
        description
            "Statically configured LSPs, without dynamic signaling";
        list bidir-static-lsp {
            key name;
            description "list of defined static LSPs";

            leaf name {
                type string;
                description "name to identify the LSP";
            }
        }
    }
}

<CODE ENDS>

Figure 4: Extended MPLS Static LSP YANG module

2.  IANA Considerations

This document registers the following URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.
This document registers two YANG modules in the YANG Module Names registry [RFC6020].


3. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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

4. Normative References


Internet-Draft      MPLS Static LSPs YANG Data Model       February 2018

[ RFC3688 ]  Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
             DOI 10.17487/RFC3688, January 2004,

             the Network Configuration Protocol (NETCONF)", RFC 6020,
             DOI 10.17487/RFC6020, October 2010,

             and A. Bierman, Ed., "Network Configuration Protocol
             (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,

[ RFC6242 ]  Wasserman, M., "Using the NETCONF Protocol over Secure
             Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011,

             Protocol (NETCONF) Access Control Model", RFC 6536,
             DOI 10.17487/RFC6536, March 2012,

Authors' Addresses

Tarek Saad
Cisco Systems, Inc.
Email: tsaad@cisco.com

Kamran Raza
Cisco Systems, Inc.
Email: skraza@cisco.com

Rakesh Gandhi
Cisco Systems, Inc.
Email: rgandhi@cisco.com

Xufeng Liu
Jabil
Email: Xufeng_Liu@jabil.com
Abstract

This document describes procedures for using Bidirectional Forwarding Detection (BFD) in Demand mode to detect data plane failures in Multiprotocol Label Switching (MPLS) point-to-point Label Switched Paths.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 21, 2018.

Copyright Notice

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

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

1. Introduction ................................. 2
2. Conventions used in this document ............. 2
   2.1. Terminology ............................... 2
   2.2. Requirements Language .................... 2
3. Use of the BFD Demand Mode .................... 3
4. IANA Considerations ........................... 4
5. Security Considerations ....................... 4
6. Normative References .......................... 4
Appendix A. Acknowledgements ........................ 5
Author's Address ................................. 5

1. Introduction

[RFC5884] defined use of the Asynchronous method of Bidirectional Detection (BFD) [RFC5880] to monitor and detect failures in the data path of a Multiprotocol Label Switching (MPLS) Label Switched Path (LSP). Use of the Demand mode, also specified in RFC5880, has not been defined so far. This document describes procedures for using the Demand mode of BFD protocol to detect data plane failures in MPLS point-to-point (p2p) LSPs.

2. Conventions used in this document

2.1. Terminology

MPLS: Multiprotocol Label Switching
LSP: Label Switched Path
LER: Label switching Edge Router
BFD: Bidirectional Forwarding Detection
p2p: Point-to-Point

2.2. Requirements Language

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

[RFC5880] defines that the Demand mode MAY be:

- asymmetric, i.e. used in one direction of a BFD session;
- switched to and from without bringing BFD session to Down state through using a Poll Sequence.

For the case of BFD over MPLS LSP, ingress Label switching Edge Router (LER) usually acts as Active BFD peer and egress LER acts as Passive BFD peer. The Active peer bootstraps the BFD session by using LSP ping. Once the BFD session is in Up state the ingress LER that supports this specification MUST switch to the Demand mode by setting Demand (D) bit in its Control packet and initiating a Poll Sequence. If the egress LER supports this specification it MUST respond with the Final (F) bit set in its BFD Control packet sent to the ingress LER and ceases further transmission of periodic BFD control packets to the ingress LER.

In this state BFD peers MAY remain as long as the egress LER is in Up state. The ingress LER MAY check liveness of the egress LER by setting the Poll flag. The egress LER will respond by transmitting BFD control packet with the Final flag set. If the ingress LER doesn’t receive BFD packet with the Final flag from its peer after the predetermined period of time, default wait time recommended 1 second, the ingress MAY transmit another packet with the Poll flag set. If ingress doesn’t receive BFD control packet with the Final flag set in response to three consecutive packets with Poll flag, it MAY declare the BFD peer non-responsive and change state of the BFD session to Down state.

If the Detection timer at the egress LER expires it MUST send BFD Control packet to the ingress LER with the Poll (P) bit set, Status (Sta) field set to Down value, and the Diagnostic (Diag) field set to Control Detection Time Expired value. The egress LER sends these Control packets to the ingress LER at the rate of one per second until either it receives the valid for this BFD session control packet with the Final (F) bit set from the ingress LER or the defect condition clears and the BFD session state reaches Up state at the egress LER.

The ingress LER transmits BFD Control packets over the MPLS LSP with the Demand (D) flag set at negotiated interval per [RFC5880], the greater of bfd.DesiredMinTxInterval and bfd.RemoteMinRxInterval, until it receives the valid BFD packet from the egress LER with the Poll (P) bit and the Diagnostic (Diag) field value Control Detection Time Expired. Reception of such BFD control packet by the ingress
LER indicates that the monitored LSP has a failure and sending BFD control packet with the Final flag set to acknowledge failure indication is likely to fail. Instead, the ingress LER transmits the BFD Control packet to the egress LER over the IP network with:

- destination IP address MUST be set to the destination IP address of the LSP Ping Echo request message [RFC8029];
- destination UDP port set to 4784 [RFC5883];
- Final (F) flag in BFD control packet MUST be set;
- Demand (D) flag in BFD control packet MUST be cleared.

The ingress LER changes the state of the BFD session to Down and changes rate of BFD Control packets transmission to one packet per second. The ingress LER in Down mode changes to Asynchronous mode until the BFD session comes to Up state once again. Then the ingress LER switches to the Demand mode.

4. IANA Considerations

TBD

5. Security Considerations

This document does not introduce new security aspects but inherits all security considerations from [RFC5880], [RFC5884], [RFC7726], [RFC8029], and [RFC6425].

6. Normative References


Appendix A. Acknowledgements

TBD

Author’s Address

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com
Abstract

This document describes procedures for using Bidirectional Forwarding Detection (BFD) for multipoint networks to detect data plane failures in Multiprotocol Label Switching (MPLS) point-to-multipoint (p2mp) Label Switched Paths (LSPs). It also describes applicability of out-band solutions to bootstrap a BFD session in this environment.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 19, 2018.

Copyright Notice

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

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

[I-D.ietf-bfd-multipoint] defines a method of using Bidirectional Detection (BFD) [RFC5880] to monitor and detect unicast failures between the sender (head) and one or more receivers (tails) in multipoint or multicast networks. This document describes procedures for using such mode of BFD protocol to detect data plane failures in Multiprotocol Label Switching (MPLS) point-to-multipoint (p2mp) Label Switched Paths (LSPs). The document also describes applicability of out-band solutions to bootstrap a BFD session in this environment.

2. Conventions used in this document

2.1. Terminology

MPLS: Multiprotocol Label Switching

LSP: Label Switched Path

BFD: Bidirectional Forwarding Detection

p2mp: Point-to-Multipoint

FEC: Forwarding Equivalence Class

G-ACh: Generic Associated Channel

ACH: Associated Channel Header
GAL: G-ACh Label

2.2. Requirements Language

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

3. Multipoint BFD Encapsulation

[I-D.ietf-bfd-multipoint] defines how the tail of multipoint BFD session demultiplexes received BFD control packet when Your Discriminator is not set, i.e. equals zero. Because [I-D.ietf-bfd-multipoint] uses BFD in Demand mode the head of BFD multipoint session transmits BFD control packets with Your Discriminator set to zero. As result, a tail cannot demultiplex BFD sessions using Your Discriminator, as defined in [RFC5880].

[I-D.ietf-bfd-multipoint] requires that in order to demultiplex BFD sessions the tail uses source IP address, My Discriminator and the identity of the multipoint tree which the Multipoint BFD Control packet was received from. The identity of the multipoint tree MAY be provided by the p2mp MPLS LSP label in case of inclusive p-tree or upstream assigned label in case of aggregate p-tree. The source IP address MAY be drawn from the IP header, if BFD control packet transmitted by the head using IP/UDP encapsulation as described in Section 3.1. Non-IP encapsulation case described in Section 3.2.

3.1. IP Encapsulation of Multipoint BFD

[I-D.ietf-bfd-multipoint] defines IP/UDP encapsulation for multipoint BFD over p2mp MPLS LSP:

UDP destination port MUST be set to 3784;

destination IP address MUST be from the 127/8 range for IPv4 and from the 0:0:0:0:0:FFFF:7F00/104 range for IPv6;

This specification further clarifies that:

if multiple alternative paths for the given p2mp LSP Forwarding Equivalence Class(FEC) exist, the MultipointHead SHOULD use Entropy Label [RFC6790] used for LSP Ping [RFC8029] to exercise that particular alternative path;
or the MultipointHead MAY use, as destination IP address, the IP address discovered by LSP Ping traceroute [RFC8029] to exercise that particular alternate path.

3.2. Non-IP Encapsulation of Multipoint BFD

Non-IP encapsulation for multipoint BFD over p2mp MPLS LSP MUST use Generic Associated Channel (G-ACh) Label (GAL) [RFC5586] at the bottom of the label stack followed by Associated Channel Header (ACH). Channel Type field in ACH MUST be set to BFD CV [RFC6428]. To provide identity of the MultipointHead for the particular multipoint BFD session this document defines new Source MEP ID type TBA1 Section 6.1 IP Address. If the Length value is 4, then the Value field contains IPv4 address. If the Length value is 16, then the Value field contains IPv6 address. Any other value of the Length field MUST be considered as error and the BFD control packet MUST be discarded.

4. Bootstrapping Multipoint BFD

4.1. LSP Ping

MultipointHead MAY use LSP Ping [RFC8029] using in Target FEC TLV, as appropriate, sub-TLVs defined in Section 3.1 [RFC6425].

4.2. Control Plane

BGP-BFD Attribute [I-D.ietf-bess-mvpn-fast-failover] MAY be used to bootstrap multipoint BFD session on a tail.

5. Security Considerations

This document does not introduce new security aspects but inherits all security considerations from [RFC5880], [RFC5884], [RFC7726], [I-D.ietf-bfd-multipoint], [RFC8029], and [RFC6425].

6. IANA Considerations

6.1. Source MEP ID IP Address Type

IANA is required to allocate value (TBD) for the Source Source MEP ID IP Address type from the "CC/CV MEP-ID TLV" registry which is under the "Pseudowire Associated Channel Types" registry.
Table 1: Source MEP ID IP Address TLV Type

7. Acknowledgements

8. Normative References

[I-D.ietf-bess-mvpn-fast-failover]

[I-D.ietf-bfd-multipoint]


Author’s Address

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com
Bidirectional Forwarding Detection (BFD) in Segment Routing Networks
Using MPLS Dataplane
draft-mirsky-spring-bfd-06

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on February 24, 2019.
1. Introduction

[RFC5880], [RFC5881], and [RFC5883] defined the operation of Bidirectional Forwarding Detection (BFD) protocol between the two systems over IP networks. [RFC5884] and [RFC7726] set rules for using BFD Asynchronous mode over point-to-point (p2p) Multiprotocol Label Switching (MPLS) Label Switched Path (LSP). These latter standards implicitly assume that the egress BFD peer, which is the egress Label Edge Router (LER), will use the shortest path route
regardless of the path the ingress LER uses to send BFD Control packets towards it.

This document defines the use of LSP Ping for Segment Routing networks over MPLS data plane [RFC8287] to bootstrap and control path of a BFD session from the egress to ingress LER using Segment Routing tunnel with MPLS data plane (SR-MPLS).

1.1. Conventions

1.1.1. Terminology

  BFD: Bidirectional Forwarding Detection
  FEC: Forwarding Equivalence Class
  MPLS: Multiprotocol Label Switching
  SR-MPLS Segment Routing with MPLS data plane
  LSP: Label Switched Path
  LSR Label Switching Router
  LER Label Edge Router
  p2p Point-to-point
  SID Segment Identifier
  SR Segment Routing

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 BCP
  14 [RFC2119] [RFC8174] when, and only when, they appear in all
  capitals, as shown here.

2. Bootstrapping BFD Session over Segment Routed Tunnel with MPLS Data Plane

  Use of an LSP Ping to bootstrap BFD over MPLS LSP is required, as
documented in [RFC5884], to establish an association between a fault
detection message, i.e., BFD Control message, and the Forwarding Equivalency Class (FEC) of a single label stack LSP in case of Penultimate Hop Popping or when the egress Label Switching Router
(LSR) distributes the Explicit NULL label to the penultimate hop router. The Explicit NULL label is not advertised as a Segment Identifier (SID) by an SR node but, as demonstrated in section 3.1 [I-D.ietf-spring-segment-routing-mpls] if the operation at the penultimate hop is NEXT; then the egress SR node will receive an IP encapsulated packet. Thus the conclusion is that LSP Ping MUST be used to bootstrap a BFD session in SR-MPLS domain.

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

The initiator, i.e., ingress LSR, MUST include FEC(s) corresponding to the destination segment.

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

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

When LSP Ping is used to bootstraping a BFD session for SR-MPLS tunnel the FEC corresponding to the segment to be associated with the BFD session MUST be as the very last sub-TLV in the Target FEC TLV.

If the target segment is an anycast prefix segment ([I-D.ietf-spring-mpls-anycast-segments]) the corresponding Anycast SID MUST be included in the Target TLV as the very last sub-TLV. Also, for BFD control packet the ingress SR node MUST use precisely the same label stack encapsulation, especially Entropy Label ([RFC6790]), as for the LSP ping with the BFD Discriminator TLV that bootstrapped the BFD session. Other operational aspects of using BFD to monitor the continuity of the path to the particular Anycast SID, advertised by a group of SR-MPLS capable nodes, will be considered in the future versions of the document.

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

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

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

4. Use Non-FEC Path TLV

For the case of MPLS data plane, Segment Routing Architecture [RFC8402] explains that "a segment is encoded as an MPLS label. An ordered list of segments is encoded as a stack of labels." YANG Data Model for MPLS Static LSPs [I-D.ietf-mpls-static-yang] models outgoing MPLS labels to be imposed as leaf-list [RFC6020], i.e., as array of rt-types:mpls-label [RFC8294].

This document defines new optional Non-FEC Path TLV. The format of the Non-FEC Path TLV is presented in Figure 1

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Non-FEC Path TLV Type    |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                             |
|                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Non-FEC Path                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Non-FEC Path TLV Format

Non-FEC Path TLV Type is two octets in length and has a value of TBD1 (to be assigned by IANA as requested in Section 7.1).

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

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

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

This document defines Static Routing MPLS Tunnel sub-TLV that MAY be
used with the Non-FEC Path TLV. The format of the sub-TLV is
presented in Figure 2.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SR MPLS Tunnel sub-TLV Type |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Label Entry 1 (Top Label)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Label Entry 2                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Label Entry N (Bottom Label)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Segment Routing MPLS Tunnel sub-TLV

The Segment Routing MPLS Tunnel sub-TLV Type is two octets in length,
and has a value of TBD2 (to be assigned by IANA as requested in
Section 7.1).

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

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

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

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

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

- ingress SR node bootstraps BFD session over SR-MPLS in Async BFD mode;
- once BFD session is Up, the ingress node switches the egress BFD node into the Demand mode by setting D field in BFD Control packet it transmits;
- if the egress BFD node detects the failure of the BFD session, it sends its BFD control packet to the ingress over the IP network with Poll sequence;
- if the ingress node receives a BFD control packet from the remote node in a Demand mode with Poll sequence and Diag field indicating the failure, the ingress transmits BFD control packet with Final over IP and switches the BFD over SR-MPLS back into Async mode, sending BFD Control packets one per second.

7. IANA Considerations

7.1. Non-FEC Path TLV

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

```
+-------+-----------------+---------------+
| Value | TLV Name         | Reference     |
+-------+-----------------+---------------+
| TBD1  | Non-FEC Path TLV| This document |
+-------+-----------------+---------------+
```

Table 1: New Non-FEC Path TLV
IANA is requested to create new Non-FEC Path sub-TLV registry for the Non-FEC Path TLV as described in Table 2.

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

Table 2: Non-FEC Path sub-TLV registry

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>This document</td>
</tr>
<tr>
<td>TBD2</td>
<td>Segment Routing MPLS Tunnel sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>65535</td>
<td>Reserved</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 3: New Segment Routing Tunnel sub-TLV

7.2. Return Code

IANA is requested to create Non-FEC Path sub-TLV sub-registry for the new Non-FEC Path TLV and 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.
8. Security Considerations

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

9. Acknowledgments

TBD

10. References

10.1. Normative References

[I-D.ietf-mpls-bfd-directed]

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

[I-D.mirsky-bfd-mpls-demand]
Mirsky, G., "BFD in Demand Mode over Point-to-Point MPLS LSP", draft-mirsky-bfd-mpls-demand-03 (work in progress), June 2018.


10.2. Informative References

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

[I-D.ietf-spring-mpls-anycast-segments]


Authors’ Addresses

Greg Mirsky
ZTE Corp.
Email: gregimirsky@gmail.com

Jeff Tantsura
Nuage Networks
Email: jefftant.ietf@gmail.com

Ilya Varlashkin
Google
Email: Ilya@nobulus.com
Abstract

As the scale of MPLS RSVP-TE networks has grown, so the number of Label Switched Paths (LSPs) supported by individual network elements has increased. Various implementation recommendations have been proposed to manage the resulting increase in control plane state.

However, those changes have had no effect on the number of labels that a transit Label Switching Router (LSR) has to support in the forwarding plane. That number is governed by the number of LSPs transiting or terminated at the LSR and is directly related to the total LSP state in the control plane.

This document defines a mechanism to prevent the maximum size of the label space limit on an LSR from being a constraint to control plane scaling on that node. That is, it allows many more LSPs to be supported than there are forwarding plane labels available.

This work introduces the notion of pre-installed ‘per Traffic Engineering (TE) link labels’ that can be shared by MPLS RSVP-TE LSPs that traverse these TE links. This approach significantly reduces the forwarding plane state required to support a large number of LSPs. This couples the feature benefits of the RSVP-TE control plane with the simplicity of the Segment Routing MPLS forwarding plane.

This document also introduces the ability to mix different types of label operations along the path of an LSP, thereby allowing the ingress router or an external controller to influence how to optimally place a LSP in the network.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on June 12, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction ........................................... 3
2. Terminology .......................................... 5
3. Allocation of TE Link Labels ............................ 5
4. Segment Routed RSVP-TE Tunnel Setup .................... 5
5. Delegating Label Stack Imposition ...................... 7
   5.1. Stacking at the Ingress ............................ 8
      5.1.1. Stack to Reach Delegation Hop ................ 8
      5.1.2. Stack to Reach Egress ....................... 9
   5.2. Explicit Delegation ............................... 10
   5.3. Automatic Delegation ................................ 10
      5.3.1. Effective Transport Label-Stack Depth (ETLD) . 10
6. Mixing TE Link Labels and Regular Labels in an RSVP-TE Tunnel 11
7. Construction of Label Stacks ................................. 12
8. Facility Backup Protection ................................. 13
   8.1. Link Protection ........................................... 13
   8.2. Node Protection ........................................... 14
9. Quantifying TE Link Labels ................................. 14
10. Protocol Extensions ........................................ 14
    10.1. Requirements ........................................... 14
    10.2. Attribute Flags TLV: TE Link Label ................. 15
    10.3. RRO Label Subobject Flag: TE Link Label .......... 15
    10.4. Attribute Flags TLV: LSI-D ............................. 15
    10.5. RRO Label Subobject Flag: Delegation Label .......... 16
    10.6. Attributes Flags TLV: LSI-D-S2E ...................... 16
    10.7. Attributes TLV: ETLD ................................... 16
11. OAM Considerations ........................................ 17
12. Acknowledgements .......................................... 17
13. Contributors ................................................ 17
14. IANA Considerations ........................................ 18
    14.1. Attribute Flags: TE Link Label, LSI-D, LSI-D-S2E .. 18
    14.2. Attribute TLV: ETLD ................................... 18
    14.3. Record Route Label Sub-object Flags: TE Link Label, Delegation Label ...... 18
15. Security Considerations .................................... 19
16. References .................................................. 19
    16.1. Normative References .................................. 19
    16.2. Informative References ................................. 20
Authors’ Addresses ............................................. 21

1. Introduction

The scaling of RSVP-TE [RFC3209] control plane implementations can be improved by adopting the guidelines and mechanisms described in [RFC2961] and [I-D.ietf-teas-rsvp-te-scaling-rec]. These documents do not make any difference to the forwarding plane state required to handle the control plane state. The forwarding plane state remains unchanged and is directly proportional to the total number of Label Switching Paths (LSPs) supported by the control plane.

This document describes a mechanism that prevents the size of the platform specific label space on a Label Switching Router (LSR) from being a constraint to pushing the limits of control plane scaling on that node.

This work introduces the notion of pre-installed ‘per Traffic Engineering (TE) link labels’ that are allocated by an LSR. Each such label is installed in the MPLS forwarding plane with a ‘pop’ operation and the instruction to forward the received packet over the TE link. An LSR advertises this label in the Label object of a Resv
message as LSPs are set up and they are recorded hop by hop in the Record Route object (RRO) of the Resv message as it traverses the network. To make use of this feature, the ingress Label Edge Router (LER) pushes a stack of labels [RFC3031] as received in the RRO. These ‘TE link labels’ can be shared by MPLS RSVP-TE LSPs that traverse the same TE link.

This forwarding plane behavior fits in the MPLS architecture [RFC3031] and is same as that exhibited by Segment Routing (SR) [I-D.ietf-spring-segment-routing] when using an MPLS forwarding plane and a series of adjacency segments. This work couples the feature benefits of the RSVP-TE control plane with the simplicity of the Segment Routing MPLS forwarding plane. The RSVP-TE tunnels that use this shared forwarding plane can co-exist with MPLS-SR LSPs [I-D.ietf-spring-segment-routing-mpls] as described in [I-D.ietf-teas-sr-rsvp-coexistence-rec].

RSVP-TE using a shared MPLS forwarding plane offers the following benefits:

1. Shared Labels: The transit label on a TE link is shared among RSVP-TE tunnels traversing the link and is used independent of the ingress and egress of the LSPs.

2. Faster LSP setup time: No forwarding plane state needs to be programmed during LSP setup and teardown resulting in faster time for provisioning and deprovisioning LSPs.

3. Hitless re-routing: New transit labels are not required during make-before-break (MBB) in scenarios where the new LSP instance traverses the exact same path as the old LSP instance. This saves the ingress LER and the services that use the tunnel from needing to update the forwarding plane with new tunnel labels and so makes MBB events faster. Periodic MBB events are relatively common in networks that deploy the ‘auto-bandwidth’ feature on RSVP-TE LSPs to monitor bandwidth utilization and periodically adjust LSP bandwidth.

4. Mix and match labels: Both ‘TE link labels’ and regular labels can be used on transit hops for a single RSVP-TE tunnel (see Section 6). This allows backward compatibility with transit LSRs that provide regular labels in Resv messages.

No additional extensions are required to routing protocols (IGP-TE) in order to support this shared MPLS forwarding plane. Functionalities such as bandwidth admission control, LSP priorities, preemption, auto-bandwidth and Fast Reroute continue to work with this forwarding plane.
The signaling procedures and extensions discussed in this document do not apply to Point to Multipoint (P2MP) RSVP-TE Tunnels.

2. Terminology

The following terms are used in this document:

- **TE link label**: An incoming label at an LSR that will be popped by the LSR with the packet being forwarded over a specific outgoing TE link to a neighbor.

- **Shared MPLS forwarding plane**: An MPLS forwarding plane where every participating LSR uses TE link labels on every LSP.

- **Segment Routed RSVP-TE tunnel**: An MPLS RSVP-TE tunnel that requests the use of a shared MPLS forwarding plane at every hop of the LSP.

3. Allocation of TE Link Labels

An LSR that participates in a shared MPLS forwarding plane MUST allocate a unique TE link label for each TE link. When an LSR encounters a TE link label at the top of the label stack it MUST pop the label and forward the packet over the TE link to the downstream neighbor on the RSVP-TE tunnel.

Multiple TE link labels MAY be allocated for the TE link to accommodate tunnels requesting no protection, link-protection and node-protection over the specific TE link.

Implementations that maintain per label bandwidth accounting at each hop must aggregate the reservations made for all the LSPs using the shared TE link label.

4. Segment Routed RSVP-TE Tunnel Setup

This section provides an example of how the RSVP-TE signaling procedure works to set up a tunnel utilizing a shared MPLS forwarding plane. The sample topology below is used to explain the example. Labels shown at each node are TE link labels that, when present at the top of the label stack, indicate that they should be popped and that the packet should be forwarded on the TE link to the neighbor.
Consider two tunnels:

RSVP-TE tunnel T1: From A to E on path A-B-C-D-E

RSVP-TE tunnel T2: From F to E on path F-B-C-D-E

Both tunnels share the TE links B-C, C-D, and D-E.

RSVP-TE is used to signal the setup of tunnel T1 (using the TE link label attributes flag defined in Section 10.2). When LSR D receives the Resv message from the egress LER E, it checks the next-hop TE link (D-E) and provides the TE link label (250) in the Resv message for the tunnel placing the label value in the Label object and also in the Label subobject carried in the RRO and setting the TE link label flag as defined in Section 10.3.

Similarly, LSR C provides the TE link label (200) for the TE link C-D, and LSR B provides the TE link label (150) for the TE link B-C.

For tunnel T2, the transit LSRs provide the same TE link labels as described for tunnel T1 as the links B-C, C-D, and D-E are common between the two LSPs.

The ingress LERs (A and F) will push the same stack of labels (from top of stack to bottom of stack) (150, 200, 250) for tunnels T1 and T2 respectively.

It should be noted that a transit LSR does not swap the top TE link label on an incoming packet (the label that it advertised in the Resv message it sent). All it has to do is pop the top label and forward the packet.

The values in the Label subobjects in the RRO are of interest to the ingress LERs in order to construct the stack of labels to impose on the packets.
If, in this example, there was another RSVP-TE tunnel T3 from F to I on path F-B-C-D-E-I, then this would also share the TE links B-C, C-D, and D-E and additionally traverse link E-I. The label stack used by F would be (150, 200, 250, 850). Hence, regardless of the ingress and egress LERs from where the LSPs start and end, they will share LSR labels at shared hops in the shared MPLS forwarding plane.

There MAY be local operator policy at the ingress LER that influences the maximum depth of the label stack that can be pushed for a Segment Routed RSVP-TE tunnel. Prior to signaling the LSP, the ingress LER may decide that it would be unable to push a label stack containing one label for each hop along the path. In this case the LER can choose either to not signal a Segment Routed RSVP-TE tunnel (using normal LSP signaling instead), or can adopt the techniques described in Section 5 or Section 6.

5. Delegating Label Stack Imposition

One or more transit LSRs can assist the ingress LER by imposing part of the label stack required for the path. Consider the example in Figure 2 with an RSVP-TE tunnel from A to L on path A-B-C-D-E-F-G-H-I-J-K-L. In this case, the LSP is too long for LER A to impose the full label stack, so it uses the assistance of delegation hops LSR D and LSR I to impose parts of the label stack.

Each delegation hop allocates a delegation label to represent a set of labels that will be pushed at this hop. When a packet arrives at a delegation hop LSR with a delegation label, the LSR pops the label and pushes a set of labels before forwarding the packet.

```
1250d
+----+100p  +----+150p  +----+200p  +----+250p  +----+300p  ++++
| A |------| B |------| C |------| D |------| E |------| F |
+----+      +----+      +----+      +----+      +----+      +----+
   +----+  +----+  +----+  +----+  +----+  +----+  +----+  +----+  +----+
| 350p |
```

```
1500d
+----+  +----+  +----+  +----+  +----+  +----+  +----+  +----+  +----+  +----+  +----+
| L |------| K |------| J |------| I |------| H |------| G +
```

Notation : <Label>p - TE link label
<Label>d - delegation label

Figure 2: Delegating Label Stack Imposition
5.1. Stacking at the Ingress

When delegation labels come into play, there are two stacking approaches that the ingress can choose from. Section 7 explains how the label stack can be constructed.

5.1.1. Stack to Reach Delegation Hop

In this approach, the stack pushed by the ingress carries a set of labels that will take the packet to the first delegation hop. When this approach is employed, the set of labels represented by a delegation label at a given delegation hop will include the corresponding delegation label from the next delegation hop. As a result, this delegation label can only be shared among LSPs that are destined to the same egress and traverse the same downstream path.

This approach is shown in Figure 3. The delegation label 1250 represents the stack {300, 350, 400, 450, 1500} and the delegation label 1500 represents the label stack {550, 600}.

```
| A |-----------------| D |-----------------| I |
+---+               +---+               +---+
Push +---+           Push +---+           Push +---+
......           ......           ......           ......           ......           ......           ......
: 150: 1250->: 300: 1500->: 550:
: 200:   : 350:   : 600:
:1250:   : 400:   ......           ......           ......           ......           ......           ......

Figure 3: Stack to Reach Delegation Hop
```

With this approach, the ingress LER A will push {150, 200, 1250} for the tunnel in Figure 2. At LSR D, the delegation label 1250 will get popped and {300, 350, 400, 450, 1500} will get pushed. And at LSR I, the delegation label 1500 will get popped and the remaining set of labels {550, 600} will get pushed.
5.1.2. Stack to Reach Egress

In this approach, the stack pushed by the ingress carries a set of labels that will take the packet all the way to the egress so that all the delegation labels are part of the stack. When this approach is employed, the set of labels represented by a delegation label at a given delegation hop will not include the corresponding delegation label from the next delegation hop. As a result, this delegation label can be shared among all LSPs traversing the segment between the two delegation hops.

The downside of this approach is that the number of hops that the LSP can traverse is dictated by the label stack push limit of the ingress.

This approach is shown in Figure 4. The delegation label 1250 represents the stack \{300, 350, 400, 450\} and the delegation label 1500 represents the label stack \{550, 600\}.

```
+----+               +----+               +----+
| A |-------.....-----| D |-------.....-----| I |-------.....
+----+               +----+               +----+

Pop 1250 &          Push
......              ......              ......              ......              ......              ......              ..... 1500:
: 150:          1250->: 300:          1500->: 550:
: 200:              : 350:              : 600:
:1250:              : 400:              ......              ......              ......              ......              ......              ......              ......              ...... 1500:
:1500:              : 450:              ......              ......              ......              ......              ...... 1500:
......              ......              ......              ......              ......              ......              ...... 1500:

Figure 4: Stack to reach egress
```

With this approach, the ingress LER A will push \{150, 200, 1250, 1500\} for the tunnel in Figure 2. At LSR D, the delegation label 1250 will get popped and \{300, 350, 400, 450\} will get pushed. And at LSR I, the delegation label 1500 will get popped and the remaining set of labels \{550, 600\} will get pushed. The signaling extension required for the ingress to indicate the chosen stacking approach is defined in Section 10.6.
5.2. Explicit Delegation

In this delegation option, the ingress LER can explicitly delegate one or more specific transit LSRs to handle pushing labels for a certain number of their downstream hops. In order to accurately pick the delegation hops, the ingress needs to be aware of the label stack depth push limit of each of the transit LSRs prior to initiating the signaling sequence. The mechanism by which the ingress or controller (hosting the path computation element) learns this information is outside the scope of this document.

The signaling extension required for the ingress LER to explicitly delegate one or more specific transit hops is defined in Section 10.4. The extension required for the delegation hop to indicate that the recorded label is a delegation label is defined in Section 10.5.

5.3. Automatic Delegation

In this approach, the ingress LER lets the downstream LSRs automatically pick suitable delegation hops during the initial signaling sequence. The ingress does not need to be aware up front of the label stack depth push limit of each of the transit LSRs. The delegation hops are picked based on a per-hop signaled attribute called the Effective Transport Label-Stack Depth (ETLD) as described in the next section.

5.3.1. Effective Transport Label-Stack Depth (ETLD)

The ETLD is signaled as a per-hop attribute in the Path message [RFC7570]. When automatic delegation is requested, the ingress MUST populate the ETLD with the maximum number of transport labels that it can potentially send to its downstream hop. This value is then decremented at each successive hop. If a node is reached where the ETLD set from the previous hop is 1, then that node MUST select itself as the delegation hop. If a node is reached and it is determined that this hop cannot receive more than one transport label, then that node MUST select itself as the delegation hop. If there is a node or a sequence of nodes along the path of the LSP that do not support ETLD, then the immediate hop that supports ETLD MUST select itself as the delegation hop. The ETLD MUST be decremented at each non-delegation transit hop by either 1 or some appropriate number based on the limitations at that hop. At each delegation hop, the ETLD MUST be reset to the maximum number of transport labels that the hop can send and the ETLD decrements start again at each successive hop until either a new delegation hop is selected or the egress is reached. The net result is that by the time the Path message reaches the egress, all delegation hops are selected. During
the Resv processing, at each delegation hop, a suitable delegation label is selected (either an existing label is reused or a new label is allocated) and recorded in the Resv message.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Consider the example shown in Figure 5. Let’s assume ingress LER A can push up to 3 transport labels while the remaining nodes can push up to 5 transport labels. The ingress LER A signals the initial Path message with ETLD set to 3. The ETLD value is adjusted at each successive hop and signaled downstream as shown. By the time the Path message reaches the egress LER L, LSRs D and I are automatically selected as delegation hops.

Figure 5: ETLD

Signaling extension for the ingress LER to request automatic delegation is defined in Section 10.4. The extension for signaling the ETLD is defined in Section 10.7. The extension required for the delegation hop to indicate that the recorded label is a delegation label is defined in Section 10.5.

6. Mixing TE Link Labels and Regular Labels in an RSVP-TE Tunnel

Labels can be mixed across transit hops in a single MPLS RSVP-TE LSP. Certain LSRs can use TE link labels and others can use regular labels. The ingress can construct a label stack appropriately based on what type of label is recorded from every transit LSR.
Figure 6: Sample Topology - TE Link Labels and Regular Labels

If the transit LSR allocates a regular label to be sent upstream in the Resv, then the label operation at the LSR is a swap to the label received from the downstream LSR. If the transit LSR is using a TE link label to be sent upstream in the Resv, then the label operation at the LSR is a pop and forward regardless of any label received from the downstream LSR. There is no change in the behavior of a penultimate hop popping (PHP) LSR [RFC3031].

Section 7 explains how the label stack can be constructed. For example, the LSP from A to I using path A-B-C-D-E-I will use a label stack of (150, 200).

7. Construction of Label Stacks

The ingress LER or delegation hop MUST check the type of label received from each transit hop as recorded in the RRO in the Resv message and generate the appropriate label stack to reach the next delegation hop or the egress.

The following logic could be used by the node constructing the label stack:

Each RRO label sub-object SHOULD be processed starting with the label sub-object from the first downstream hop. Any label provided by the first downstream hop MUST always be pushed on the label stack regardless of the label type. If the label type is a TE link label, then any label from the next downstream hop MUST also be pushed on the constructed label stack. If the label type is a regular label, then any label from the next downstream hop MUST NOT be pushed on the constructed label stack. If the label type is a delegation label, then the stacking procedure stops at...
that delegation hop. Approaches in Section 5.1 SHOULD be used to
determine how the delegation labels are pushed in the label stack.

8. Facility Backup Protection

The following section describe how link and node protection works
with facility backup protection [RFC4090] for the Segment Routed
RSVP-TE tunnels.

8.1. Link Protection

To provide link protection at a Point of Local Repair (PLR) with a
shared MPLS forwarding plane, the LSR SHOULD allocate a separate TE
link label for the TE link that will be used for RSVP-TE tunnels that
request link-protection from the ingress. No signaling extensions
are required to support link protection for RSVP-TE tunnels over the
shared MPLS forwarding plane.

At each LSR, link protected TE link labels can be allocated for each
TE link and a link protecting facility backup LSP can be created to
protect the TE link. The link protected TE link label can be sent by
the LSR for LSPs requesting link-protection over the specific TE
link. Since the facility backup terminates at the next-hop (merge
point), the incoming label on the packet will be what the merge point
expects.

Consider the network shown in Figure 7. LSR B can install a facility
backup LSP for the link protected TE link label 151. When the TE
link B-C is up, LSR B will pop 151 and send the packet to C. If the
TE link B-C is down, the LSR can pop 151 and send the packet via the
facility backup to C.

```
     101(*)  151(*)  201(*)  251(*)
  +----+100----+----+150----+----+200----+----+250----+----+
  | A |------| B |------| C |------| D |------| E |
  +----+      +----+      +----+      +----+      +----+
  |110       |450       |550       |650       |850 |
  |          |400       |500       |600       |800 |
  +--------| F |------|G  |------|H  |------|I  |
             +----+      +----+      +----+      +----+
             |+++=300----+----+350----+----+700----+----+
```

Notation : (*) denotes link protection TE link labels

Figure 7: Link Protection Topology
8.2. Node Protection

The solutions for the PLR to provide node-protection for the Segment Routed RSVP-TE tunnel will be explained in a future version of this document.

9. Quantifying TE Link Labels

This section quantifies the number of labels required in the forwarding plane to provide sharing of labels across Segment Routed RSVP-TE tunnels. An MPLS RSVP-TE tunnel offers either no protection, link protection, or node protection and only one of these labels is required per tunnel during signaling. The scale of the number of TE link labels required per LSR can be deduced as follows:

- For an LSR having X neighbors reachable across Y interfaces, the number of unprotected TE link labels is X.
- For a PLR having X neighbors reachable across Y interfaces, the number of link protected TE link labels is X.
- For a PLR having X neighbors, each having Nx neighbors (i.e. next-nexthops for the PLR), number of node protected TE link labels is \( \sum_{\text{ALL}}(Nx) \).

The total number of TE link labels is given by:

\[
\text{Unprotected TE link labels} + \\
\text{link protected TE link labels} + \\
\text{node protected TE link labels} = 2X + \sum_{\text{ALL}}(Nx)
\]

10. Protocol Extensions

10.1. Requirements

The functionality discussed in this document imposes the following requirements on the signaling protocol.

- The Ingress of the LSP SHOULD have the ability to mandate/request the use and recording of TE link labels at all hops along the path of the LSP.
- When the use of TE link labels is mandated/requested for the path:
  * the node recording the TE link label SHOULD have the ability to indicate if the recorded label is a TE link label.
* the ingress SHOULD have the ability to delegate label stack imposition by:
  + explicitly mandating specific hops to be delegation hops (or)
  + requesting automatic delegation.
* When explicit delegation is mandated or automatic delegation is requested:
  + the ingress SHOULD have the ability to indicate the chosen stacking approach (and)
  + the delegation hop SHOULD have the ability to indicate that the recorded label is a delegation label.

10.2. Attribute Flags TLV: TE Link Label

Bit Number (TBD1): TE Link Label

The presence of this in the LSP_ATTRIBUTES/LSP_REQUIRED_ATTRIBUTES object of a Path message indicates that the ingress has requested/mandated the use and recording of TE link labels at all hops along the path of this LSP. When a node that does not cater to the mandate receives a Path message carrying the LSP_REQUIRED_ATTRIBUTES object with this flag set, it MUST send a PathErr message with an error code of ‘routing problem’ and an error value of ‘TE link label usage failure’.

10.3. RRO Label Subobject Flag: TE Link Label

Bit Number (TBD2): TE Link Label

The presence of this flag indicates that the recorded label is a TE link label. This flag MUST be used by a node only if the use and recording of TE link labels is requested/mandated for the LSP.

10.4. Attribute Flags TLV: LSI-D

Bit Number (TBD3): Label Stack Imposition - Delegation (LSI-D)

Automatic Delegation: The presence of this flag in the LSP_ATTRIBUTES object of a Path message indicates that the ingress has requested automatic delegation of label stack imposition. This flag MUST be set in the LSP_ATTRIBUTES object of a Path message only if the use and recording of TE link labels is requested/mandated for this LSP.
Explicit Delegation: The presence of this flag in the HOP_ATTRIBUTES subobject [RFC7570] of an ERO object in the Path message indicates that the hop identified by the preceding IPv4 or IPv6 or Unnumbered Interface ID subobject has been picked as an explicit delegation hop. The HOP_ATTRIBUTES subobject carrying this flag MUST have the R (Required) bit set. This flag MUST be set in the HOP_ATTRIBUTES subobject of an ERO object in the Path message only if the use and recording of TE link labels is requested/mandated for this LSP. If the hop is not able to comply with this mandate, it MUST send a PathErr message with an error code of 'routing problem' and an error value of 'label stack imposition failure'.

10.5. RRO Label Subobject Flag: Delegation Label

Bit Number (TBD4): Delegation Label

The presence of this flag indicates that the recorded label is a delegation label. This flag MUST be used by a node only if the use and recording of TE link labels and delegation are requested/mandated for the LSP.

10.6. Attributes Flags TLV: LSI-D-S2E

Bit Number (TBD5): Label Stack Imposition - Delegation - Stack to reach egress (LSI-D-S2E)

The presence of this flag in the LSP_ATTRIBUTES object of a Path message indicates that the ingress has chosen to use the "Stack to reach egress" approach for stacking. The absence of this flag in the LSP_ATTRIBUTES object of a Path message indicates that the ingress has chosen to use the "Stack to reach delegation hop" approach for stacking. This flag MUST be set in the LSP_ATTRIBUTES object of a Path message only if the use and recording of TE link labels and delegation are requested/mandated for this LSP.

10.7. Attributes TLV: ETLD

The format of the ETLD Attributes TLV is shown in Figure 8. The Attribute TLV Type is TBD6.
The presence of this TLV in the HOP_ATTRIBUTES subobject of an RRO object in the Path message indicates that the hop identified by the preceding IPv4 or IPv6 or Unnumbered Interface ID subobject supports automatic delegation. This attribute MUST be used only if the use and recording of TE link labels is requested/mandated and automatic delegation is requested for the LSP. The ETLD field specifies the maximum number of transport labels that this hop can potentially send to its downstream hop.

11. OAM Considerations

MPLS LSP ping and traceroute [RFC8029] are applicable for Segment Routed RSVP-TE tunnels. The existing procedures allow for the label stack imposed at a delegation hop to be reported back in the Label Stack Sub-TLV in the MPLS echo reply for traceroute.

12. Acknowledgements

The authors would like to thank Adrian Farrel, Kireeti Kompella, Markus Jork and Ross Callon for their input from discussions.

Adrian Farrel provided a review and text suggestion for clarity and readability.

13. Contributors

The following individuals contributed to this document:

Raveendra Torvi
Juniper Networks
Email: rtorvi@juniper.net

Chandra Ramachandran
Juniper Networks
Email: csekar@juniper.net

George Swallow
Email: swallow.ietf@gmail.com
14. IANA Considerations

14.1. Attribute Flags: TE Link Label, LSI-D, LSI-D-S2E


<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Name</th>
<th>Attribute FlagsPath</th>
<th>Attribute FlagsResv</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD1</td>
<td>TE Link Label</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TBD3</td>
<td>LSI-D</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TBD5</td>
<td>LSI-D-S2E</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

14.2. Attribute TLV: ETLD

IANA manages the "Attribute TLV Space" registry as part of the ‘Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Parameters’ registry located at http://www.iana.org/assignments/rsvp-te-parameters. This document introduces a new Attribute TLV.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Allowed on LSP</th>
<th>Allowed on LSP REQUIRED</th>
<th>Allowed on LSP Hop</th>
<th>Allowed on Attributes</th>
<th>Allowed on Attributes</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD6</td>
<td>ETLD</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>This document</td>
<td>(Section 11.7)</td>
</tr>
</tbody>
</table>

14.3. Record Route Label Sub-object Flags: TE Link Label, Delegation Label

IANA manages the ‘Record Route Object Sub-object Flags’ registry as part of the ‘Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Parameters’ registry located at http://www.iana.org/assignments/rsvp-te-parameters. This registry currently does not include Label Sub-object Flags. This document requests the addition of a new sub-regISTRY for Label Sub-object Flags as shown below.
15. Security Considerations

This document does not introduce new security issues. The security considerations pertaining to the original RSVP protocol [RFC2205] and RSVP-TE [RFC3209] and those that are described in [RFC5920] remain relevant.

16. References

16.1. Normative References


16.2. Informative References


Service Chaining using Unified Source Routing Instructions
draft-xu-mpls-service-chaining-03

Abstract

Source Packet Routing in Networking (SPRING) WG is developing an MPLS source routing mechanism. The MPLS source routing mechanism can be leveraged to realize a unified source routing instruction which works across both IPv4 and IPv6 underlays in addition to the MPLS underlay. This document describes how to leverage the unified source routing instruction to realize a transport-independent service function chaining by encoding the service function path information or service function chain information as an MPLS label stack.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on December 31, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction ............................................. 2
2. Terminology ............................................. 3
3. Solution Description ..................................... 3
   3.1. Encoding SFP Information by an MPLS Label Stack .... 4
   3.2. Encoding SFC Information by an MPLS Label Stack .... 7
   3.3. How to Contain Metadata within an MPLS Packet .... 9
4. Acknowledgements ......................................... 10
5. IANA Considerations ..................................... 10
6. Security Considerations .................................. 10
7. References ............................................... 10
   7.1. Normative References ............................... 10
   7.2. Informative References ............................. 10
Authors’ Addresses ......................................... 11

1. Introduction

When applying a particular Service Function Chain (SFC) [RFC7665] to the traffic selected by a service classifier, the traffic need to be steered through an ordered set of Service Functions (SF) in the network. This ordered set of SFs in the network indicates the Service Function Path (SFP) associated with the above SFC. In order
to steer the selected traffic through the required ordered list of SFs, the service classifier needs to attach information to the packet specifying exactly which Service Function Forwarders (SFFs) and which SFs are to be visited by traffic), the SFC, or the partially specified SFP which is in between the former two extremes.

The Source Packet Routing in Networking (SPRING) WG is developing an MPLS source routing mechanism which can be used to steer traffic through an ordered set of routers (i.e., an explicit path) and instruct nodes on that path to execute specific operations on the packet. By leveraging the MPLS source routing mechanism, [I-D.xu-mpls-unified-source-routing-instruction] describes a unified source routing instruction which works across both IPv4 and IPv6 underlays in addition to the MPLS underlay. This document describes how to leverage the unified source routing instruction to realize a transport-independent service function chaining by encoding the service function path information or service function chain information as an MPLS label stack.

2. Terminology

This memo makes use of the terms defined in [I-D.ietf-spring-segment-routing-mpls], [I-D.xu-mpls-unified-source-routing-instruction] and [RFC7665].

3. Solution Description

As shown in Figure 1, SFF1 and SFF2 are two MPLS-SPRING-capable nodes. They are also SFFs, each with one SF attached. In addition, they have allocated and advertised MPLS labels for their locally attached SFs. For example, SFF1 allocates and advertises a label (i.e., L(SF1)) for SF1 while SFF2 allocates and advertises a label (i.e., L(SF2)) for SF2. These labels, which are used to indicate SFs are referred to as SF labels. To encode the SFP information as an
MPLS label stack, local MPLS labels are allocated from SFFs' (e.g., SFF1 in Figure 1) label spaces to identify their locally attached SFs (e.g., SF1 in Figure 1), whilst the SFFs are identified by either nodal SIDs or adjacency SIDs depending on how strictly the network path needs to be specified. In addition, assume node SIDs for SFF1 and SFF2 are L(SFF1) and L(SFF2) respectively. In contrast, to encode the SFC information by an MPLS label stack, those SF labels MUST be domain-wide unique MPLS labels.

Now assume a given traffic flow destined for destination D is selected by the service classifier to go through a particular SFC (i.e., SF1-> SF2) before reaching its final destination D.  Section 3.1 and 3.2 describe approaches of leveraging the MPLS- based source routing mechanisms to realize the service function chaining by encoding the SFP information within an MPLS label stack and by encoding the SFC information within an MPLS label stack respectively. Since the encoding of the partially specified SFP is just a simple combination of the encoding of the SFP and the encoding of the SFC, this document would not describe how to encode the partially specified SFP anymore.

3.1. Encoding SFP Information by an MPLS Label Stack
As shown in Figure 2, since the selected packet needs to travel through an SFC (i.e., SF1->SF2), the service classifier would attach a segment list of (i.e., SID(SFF1)->SID(SF1)->SID(SFF2)->SID(SF2)) which indicates the corresponding SFP to the packet. This segment list is represented by an MPLS label stack. To some extent, the MPLS label stack here could be looked as a specific implementation of the SFC encapsulation used for containing the SFP information [RFC7665].

When the encapsulated packet arrives at SFF1, SFF1 would know which SF should be performed according to the top label (i.e., SID (SF1)) of the received MPLS packet. We first consider the case where SF1 is an encapsulation aware SF, i.e., it understands how to process a packet with a pre-pended MPLS label stack. In this case the packet would be sent to SF1 by SFF1 with the label stack SID(SFF2)->SID(SF2). SF1 would perform the required service function on the received MPLS packet where the payload is constrained to be an IP packet, and the SF needs to process both IPv4 and IPv6 packets (note that the SF would use the first nibble of the MPLS payload to
identify the payload type). After the MPLS packet is returned from SF1, SFF1 would send it to SFF2 according to the top label (i.e., SID (SFF2)).

If SF1 is a legacy SF, i.e., one that is unable to process the MPLS label stack, the remaining MPLS label stack (i.e., SID (SFF2) -> SID (SF2)) MUST be saved and stripped from the packet before sending the packet to SF1. When the packet is returned from SF1, SFF1 would re-impose the MPLS label stack which had been previously stripped and then send the packet to SFF2 according to the current top label (i.e., SID (SFF2)). As for how to associate the corresponding MPLS label stack with the packets returned from legacy SFs, those mechanisms as described in [I-D.song-sfc-legacy-sf-mapping] could be considered.

When the encapsulated packet arrives at SFF2, SFF2 would perform the similar action to that described above.

As shown in Figure 3, if there is no MPLS LSP towards the next node segment (i.e., the next SF identified by the current top label), the corresponding IP-based tunnel for MPLS (e.g., MPLS-in-IP/GRE tunnel [RFC4023], MPLS-in-UDP tunnel [RFC7510] or MPLS-in-L2TPv3 tunnel [RFC4817]) would be used instead, according to the unified source routing instruction as described in [I-D.xu-mpls-unified-source-routing-instruction].
Since the transport (i.e., the underlay) could be IPv4, IPv6 or even MPLS networks, the above approach of encoding the SFP information by an MPLS label stack is fully transport-independent which is one of the major requirements for the SFC encapsulation [RFC7665].

3.2. Encoding SFC Information by an MPLS Label Stack

Since the selected packet needs to travel through an SFC (i.e., SF1->SF2), the service classifier would attach an MPLS label stack (i.e., L(SF1)->L(SF2)) which indicates that SFC to the packet. Since it’s known to the service classifier that SFF1 is attached with an instance of SF1, the service classifier would therefore send the MPLS encapsulated packet through either an MPLS LSP tunnel or an IP-based tunnel towards SF1 (as shown in Figure 4 and 5 respectively). When the MPLS encapsulated packet arrives at SFF1, SFF1 would know which SF should be performed according to the current top label (i.e.,
Similarly, SFF1 would send the packet returned from SF1 to SFF2 through either an MPLS LSP tunnel or an IP-based tunnel towards SFF2 since it’s known to SFF1 that SFF2 is attached with an instance of SF2. When the encapsulated packet arrives at SFF2, SFF2 would do the similar action as what has been done by SFF1. Since the transport (i.e., the underlay) could be IPv4, IPv6 or even MPLS networks, the above approach of encoding the SFC information by an MPLS label stack is fully transport-independent which is one of the major requirements for the SFC encapsulation [RFC7665].

Figure 4: Packet Walk in MPLS underlay
3.3. How to Contain Metadata within an MPLS Packet

Since the MPLS encapsulation has no explicit protocol identifier field to indicate the protocol type of the MPLS payload, how to indicate the presence of metadata (i.e., the NSH which is only used as a metadata container) in an MPLS packet is a potential issue to be addressed. One possible way to address the above issue is: SFFs allocate two different labels for a given SF, one indicates the presence of NSH while the other indicates the absence of NSH. This approach has no change to the current MPLS architecture but it would require more than one label binding for a given SF. Another possible way is to introduce a protocol identifier field within the MPLS packet as described in [I-D.xu-mpls-payload-protocol-identifier].

More details about how to contain metadata within an MPLS packet would be considered in the future version of this draft.

Figure 5: Packet Walk in IP underlay
4. Acknowledgements

The authors would like to thank Loa Andersson, Andrew G. Malis, Adrian Farrel, Alexander Vainshtein and Joel M. Halpern for their valuable comments and suggestions on the document.

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

It is fundamental to the SFC design that the classifier is a trusted resource which determines the processing that the packet will be subject to, including for example the firewall. It is also fundamental to the SPRING design that packets are routed through the network using the path specified by the node imposing the SIDs. Where an SF is not encapsulation aware the packet may exist as an IP packet, however this is an intrinsic part of the SFC design which needs to define how a packet is protected in that environment. Where a tunnel is used to link two non-MPLS domains, the tunnel design needs to specify how it is secured. Thus the security vulnerabilities are addressed in the underlying technologies used by this design, which itself does not introduce any new security vulnerabilities.

7. References

7.1. Normative References


7.2. Informative References


[I-D.song-sfc-legacy-sf-mapping]  

[I-D.xu-mpls-payload-protocol-identifier]  

[I-D.xu-mpls-unified-source-routing-instruction]  

[RFC4023]  

[RFC4817]  

[RFC7510]  

[RFC7665]  

Authors' Addresses

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com
Stewart Bryant
Huawei
Email: stewart.bryant@gmail.com

Hamid Assarpour
Broadcom
Email: hamid.assarpour@broadcom.com

Himanshu Shah
Ciena
Email: hshah@ciena.com

Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid, 28050
Spain
Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://people.tid.es/LuisM.Contreras/

Daniel Bernier
Bell Canada
Email: daniel.bernier@bell.ca

Jeff Tantsura
Individual
Email: jefftant@gmail.com

Shaowen Ma
Juniper
Email: mashaowen@gmail.com
Unified Source Routing Instructions using MPLS Label Stack
draft-xu-mpls-unified-source-routing-instruction-04

Abstract

MPLS Segment Routing (SR-MPLS in short) is an MPLS data plane-based source routing paradigm in which a sender of a packet is allowed to partially or completely specify the route the packet takes through the network by imposing stacked MPLS labels to the packet. SR-MPLS could be leveraged to realize a unified source routing mechanism across MPLS, IPv4 and IPv6 data planes by using an MPLS label stack as a unified source routing instruction set while preserving backward compatibility with SR-MPLS.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 1, 2018.
1. Introduction

MPLS Segment Routing (SR-MPLS in short) [I-D.ietf-spring-segment-routing-mpls] is an MPLS data plane-based source routing paradigm in which a sender of a packet is allowed to partially or completely specify the route the packet takes through the network by imposing stacked MPLS labels to the packet. SR-MPLS could be leveraged to realize a unified source routing mechanism across MPLS, IPv4 and IPv6 data planes by using an MPLS label stack as a unified source routing instruction set while preserving backward compatibility with SR-MPLS. More specifically, the source routing instruction set information contained in a source routed packet could be uniformly encoded as an MPLS label stack no matter the underlay is IPv4, IPv6 or MPLS.
Although the source routing instructions are encoded as MPLS labels, this is a hardware convenience rather than an indication that the whole MPLS protocol stack and in particular the MPLS control protocols need to be deployed. Note that the complexity associated with the whole MPLS protocol stack is largely due to the complex control plane protocols.

Section 3 describes various use cases for the unified source routing instruction mechanism and Section 4 describes a typical application scenario and how the packet forwarding happens.

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].

2. Terminology

This memo makes use of the terms defined in [RFC3031] and [I-D.ietf-spring-segment-routing-mpls].

3. Use Cases

The unified source routing mechanism across IPv4, IPv6 and MPLS is useful at least in the following use cases:

- Incremental deployment of the SR-MPLS technology [I-D.xu-mpls-spring-islands-connection-over-ip]. Since there is no need to run any other label distribution protocol (e.g., LDP, see [I-D.ietf-spring-segment-routing-ldp-interop] for more details.) on those non-SR-MPLS routers for incremental deployment purposes, the network provisioning is greatly simplified, which is one of the major claimed benefits of the SR-MPLS technology (i.e., running a single protocol).

- Overcome the load-balancing dilemma encountered by SR-MPLS. In fact, this unified source routing mechanism is even useful in a fully upgraded SR-MPLS network since the load-balancing dilemma encountered by SR-MPLS [I-D.ietf-mpls-spring-entropy-label] due to the maximum Readable Label-stack Depth (RLD) hardware limitation [I-D.ietf-ospf-mpls-elsc] [I-D.ietf-isis-mpls-elsc] [I-D.ietf-idr-bgp-ls-segment-routing-rld] and the Maximum SID Depth (MSD) hardware limitation [I-D.ietf-ospf-segment-routing-msd] [I-D.ietf-isis-segment-routing-msd] [I-D.ietf-idr-bgp-1s-segment-routing-msd] by using the MPLS-in-UDP
encapsulation [RFC7510] where the source port of the UDP tunnel header is used as an entropy field.

- A poor man’s light-weight alternative to SRv6 [I-D.ietf-6man-segment-routing-header]. At least, it could be deployed as an interim until full featured SRv6 is available on more platforms. Since the Source Routing Header (SRH) [I-D.ietf-6man-segment-routing-header] consisting of an ordered list of 128-bit long IPv6 addresses is now replaced by an ordered list of 32-bit long label entries (i.e., label stack), the encapsulation overhead and forwarding performance issues associated with SRv6 are eliminated.

- A new IPv4 source routing mechanism which has overcome the security vulnerability issues associated with the traditional IPv4 source routing mechanism.

- Traffic Engineering scenarios where only a few routers (e.g., the entry and exit nodes of each plane in the dual-plane network case or the egress node in the Egress Peer Engineering (EPE) case) are specified as segments of explicit paths. In this way, only a few routers are required to support the SR-MPLS capability while all the other routers just need to support IP forwarding capability, which would significantly reduce the deployment cost of the SR-MPLS technology.

- MPLS-based Service Function Chaining (SFC) [I-D.xu-mpls-service-chaining]. Based on the unified source routing mechanism as described in this document, only SFC-related nodes including Service Function Forwarders (SFF), Service Functions (SF) and classifiers are required to recognize the SFC encapsulation header in the MPLS label stack form, while the intermediate routers just need to support vanilla IP forwarding (either IPv4 or IPv6). In other words, it undoubtedly complies with the transport-independence requirement for the SFC encapsulation header as listed in the SFC architecture document [RFC7665].

4. Packet Forwarding Procedures

The primary objective of this document is to describe how SR-MPLS capable routers and IP-only routers can seamlessly co-exist and interoperate. This section describes the forwarding information base (FIB) entry and the forwarding behavior that allow the deployment of SR-MPLS when some routers are IPv4 only or IPv6 only. Note that OSPF or ISIS is assumed to be enabled in the following examples as described in Section 4.1 and 4.2, in fact, it’s no doubt that BGP could be used as a replacement.
4.1. Forwarding Entry Construction

This sub-section describes the how to construct the forwarding information base (FIB) entry on an SR-MPLS-capable router when some or all of the next-hops along the shortest path towards a prefix-SID are IPv4-only or IPv6-only routers. Consider the router "A" receiving a labeled packet whose top label L(E) corresponds to the prefix-SID is "SID(E)" of prefix "P(E)" advertised by the router "E". Suppose the ith next-hop router "NHi" along the shortest path from the router "A" towards the prefix-SID "SID(E)" is not SR-MPLS capable. That is both routers "A" and "E" are SR-MPLS capable but the next hop "NHi" along the shortest path from "A" to "E". The following applies:

- It is assumed that the router "E" advertises the SR-Capabilities sub-TLV as described in and [I-D.ietf-ospf-segment-routing-extensions], which includes the SRGB because router "E" is SR-MPLS capable.

- The owning router "E" MUST advertise the encapsulation endpoint and the tunnel type using [I-D.ietf-isis-encapsulation-cap] and/or [I-D.ietf-ospf-encapsulation-cap].

- If "A" and "E" are in different areas/levels, then
  - The OSPF Tunnel Encapsulation TLV [I-D.ietf-ospf-encapsulation-cap] and/or the ISIS Tunnel Encapsulation sub-TLV [I-D.ietf-isis-encapsulation-cap] are flooded domain-wide.
  - The OSPF SID/label range TLV [I-D.ietf-ospf-segment-routing-extensions] and the ISIS SR-Capabilities Sub-TLV [I-D.ietf-isis-segment-routing-extensions] are advertised domain-wide. This way router "A" knows the characteristics of the owning router "E".
  - When the owning router "E" is running ISIS and advertises the prefix "P(E) ", the router "E" uses the extended reachability TLV (TLVs 135, 235, 236, 237) and associates the IPv4/IPv6 and/or IPv4/IPv6 source router ID sub-TLV(s) [RFC7794].
  - When the owning router "E" is running OSPF and advertises the prefix "P(E) ", the router "E" uses the OSPFv2 Extended Prefix Opaque LSA [RFC7684] and sets the flooding scope to AS-wide.
  - When the owning router "E" is running ISIS and advertises the ISIS capabilities TLV (TLV 242) [RFC7981], it must set the "router-ID" field to a valid value or include IPv6 TE router-
ID sub-TLV (TLV 12), or do both. The "S" bit (flooding scope) of the ISIS capabilities TLV (TLV 242) MUST be set to "1".

- Router "A" programs the FIB entry corresponding to the "SID(E)" as follows:

  * If NP (OSPF) or P (ISIS) flag is clear,
  *
    + pop the outer label.
  *
  * If NP (OSPF) or P (ISIS) is set,
  *
    + the outer label is SID(E) plus the lower bound of the SRGB of "E".

  * Encapsulate the packet according to the encapsulation advertised in [I-D.ietf-isis-encapsulation-cap] or [I-D.ietf-ospf-encapsulation-cap].

  * Send the packet towards the next hop "NHi".

4.2. Packet Forwarding Procedures

```
+-----+       +-----+       +-----+        +-----+        +-----+
|  A  +-------+  B  +-------+  C  +--------+  D  +--------+  H  |
+-----+       +--+--+       +--+--+        +--+--+        +-----+
                   |              |
                   |              |
                   +--+--+       +--+--+        +--+--+
                   |  E  +-------+  F  +--------+  G  |
                   +-----+       +-----+        +-----+
                                   +--------+
                                   |IP(A->E)|
                                   +--------+
                                   +--------+       +--------+
                                   | L(G) |       | IP(E->G)|
                                   +--------+       +--------+       +--------+
                                   | L(H) |       | L(H) |       | IP(G->H)|
                                   +--------+       +--------+       +--------+
                                   | Packet |  ---> | Packet |  ---> | Packet |
```

Figure 1
As shown in Figure 1, Assume Router A, E, G and H are SR-MPLS-capable routers while the remaining are only capable of forwarding IP packets. Router A, E, G and H advertise their Segment Routing related information via IS-IS or OSPF. Now assume router A wants to send a given IP or MPLS packet via an explicit path of \{E->G->H\}, router A would impose an MPLS label stack corresponding to that explicit path on the received IP packet. Since there is no Label Switching Path (LSP) towards router E, router A would replace the top label indicating router E with an IP-based tunnel for MPLS (e.g., MPLS-over-UDP [RFC7510]) towards router E and then send it out. In other words, router A would pop the top label and then encapsulate the MPLS packet with an IP-based tunnel towards router E. When the IP-encapsulated MPLS packet arrives at router E, router E would strip the IP-based tunnel header and then process the decapsulated MPLS packet accordingly. Since there is no LSP towards router G which is indicated by the current top label of the decapsulated MPLS packet, router E would replace the current top label with an IP-based tunnel towards router G and send it out. When the packet arrives at router G, router G would strip the IP-based tunnel header and then process the decapsulated MPLS packet. Since there is no LSP towards router H, router G would replace the current top label with an IP-based tunnel towards router H. Now the packet encapsulated with the IP-based tunnel towards router H is exactly the original packet that router A had intended to send towards router H. If the packet is an MPLS packet, router G could use any IP-based tunnel for MPLS (e.g., MPLS-over-UDP [RFC7510]). If the packet is an IP packet, router G could use any IP tunnel for IP (e.g., IP-in-UDP [I-D.xu-intarea-ip-in-udp]). That original IP or MPLS packet would be forwarded towards router H via an IP-based tunnel. When the encapsulated packet arrives at router H, router H would decapsulate it into the original packet and then process it accordingly.

Note that in the above description, it’s assumed that the label associated with each prefix-SID advertised by the owner of the prefix-SID is a Penultimate Hop Popping (PHP) label (e.g., the NP-flag [I-D.ietf-ospf-segment-routing-extensions] associated with the corresponding prefix SID is not set).

Figure 2 demonstrates the packet walk in the case where the label associated with each prefix-SID advertised by the owner of the prefix-SID is not a Penultimate Hop Popping (PHP) label (e.g., the NP-flag [I-D.ietf-ospf-segment-routing-extensions] associated with the corresponding prefix SID is set).
Although the above description is based on the use of prefix-SIDs, the unified source routing instruction approach is actually applicable to the use of adj-SIDs as well. For instance, when the top label of a received MPLS packet indicates a given adj-SID and the corresponding adjacent node to that adj-SID is not MPLS-capable, the top label would be replaced by an IP-based tunnel towards that adjacent node and then forwarded over the corresponding link indicated by that adj-SID.

When encapsulating an MPLS packet with an IP-based tunnel header (e.g., a UDP header as per [RFC7510]), the corresponding entropy field (i.e., the source port in the MPLS-in-UDP case) should be filled with an entropy value that is generated by the encapsulator to uniquely identify a flow. However, what constitutes a flow is locally determined by the encapsulator. For instance, if the MPLS label stack contains at least one entropy label and the encapsulator is capable of reading that entropy label, the entropy label value could be directly copied to the entropy field (e.g., the source port of the UDP header). Otherwise, the encapsulator may have to perform a hash on the whole label stack or the five-tuple of the MPLS payload if the payload is determined as an IP packet. To avoid re-performing hash on the whole packet when re-encapsulating the packet with an IP-based tunnel header (e.g., a UDP tunnel header), especially when the encapsulator could not obtain at least one entropy label due to some reasons (e.g., 1) there is no EL at all in the label stack; 2) the encapsulator couldn’t recognize the ELI; 3) the encapsulator could
not read the EL due to the RLD limit), it’s RECOMMENDED that the entropy value contained in the packet (e.g., the UDP source port value) is kept when stripping the IP-based tunnel header (e.g., the UDP tunnel header). As such, the entropy value could be directly copied to the entropy field (e.g., the source port of the UDP tunnel header) when re-encapsulating the packet with an IP-based tunnel header (e.g., a UDP tunnel header). As such, the load-balancing dilemma encountered by SR-MPLS as described in [I-D.ietf-mpls-spring-entropy-label] due to the maximum Readable Label-stack Depth (RLD) hardware limitation [I-D.ietf-ospf-mpls-elc] [I-D.ietf-isis-mpls-elc] and the Maximum SID Depth (MSD) hardware limitation [I-D.ietf-ospf-segment-routing-msd] [I-D.ietf-isis-segment-routing-msd] is gone. That’s the reason why this unified source routing mechanism is even useful in a fully upgraded SR-MPLS network environment.

5. Contributors

Clarence Filsfils
Cisco
Email: cfilsfils@cisco.com

Robert Raszuk
Bloomberg LP
Email: robert@raszuk.net

Uma Chunduri
Huawei
Email: uma.chunduri@gmail.com

Luis M. Contreras
Telefonica I+D
Email: luismiguel.contrerasmurillo@telefonica.com

Luay Jalil
Verizon
Email: luay.jalil@verizon.com

Gunter Van De Velde
Nokia
Email: gunter.van_de_velde@nokia.com

Tal Mizrahi
Marvell
Email: talmi@marvell.com
6. Acknowledgements

Thanks Joel Halpern, Bruno Decraene, Loa Andersson and Stewart Bryant for their insightful comments on this document.

7. IANA Considerations

No IANA action is required.

8. Security Considerations

TBD.

9. References

9.1. Normative References


9.2. Informative References


[I-D.ietf-isis-mpls-elc]

[I-D.ietf-isis-segment-routing-extensions]

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

[I-D.ietf-mpls-spring-entropy-label]

[I-D.ietf-ospf-encapsulation-cap]

[I-D.ietf-ospf-mpls-elc]

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

[I-D.ietf-ospf-segment-routing-msd]
[I-D.ietf-spring-segment-routing-ldp-interop]  
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., and  
S. Litkowski, "Segment Routing interworking with LDP",  
draft-ietf-spring-segment-routing-ldp-interop-08 (work in  
progress), June 2017.

[I-D.ietf-spring-segment-routing-mpls]  
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B.,  
Litkowski, S., and R. Shakir, "Segment Routing with MPLS  
data plane", draft-ietf-spring-segment-routing-mpls-10  
(work in progress), June 2017.

[I-D.xu-intarea-ip-in-udp]  
Xu, X., Lee, Y., and F. Yongbing, "Encapsulating IP in  
UDP", draft-xu-intarea-ip-in-udp-04 (work in progress),  
December 2016.

[I-D.xu-mpls-service-chaining]  
Xu, X., Bryant, S., Assarpour, H., Shah, H., Contreras,  
L., daniel.bernier@bell.ca, d., jefftant@gmail.com, j.,  
Ma, S., and M. Vigoureux, "Service Chaining using Unified  
Source Routing Instructions", draft-xu-mpls-service-  
chaining-03 (work in progress), June 2017.

[I-D.xu-mpls-spring-islands-connection-over-ip]  
Xu, X., Raszuk, R., Chunduri, U., Contreras, L., and L.  
Jalil, "Connecting MPLS-SPRING Islands over IP Networks",  
draft-xu-mpls-spring-islands-connection-over-ip-00 (work  
in progress), October 2016.

[RFC2784]  
Farinacci, D., Li, T., Hanks, S., Meyer, D., and P.  
Traina, "Generic Routing Encapsulation (GRE)", RFC 2784,  
DOI 10.17487/RFC2784, March 2000,  

[RFC3031]  
Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol  
Label Switching Architecture", RFC 3031,  
DOI 10.17487/RFC3031, January 2001,  

[RFC4817]  
Townsley, M., Pignataro, C., Wainner, S., Seely, T., and  
J. Young, "Encapsulation of MPLS over Layer 2 Tunneling  


Authors’ Addresses

Xiaohu Xu
Huawei
Email: xuxiaohu@huawei.com

Ahmed Bashandy
Cisco
Email: bashandy@cisco.com

Hamid Assarpour
Broadcom
Email: hamid.assarpour@broadcom.com
Internet-Draft  USR  September 2017

Shaowen Ma
Juniper

Email: mashao@juniper.net

Wim Henderickx
Nokia

Email: wim.henderickx@nokia.com

Jeff Tantsura
Individual

Email: jefftant@gmail.com