

MPLS Working Group
Internet-Draft
Intended status: Informational
Expires: December 28, 2017

S. Bryant
M. Chen
Z. Li
Huawei
G. Swallow
S. Sivabalan
Cisco Systems
G. Mirsky
Ericsson
June 26, 2017

Synonymous Flow Label Framework
draft-bryant-mpls-sfl-framework-05

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.

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. Requirements Language	2
3. Synonymous Flow Labels	3
4. User Service Traffic in the Data Plane	4
4.1. Applications Label Present	4
4.1.1. Setting TTL and the Traffic Class Bits	5
4.2. Single Label Stack	5
4.2.1. Setting TTL and the Traffic Class Bits	6
4.3. Aggregation of SFL Actions	6
5. Equal Cost Multipath Considerations	7
6. Privacy Considerations	8
7. Security Considerations	8
8. IANA Considerations	8
9. References	8
9.1. Normative References	8
9.2. Informative References	9
Authors' Addresses	9

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

14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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.

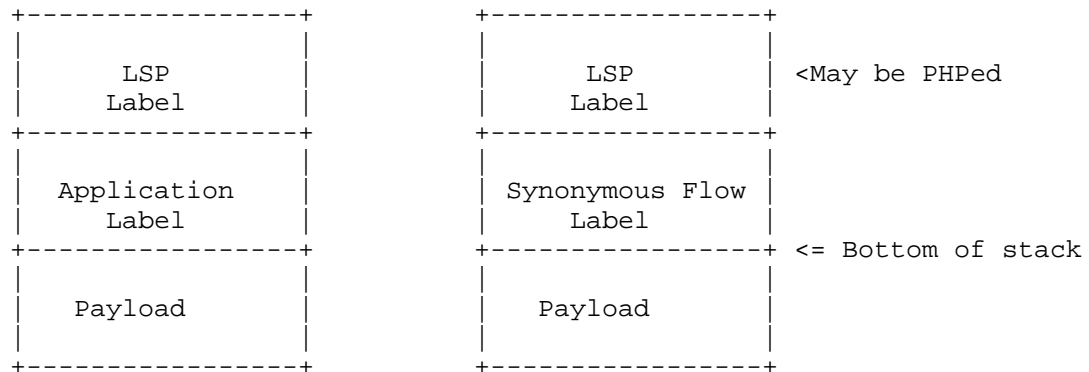
4. User Service Traffic in the Data Plane

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.



"Normal" Label Stack

Label Stack with SFL

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

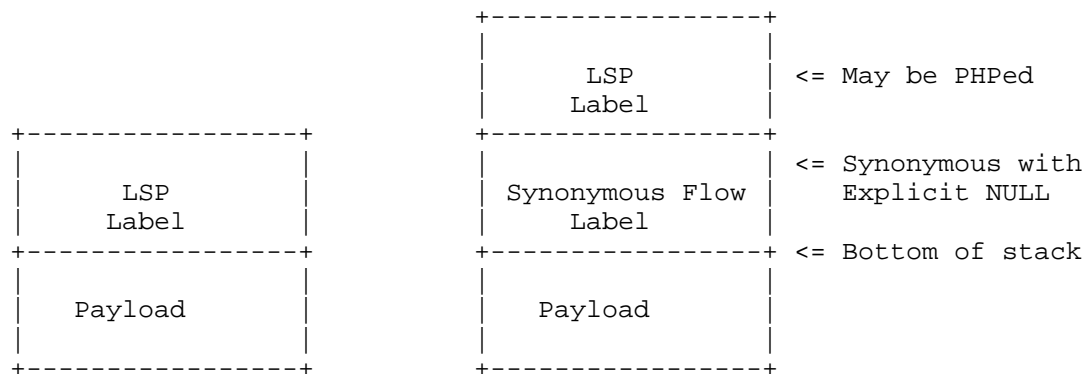
At the egress LSR the LSP label is popped (if present). Then the SFL is processed in exactly the same way as the corresponding application label would have been processed.

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.

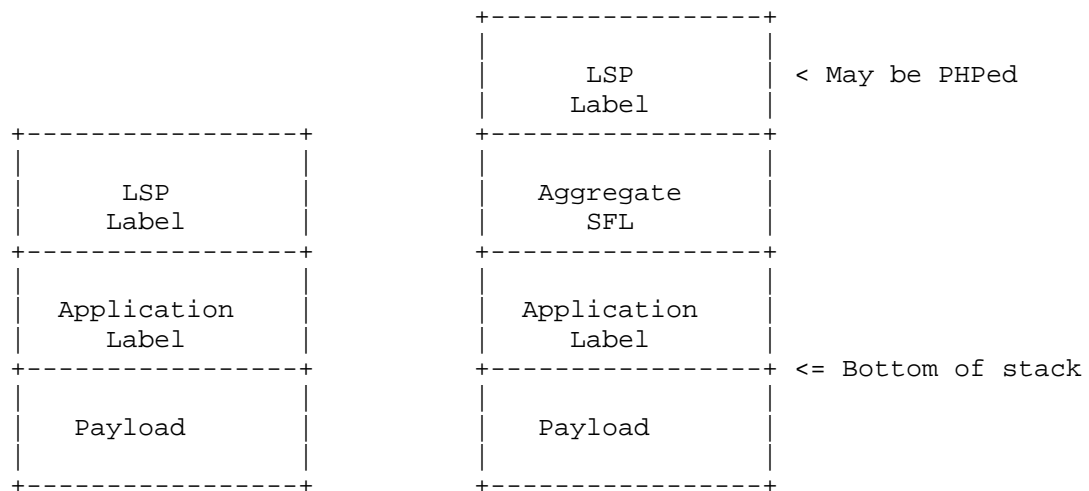
If the LSP label is not present due to PHP action in the upstream LSR, two almost equivalent processing actions can take place. Either the SFL can be treated as an LSP label that was not PHPed and the additional associated SFL action is taken when the label is processed. Alternatively, it can be treated 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.



"Normal" Label Stack

Label Stack with SFL

Figure 3: Aggregate SFL Actions

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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <<http://www.rfc-editor.org/info/rfc5462>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<http://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [I-D.ietf-mpls-flow-ident]
Bryant, S., Pignataro, C., Chen, M., Li, Z., and G. Mirsky, "MPLS Flow Identification Considerations", draft-ietf-mpls-flow-ident-04 (work in progress), February 2017.
- [RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <<http://www.rfc-editor.org/info/rfc6374>>.
- [RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <<http://www.rfc-editor.org/info/rfc6790>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<http://www.rfc-editor.org/info/rfc7258>>.

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

S. Bryant, Ed.
Huawei
A. Farrel, Ed.
J. Drake
Juniper Networks
J. Tantsura
Individual
October 30, 2017

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

11. IANA Considerations	15
12. Acknowledgements	15
13. Contributors	15
14. References	15
14.1. Normative References	15
14.2. Informative References	16
Authors' Addresses	17

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.

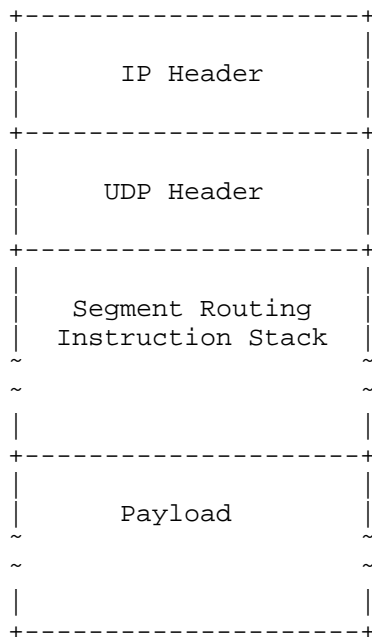


Figure 1: Packet Encapsulation

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:

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

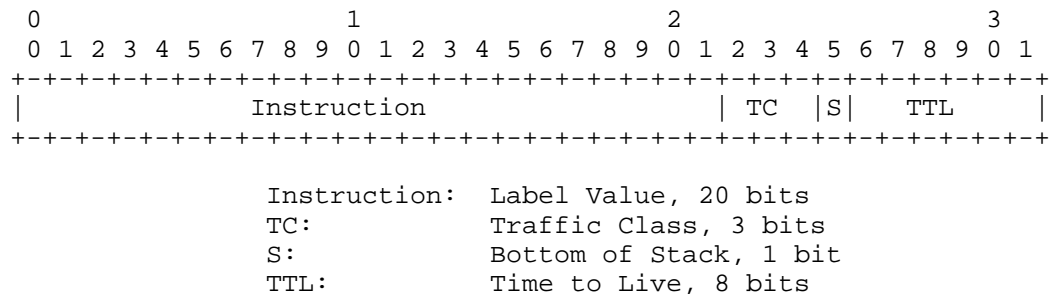


Figure 2: SRIS Label Stack Entry

As with [I-D.ietf-spring-segment-routing-mpls] a 32 bit LSE is used to carry each SR instruction. The instruction itself is carried in the 20 bit Label Value field. The TC field has the normal meaning as

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:

- o 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.
- o Legacy transit nodes that are IP routers but that are not SR capable (i.e., are not able to perform segment routing).
- o Transit nodes that are SR capable but that are not identified by a SID in the SID stack.
- o Transit nodes that are SR capable and need to perform SR routing because they are identified by a SID in the SID stack.
- o The penultimate SR capable node on the path that processes the last SID on the stack on behalf of the domain egress node.
- o 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:

- o Perform TTL processing as normal for an IP packet.
- o Determine that the packet is addressed to the local node.
- o Find that the payload is UDP and that the destination port indicates MPLS-in-UDP.
- o Strip the IP and UDP headers.
- o Pop the top label from the SID stack and retain it for use below.
- o 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.
- o 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.
- o 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:

- o Perform TTL processing as normal for an IP packet.
- o Determine that the packet is addressed to the local node.

- o Find that the payload is UDP and that the destination port indicates MPLS-in-UDP.
- o Strip the IP and UDP headers.
- o Pop the top label from the SID stack and retain it for use below.
- o Pop the next label from the SID stack.
- o 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.
- o 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.
- o 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:

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

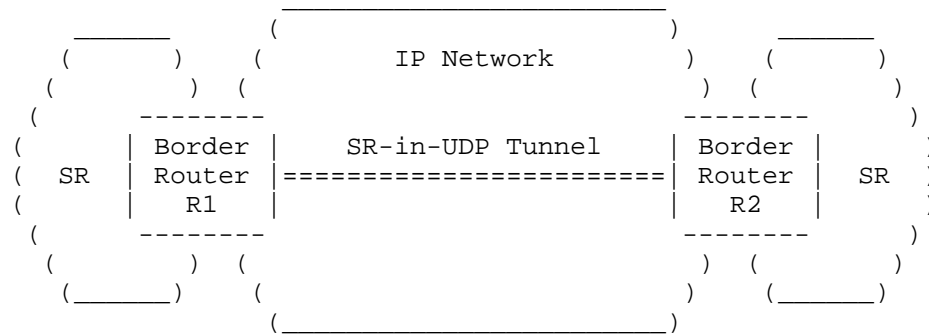


Figure 3: SR in UDP to Tunnel Between SR Sites

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.

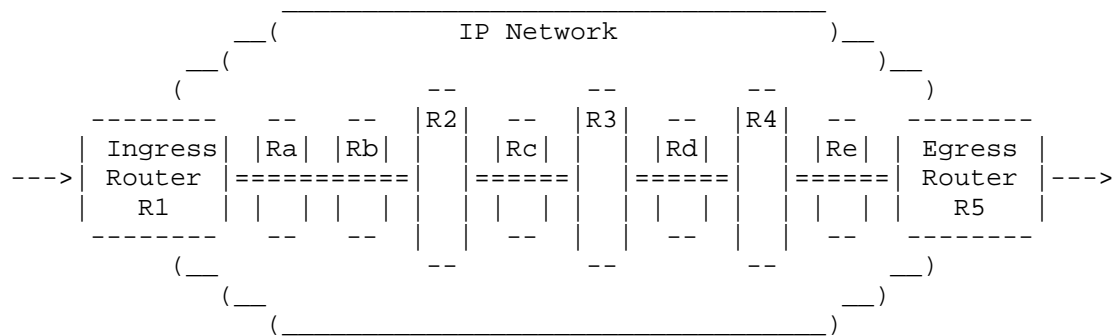


Figure 4: SR Within an IP Network

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:

- o In IS-IS, by using the SR-Capabilities TLV as defined in [I-D.ietf-isis-segment-routing-extensions]
- o 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:

- o For IS-IS, the N (no-PHP) flag in the Prefix-SID sub-TLV indicates whether PHP is not to be used.
- o 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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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, <<https://www.rfc-editor.org/info/rfc3032>>.

- [RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <<https://www.rfc-editor.org/info/rfc5462>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <<https://www.rfc-editor.org/info/rfc7510>>.

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-segment-routing-header-07 (work in progress), July 2017.
- [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. jeffrant@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", draft-ietf-ospf-segment-
routing-extensions-21 (work in progress), October 2017.
- [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. jeffrant@gmail.com, "Unified Source Routing
Instructions using MPLS Label Stack", draft-xu-mpls-
unified-source-routing-instruction-04 (work in progress),
September 2017.
- [RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path
Algorithm", RFC 2992, DOI 10.17487/RFC2992, November 2000,
<<https://www.rfc-editor.org/info/rfc2992>>.
- [RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation
Edge-to-Edge (PWE3) Architecture", RFC 3985,
DOI 10.17487/RFC3985, March 2005,
<<https://www.rfc-editor.org/info/rfc3985>>.
- [RFC4023] Worster, T., Rekhter, Y., and E. Rosen, Ed.,
"Encapsulating MPLS in IP or Generic Routing Encapsulation
(GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005,
<<https://www.rfc-editor.org/info/rfc4023>>.

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

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: 18 August 2022

G. Mirsky
Ericsson
J. Tantsura
Juniper Networks
I. Varlashkin
Google
M. Chen
Huawei
14 February 2022

Bidirectional Forwarding Detection (BFD) Directed Return Path for MPLS
Label Switched Paths (LSPs)
draft-ietf-mpls-bfd-directed-19

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 18 August 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
1.1. Conventions used in this document	3
1.1.1. Requirements Language	3
2. Problem Statement	3
3. Control of the Reverse BFD Path	3
3.1. BFD Reverse Path TLV	3
3.2. Return Codes	5
4. Use Case Scenario	5
5. Operational Considerations	6
6. IANA Considerations	6
6.1. BFD Reverse Path TLV	6
6.2. Return Code	7
7. Implementation Status	7
8. Security Considerations	8
9. Normative References	8
Appendix A. Acknowledgments	9
Authors' Addresses	9

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, while not defining means to control the path an egress BFD system uses 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 system 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]. 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:

- * detection by an ingress node of a failure on the reverse path may not be unambiguously interpreted as the failure of the path in the forward direction.

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 MAY contain none, one or more sub-TLVs 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.

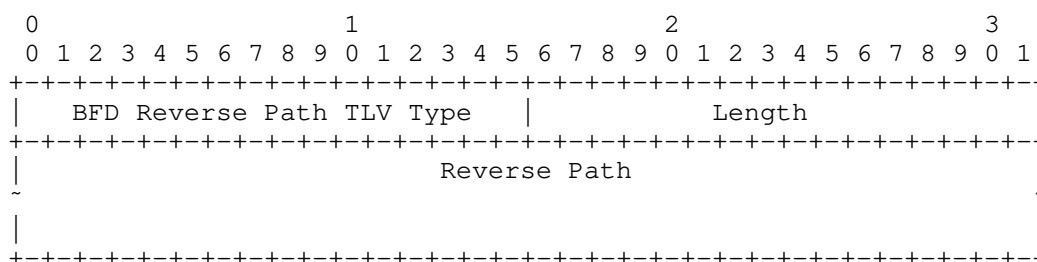


Figure 1: BFD Reverse Path TLV

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 6).

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

Reverse Path field contains none, one or more sub-TLVs. 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 3.2. 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 3.2. 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 an IP network, as defined in [RFC5884].

The BFD Reverse Path TLV MAY be used in the bootstrapping of a BFD session process described in Section 6 [RFC5884]. A system that supports this specification MUST support using the BFD Reverse Path TLV after the BFD session has been established. If a system that supports this specification receives an LSP Ping with the BFD Discriminator TLV and no BFD Reverse Path TLV even though the reverse path for the specified BFD session has been established according to the previously received BFD Reverse Path TLV, the egress LSR MUST transition to transmitting periodic BFD Control messages as defined in Section 7 [RFC5884].

3.2. 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.

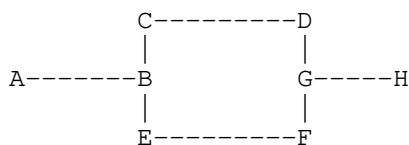


Figure 2: Use Case for BFD Reverse Path TLV

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. Operational Considerations

When an explicit path 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 monitors whether at the egress BFD node the reverse LSP is mapped to the same FEC as the BFD session. Selection and control of the rate of LSP Ping with Return Path TLV follows the recommendation of [RFC5884]: "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."

Suppose an operator planned network maintenance activity that possibly affects FEC used in the BFD Reverse Path TLV. In that case, the operator MUST avoid the unnecessary disruption using the LSP Ping with a new FEC in the BFD Reverse Path TLV. But in some scenarios, proactive measures cannot be taken. Because the frequency of LSP Ping messages will be lower than the defect detection time provided by the BFD session. As a result, a change in the reverse-path FEC will first be detected as the BFD session's failure. In such a case, the ingress BFD node SHOULD immediately transmit the LSP Ping Echo request with Return Path TLV to verify whether the FEC is still valid. If the failure was caused by the change in the FEC used for the reverse direction of the BFD session, the ingress BFD node SHOULD bootstrap a new BFD session using another FEC in BFD Reverse Path TLV.

6. IANA Considerations

6.1. 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.

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

Table 1: New BFD Reverse Type TLV

6.2. 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.

Value	Description	Reference
(TBD3)	Inappropriate Target FEC Stack sub-TLV present.	This document
(TBD4)	Failed to establish the BFD session. The specified reverse path was not found.	This document

Table 2: New Return Code

7. Implementation Status

- The organization responsible for the implementation: ZTE Corporation.
 - The implementation's name ROSng empowers traditional routers, e.g., ZXCTN 6000.
 - A brief general description: A Return Path can be specified for a BFD session over RSVP tunnel or LSP. The same can be specified for a backup RSVP tunnel/LSP.
- The implementation's level of maturity: production.
- Coverage: RSVP LSP (no support for Static LSP)
 - Version compatibility: draft-ietf-mpls-bfd-directed-10.
 - Licensing: proprietary.

- Implementation experience: simple once you support RFC 7110.
- Contact information: Qian Xin qian.xin2@zte.com.cn
- The date when information about this particular implementation was last updated: 12/16/2019

Note to RFC Editor: This section MUST be removed before publication of the document.

8. Security Considerations

Security considerations discussed in [RFC5880], [RFC5884], [RFC7726], [RFC8029], and [RFC7110] apply to this document.

9. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <<https://www.rfc-editor.org/info/rfc5881>>.
- [RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883, June 2010, <<https://www.rfc-editor.org/info/rfc5883>>.
- [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010, <<https://www.rfc-editor.org/info/rfc5884>>.
- [RFC7110] Chen, M., Cao, W., Ning, S., Jounay, F., and S. Delord, "Return Path Specified Label Switched Path (LSP) Ping", RFC 7110, DOI 10.17487/RFC7110, January 2014, <<https://www.rfc-editor.org/info/rfc7110>>.

- [RFC7726] Govindan, V., Rajaraman, K., Mirsky, G., Akiya, N., and S. Aldrin, "Clarifying Procedures for Establishing BFD Sessions for MPLS Label Switched Paths (LSPs)", RFC 7726, DOI 10.17487/RFC7726, January 2016, <<https://www.rfc-editor.org/info/rfc7726>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

Appendix A. Acknowledgments

The authors greatly appreciate a thorough review and the most helpful comments from Eric Gray and Carlos Pignataro. The authors much appreciate the help of Qian Xin, who provided information about the implementation of this specification.

Authors' Addresses

Greg Mirsky
Ericsson

Email: gregimirsky@gmail.com

Jeff Tantsura
Juniper Networks

Email: jefftant.ietf@gmail.com

Ilya Varlashkin
Google

Email: Ilya@nobulus.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 21, 2020

K. Raza, Ed.
R. Asati
Cisco Systems

X. Liu
Volta Networks

S. Esale
Juniper Networks

X. Chen
Huawei Technologies

H. Shah
Ciena Corporation

March 20, 2020

YANG Data Model for MPLS LDP
draft-ietf-mpls-ldp-yang-09

Abstract

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

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

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 21, 2020.

Copyright Notice

Copyright (c) 2020 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. The Complete Tree	7
5. Configuration	16
5.1. Configuration Hierarchy	19
5.1.1. Global parameters	20
5.1.2. Capabilities parameters	20
5.1.3. Per-Address-Family parameters	20
5.1.4. Hello Discovery parameters	20
5.1.5. Peer parameters	21
5.1.6. Forwarding parameters	21
6. Operational State	22
6.1. Adjacency state	22
6.2. Peer state	23
6.3. Bindings state	24
6.4. Capabilities state	26
7. Notifications	27
8. Action	27
9. YANG Specification	27
9.1. Base	27
9.2. Extended	59
10. Security Considerations	80
10.1. YANG model	80
10.1.1. Writable nodes	81
10.1.2. Readable nodes	81
10.1.3. RPC operations	82
10.1.4. Notifications	83
11. IANA Considerations	83
12. Acknowledgments	83
13. Contributors	84
14. Normative References	84
15. Informative References	87
Appendix A. Data Tree Example	88
Authors' Addresses	92

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] is one of the network management protocols that defines mechanisms to manage network devices. YANG [RFC6020] [RFC7950] 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] specifications.

The data model is defined for the 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 the 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 other non-base features. All the items in a base category are mandatory and hence no "if-feature" is allowed under the "base" category. 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 [RFC5443], downstream-on-demand etc. It 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 both the "base" and "extended" models support from the vendors.

2. Specification of Requirements

In this document, the word "IP" is used to refer to both IPv4 and IPv6, unless otherwise explicitly stated. For example, "IP address family" should 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 specifies the base LDP features and augments /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol defined in [RFC8349]. We define new identity 'mpls-ldp' for LDP and the model allows only a single instance of 'mpls-ldp'.
- * "ietf-mpls-ldp-extended" module that specifies the extended LDP features and augments the base LDP module.

It is to be noted that mLDP YANG model [I-D.ietf-mpls-mldp-yang] augments LDP base and extended modules to specify the mLDP specific base and extended features.

There are four types of containers in our module(s):

- * Read-Write parameters for configuration (Section 5)
- * Read-only parameters for operational state (Section 6)
- * Notifications for events (Section 7)
- * RPCs for executing commands to perform some action (Section 8)

The modules in this document conforms to the Network Management Datastore Architecture (NMDA) defined in [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407]. 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 control-plane-protocol
      +-- rw mpls-ldp
        +-- rw ...
          +-- rw ... // base
          |   +-- rw ...
          |   +-- ro ...
          |   +--
          +-- ro ...
          |   +-- ro ...
          |   +-- ro ...
          |   +--
          +-- rw ldp-ext: .... // extended
          |   +-- rw ...
          |   +-- ro ...
          |   +--
          +-- ro ...
          |   +-- ro ...
          |   +-- ro ...

```

rpcs:

```

+-- x mpls-ldp-some_action
+-- x . . . . .

```

notifications:

```

+--- n mpls-ldp-some_event
+--- n ...

```

Figure 1: LDP YANG tree organization

Before going into data model details, it is important to take note of the following points:

- * This model aims to address only the core LDP parameters as per RFC specification, as well as well-known and widely deployed manageability controls (such as label filtering policies to apply filtering rules on the assignment, advertisement, and acceptance for label bindings). 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 model does not cover any applications running on top of LDP, nor does it cover any OAM procedures for LDP.
- * This model is a VPN Routing and Forwarding (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 [RFC8529], 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 prefix-set and neighbor-set respectively as defined in routing-policy model [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 (GR) 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 simplified graphical tree representation of base and extended LDP YANG data model is presented in Figure 2. The meaning of the symbols in these tree diagrams is defined in [RFC8340].

The actual YANG specification for base and extended modules is captured in Section 9.

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

4. The Complete Tree

Following is a complete tree representation of configuration, state, notification, and RPC items under LDP base and extended modules.

```

module: ietf-mpls-ldp
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol:
      +--rw mpls-ldp
        +--rw global
          +--rw capability
            +--rw ldp-ext:end-of-lib {capability-end-of-lib}?
            |   +--rw ldp-ext:enabled?    boolean
            +--rw ldp-ext:typed-wildcard-fec
            |   {capability-typed-wildcard-fec}?
            |   +--rw ldp-ext:enabled?    boolean
            +--rw ldp-ext:upstream-label-assignment
            |   {capability-upstream-label-assignment}?
            |   +--rw ldp-ext:enabled?    boolean
          +--rw graceful-restart
            +--rw enabled?                  boolean
            +--rw reconnect-time?          uint16
            +--rw recovery-time?           uint16
            +--rw forwarding-holdtime?     uint16
            +--rw ldp-ext:helper-enabled?  boolean
            |   {graceful-restart-helper-mode}?
          +--rw lsr-id?
            |   rt-types:router-id
          +--rw address-families
            +--rw ipv4!
            |   +--rw enabled?              boolean
            |   +--ro label-distribution-control-mode?  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:enabled?    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:enabled?
|       | boolean
+---rw ldp-ext:label-policy
|   +---rw ldp-ext:advertise
|       | +---rw ldp-ext:egress-explicit-null
|       | | +---rw ldp-ext:enabled?    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-control-mode?
|   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          identityref
|     |     |   +---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 enabled?          boolean
|     |       |   |   +---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:enabled?          boolean
+--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
|   {per-interface-timer-config}?
+--rw ldp-ext:hello-interval?      uint16
|   {per-interface-timer-config}?

```



```

    |         +---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 enabled?               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 enabled?           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:enabled?             boolean
+---rw ldp-ext:local-address?
|   inet:ipv6-address
+---rw peers
+---rw authentication
+---rw (authentication-type)?
+---: (password)
|   +---rw key?                    string
|   +---rw crypto-algorithm?       identityref
+---: (ldp-ext: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 (authentication-type)?
+---: (password)
|   +---rw key?                    string
|   +---rw crypto-algorithm?       identityref
+---: (ldp-ext:key-chain) {key-chain}?
+---rw ldp-ext:key-chain?          key-chain:key-chain-ref
+---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
+--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!
+--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:interface?
|       |   if:interface-ref
+--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
+--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 enabled?      boolean
|   |   +--ro reconnect-time?  uint16
|   |   +--ro recovery-time?   uint16
|   +--ro capability
|   |   +--ro end-of-lib
|   |   |   +--ro enabled?    boolean
|   |   +--ro typed-wildcard-fec
|   |   |   +--ro enabled?    boolean
|   |   +--ro upstream-label-assignment
|   |   |   +--ro enabled?    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?
|   rt-types:timeticks64
+--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
+---rw ldp-ext:admin-down?         boolean
|   {per-peer-admin-down}?
+---rw ldp-ext:graceful-restart
|   {per-peer-graceful-restart-config}?
+---rw ldp-ext:enabled?            boolean
+---rw ldp-ext:reconnect-time?     uint16
+---rw ldp-ext:recovery-time?      uint16
+---rw ldp-ext:session-ka-holdtime? uint16
|   {per-peer-session-attributes-config}?
+---rw ldp-ext:session-ka-interval? uint16
|   {per-peer-session-attributes-config}?
+---rw ldp-ext:session-downstream-on-demand
|   {session-downstream-on-demand-config}?
+---rw ldp-ext:enabled?            boolean
+---rw ldp-ext:peer-list?          peer-list-ref
+---rw ldp-ext:dual-stack-transport-preference
|   {peers-dual-stack-transport-preference}?
+---rw ldp-ext:max-wait?            uint16
+---rw ldp-ext:prefer-ipv4!
|   +---rw ldp-ext:peer-list?      peer-list-ref

rpccs:
+---x mpls-ldp-clear-peer
|   +---w input
|   |   +---w protocol-name?      leafref
|   |   +---w lsr-id?             leafref
|   |   +---w label-space-id?     leafref
+---x mpls-ldp-clear-hello-adjacency
|   +---w input
|   |   +---w hello-adjacency
|   |   |   +---w protocol-name?    leafref
|   |   |   +---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 protocol-name?   leafref
|       +---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 protocol-name?   leafref
|       +--ro lsr-id?          leafref
|       +--ro label-space-id?  leafref
+---n mpls-ldp-hello-adjacency-event
|   +--ro event-type?   oper-status-event-type
|   +--ro protocol-name?   leafref
|   +--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 protocol-name?   leafref
|   +--ro fec?           inet:ip-prefix

```

Figure 2: Complete Tree

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/rt:control-plane-protocol that is defined in [RFC8349] and follows NMDA as mentioned earlier.

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

```

augment /rt:routing/rt:control-plane-protocols:
  /rt:control-plane-protocol:
    +-- 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]
          +-- ...
          +-- ...

```

Figure 3: Base Configuration organization

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

```

augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protoc
ol
    +-- mpls-ldp
    +-- global
    |   +-- ...
    |   +-- ...
    |   +-- address-families
    |   |   +-- ipv4
    |   |   |   +-- . . .
    |   |   |   +-- . . .
    |   |   |   +-- label-policy
    |   |   |       +-- ...
    |   |   |       +-- ...
    |   |   +-- ipv6
    |   |   |   +-- . . .
    |   |   |   +-- . . .
    |   |   |   +-- label-policy
    |   |   |       +-- ...
    |   |   |       +-- ...
    |   +-- capability
    |   |   +-- ...
    |   |   +-- ...
    |   +-- discovery
    |   |   +-- interfaces
    |   |   |   +-- ...
    |   |   |   +-- ...
    |   |   |   +-- interface* [interface]
    |   |   |       +-- ...
    |   |   |       +-- address-families
    |   |   |       |   +-- ipv4
    |   |   |       |   |   +-- ...
    |   |   |       |   |   +-- ...
    |   |   |       |   +-- ipv6
    |   |   |       |       +-- ...
    |   |   |       |       +-- ...
    |   |   +-- targeted
    |   |   |   +-- ...
    |   |   |   +-- address-families
    |   |   |       +-- ipv6
    |   |   |       |   +-- target* [adjacent-address]
    |   |   |       |       +-- ...
    |   |   |       |       +-- ...
    +-- forwarding-nextthop
    |   +-- ...
    |   +-- ...
    +-- peers
    |   +-- ...
    |   +-- ...
    +-- peer*

```



```

+-- ...
+-- ...
+-- label-policy
|   +-- ..
+-- address-families
    +-- ipv4
    |   +-- ...
    +-- ipv6
        +-- ...

```

Figure 4: Extended Configuration organization

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 parents. 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 Hierarchy

LDP module resides under a 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.

- * Global parameters
- * Per-address-family parameters
- * LDP Capabilities parameters
- * Hello Discovery parameters
 - interfaces
 - o Global
 - o Per-interface: Global
 - o Per-interface: Per-address-family
 - targeted
 - o Global

- o Per-address-family: Per-target
- * Peer parameters
 - Global
 - Per-peer: Global
 - Per-peer: Per-address-family

- * Forwarding parameters

Following subsections briefly explain these configuration areas.

5.1.1. 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 (within a routing domain) as an LSR Id.

5.1.2. Capabilities parameters

This container falls under the 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.

5.1.3. 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.

5.1.4. 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 the former is interface hello timers, and example of the 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 means to configure multiple target addresses to perform extended discovery to a specific destination target, as well as to fine-tune the per-target parameters.

5.1.5. 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 by its LSR Id.

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 the former is per-peer session password configuration, whereas the example of the latter is prefix-list based label policies (inbound and outbound) that apply to a given peer.

5.1.6. 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/rt:control-plane-protocol) as the configuration.

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

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

6.1. 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). It 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. The tree is shown for ipv4 address-family only; a similar tree exists for ipv6 address-family as well.

```

+--rw mpls-ldp!
  +--rw discovery
    +--rw interfaces
      +--rw interface* [interface]
        +--rw address-families
          +--rw ipv4
            +--ro hello-adjacencies
              +--ro hello-adjacencies* [adjacent-address]
                +--ro adjacent-address
                  . . . .
            +--ro targeted
              +--rw address-families
                +--rw ipv4
                  +--ro hello-adjacencies
                    +--ro hello-adjacencies*
                      | [local-address adjacent-address]
                    +--ro local-address
                      +--ro adjacent-address
                        . . . .
                        . . . .

```

Figure 5: Adjacency state

6.2. Peer 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. The peer's hello adjacencies tree is shown for ipv4 address-family only; a similar tree exists for ipv6 address-family as well.

```

+--rw mpls-ldp!
  +--rw peers
    +--rw peer* [lsr-id label-space-id]
      +--rw lsr-id
      +--rw label-space-id
      +--ro label-advertisement-mode
      +--ro session-state
      +--ro tcp-connection
      +--ro session-holdtime?
      +--ro up-time
      +-- . . . .
      +--ro address-families
        +--ro ipv4
          +--ro hello-adjacencies
            +--ro hello-adjacencies*
              [local-address adjacent-address]
              . . . .
              . . . .
      +--ro received-peer-state
        +--ro . . . .
        +--ro capability
          +--ro . . . .
      +--ro statistics
        +-- . . . .
        +-- received
          +-- ...
        +-- sent
          +-- ...

```

Figure 6: Peer state

6.3. 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:
    . . . .
  FEC 2001:db8:0:2::
    . . . .
  FEC 2001:db8:0:3::
    . . . .

Address bindings:
  Addr 192.0.2.10:
    advertised
  Addr 2001:db8:0:10::
    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
  Addr 2001:db8:0:2::
    received, peer 192.0.2.2:0
  Addr 2001:db8:0:3::
    received, peer 192.0.2.3:0

```

Figure 7: Example Bindings

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. The tree shown below is for ipv4 address-family only; a similar tree exists for ipv6 address-family as well.

```

+--rw mpls-ldp!
  +--rw global
    +--rw address-families
      +--rw ipv4
        +--ro bindings
          +--ro address* [address]
            +--ro address (ipv4-address or ipv6-address)
            +--ro advertisement-type? advertised-received
            +--ro peer? leafref
          +--ro fec-label* [fec]
            +--ro fec (ipv4-prefix or ipv6-prefix)
            +--ro peer* [peer advertisement-type]
              +--ro peer leafref
              +--ro advertisement-type? advertised-received
              +--ro label? mpls:mpls-label
              +--ro used-in-forwarding? boolean

```

Figure 8: Bindings state

6.4. 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 9: Capabilities state

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 (Next Hop Label Forwarding Entry) with outgoing label.

A simplified graphical representation of the data model for LDP notifications is shown in Figure 2.

8. Action

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.

A simplified graphical representation of the data model for LDP actions is shown in Figure 2.

9. YANG Specification

Following sections specify the actual YANG (module) specification for LDP constructs defined earlier in the document.

9.1. Base

This YANG module imports types defined in [RFC6991], [RFC8349], [RFC8294], [RFC8343], and [RFC8344].

```
<CODE BEGINS> file "ietf-mpls-ldp@2020-02-25.yang"
```

```
// RFC Editor: replace the above date 2020-02-25 with the date of  
// publication and remove this note.
```

```
module ietf-mpls-ldp {  
  yang-version 1.1;  
  
  namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp";  
  prefix "ldp";  
  
  import ietf-inet-types {
```

```
    prefix "inet";
    reference "RFC 6991: Common YANG Data Types";
}

import ietf-yang-types {
    prefix "yang";
    reference "RFC 6991: Common YANG Data Types";
}

import ietf-routing {
    prefix "rt";
    reference
        "RFC 8349: A YANG Data Model for Routing Management (NMDA
        version)";
}

import ietf-routing-types {
    prefix "rt-types";
    reference
        "RFC 8294: Common YANG Data Types for the Routing Area";
}

import ietf-interfaces {
    prefix "if";
    reference "RFC 8343: A YANG Data Model for Interface Management";
}

import ietf-ip {
    prefix "ip";
    reference "RFC 7277: A YANG Data Model for IP Management";
}

import ietf-key-chain {
    prefix "key-chain";
    reference "RFC 8177: YANG Data Model for Key Chains";
}

organization
    "IETF MPLS Working Group";
contact
    "WG Web:  <http://tools.ietf.org/wg/mpls/>
    WG List:  <mailto:mpls@ietf.org>

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

    Editor:   Rajiv Asati
              <mailto:rajiva@cisco.com>
```

Editor: Xufeng Liu
<mailto:xufeng.liu.ietf@gmail.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) 2020 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.";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

```
revision 2020-02-25 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for MPLS LDP.";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

/*
 * Typedefs
 */
typedef advertised-received {
  type enumeration {
```

```
    enum advertised {
      description "Advertised information.";
    }
    enum received {
      description "Received information.";
    }
  }
  description
    "Received or advertised.";
}

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

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 mpls-ldp {
  base rt:control-plane-protocol;
  description
    "LDP protocol.";
  reference
    "RFC 5036: LDP Specification";
}

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 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;
            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;
    description
      "The time interval for which a LDP link Hello adjacency
      is maintained in the absence of link Hello messages from
      the LDP neighbor.
      This leaf may be configured at the per-interface level or
      the global level, with precedence given to the value at the
      per-interface level. If the leaf is not configured at
      either level, the default value at the global level is
      used.";
  }
  leaf hello-interval {
    type uint16 {
      range 5..1200;
    }
    units seconds;
    description
      "The interval between consecutive LDP link Hello messages
      used in basic LDP discovery.
      This leaf may be configured at the per-interface level or
      the global level, with precedence given to the value at the
      per-interface level. If the leaf is not configured at
      either level, the default value at the global level is
      used.";
  }
} // 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-from-binding;
  }
} // 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-from-binding {
      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. On the peer side, these attributes
        are operational state data received from the peer.";

    container graceful-restart {
        description
            "Attributes for graceful restart.";
        leaf enabled {
            type boolean;
            description
                "Enable or disable graceful restart.
                This leaf may be configured at the per-peer level or the
                global level, with precedence given to the value at the
                per-peer level. If the leaf is not configured at either
                level, the default value at the global level is used.";
        }
        leaf reconnect-time {
            type uint16 {
                range 10..1800;
            }
            units seconds;
            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.
                This leaf may be configured at the per-peer level or the
                global level, with precedence given to the value at the
                per-peer level. If the leaf is not configured at either
                level, the default value at the global level is used.";
        }
        leaf recovery-time {
            type uint16 {
                range 30..3600;
            }
            units seconds;
            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.
        This leaf may be configured at the per-peer level or the
        global level, with precedence given to the value at the
        per-peer level. If the leaf is not configured at either
        level, the default value at the global level is used.";
    }
} // 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-absolute {
    description
        "An absolute reference to an LDP peer, by the LDP ID, which
        consists of the LSR ID and the Label Space ID.";

    leaf protocol-name {
        type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
                + "rt:control-plane-protocol/rt:name";
        }
        description
            "The name of the LDP protocol instance.";
    }
    leaf lsr-id {
        type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
                + "rt:control-plane-protocol"
                + "[rt:name=current()]/../protocol-name/"
                + "ldp:mpls-ldp/ldp:peers/ldp:peer/ldp:lsr-id";
        }
    }
}
```

```
        description
            "The LSR ID of the peer, as a portion of the peer LDP ID.";
    }
    leaf label-space-id {
        type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
                + "rt:control-plane-protocol"
                + "[rt:name=current()/../protocol-name]/"
                + "ldp:mpls-ldp/ldp:peers/"
                + "ldp:peer[ldp:lsr-id=current()/../lsr-id]/"
                + "ldp:label-space-id";
        }
        description
            "The Label Space ID of the peer, as a portion of the peer
            LDP ID.";
    }
} // ldp-peer-ref-absolute

grouping ldp-peer-ref-from-binding {
    description
        "A relative 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 {
            path "../..../..../..../ldp:peers/ldp:peer/ldp:lsr-id";
        }
        description
            "The LSR ID of the peer, as a portion of the peer LDP ID.";
    }
    leaf label-space-id {
        type leafref {
            path "../..../..../..../ldp:peers/"
                + "ldp:peer[ldp:lsr-id=current()/../lsr-id]/"
                + "ldp:label-space-id";
        }
        description
            "The Label Space ID of the peer, as a portion of the peer
            LDP ID.";
    }
} // ldp-peer-ref-from-binding

grouping ldp-peer-ref-from-interface {
    description
        "A relative reference to an LDP peer, by the LDP ID, which
        consists of the LSR ID and the Label Space ID.";

    container peer {
```

```

description
  "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 {
    path "../../../../../ldp:peers/ldp:peer/"
      + "ldp:lsr-id";
  }
  description
    "The LSR ID of the peer, as a portion of the peer LDP ID.";
}
leaf label-space-id {
  type leafref {
    path "../../../../../ldp:peers/"
      + "ldp:peer[ldp:lsr-id=current()/../lsr-id]/"
      + "ldp:label-space-id";
  }
  description
    "The Label Space ID of the peer, as a portion of the peer
    LDP ID.";
}
} // peer
} // ldp-peer-ref-from-interface

grouping ldp-peer-ref-from-target {
  description
    "A relative reference to an LDP peer, by the LDP ID, which
    consists of the LSR ID and the Label Space ID.";

  container peer {
    description
      "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 {
        path "../../../../../ldp:peers/ldp:peer/"
          + "ldp:lsr-id";
      }
      description
        "The LSR ID of the peer, as a portion of the peer LDP ID.";
    }
    leaf label-space-id {
      type leafref {
        path "../../../../../ldp:peers/"
          + "ldp:peer[ldp:lsr-id=current()/../lsr-id]/"
          + "ldp:label-space-id";
      }
      description

```

```
        "The Label Space ID of the peer, as a portion of the peer
        LDP ID.";
    }
} // peer
} // ldp-peer-ref-from-target

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;
        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.
            This leaf may be configured at the per-peer level or the
            global level, with precedence given to the value at the
            per-peer level. If the leaf is not configured at either
            level, the default value at the global level is used.";
    }
    leaf session-ka-interval {
        type uint16 {
            range 15..1200;
        }
        units seconds;
        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.
            This leaf may be configured at the per-peer level or the
            global level, with precedence given to the value at the
            per-peer level. If the leaf is not configured at either
            level, the default value at the global level is used.";
    }
} // 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 authentication-type {
  description
    "Choice of authentication.";
  case password {
    leaf key {
      type string;
      description
        "This leaf specifies the authentication key. The length
        of the key may be dependent on the cryptographic
        algorithm.";
    }
    leaf crypto-algorithm {
      type identityref {
        base key-chain:crypto-algorithm;
      }
      description
        "Cryptographic algorithm associated with key.";
    }
  }
}
} // 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;
      description
        "Local Label Advertisement Mode.";
    }
    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 duration from now until sending 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 "Peer capability information.";
            container end-of-lib {
                description
                    "Peer's end-of-lib capability.";
                leaf enabled {
                    type boolean;
                    description
                        "'true' if peer's end-of-lib capability is enabled.";
                }
            }
        }
        container typed-wildcard-fec {
            description
                "Peer's typed-wildcard-fec capability.";
            leaf enabled {
                type boolean;
                description
                    "'true' if peer's typed-wildcard-fec capability is
                    enabled.";
            }
        }
        container upstream-label-assignment {
            description
                "Peer's upstream label assignment capability.";
            leaf enabled {
                type boolean;
                description
                    "'true' if peer's upstream label assignment is
                    enabled.";
            }
        }
    }
}
```

```
    } // 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.";
  }
} // tcp-connection

leaf up-time {
  type rt-types:timeticks64;
  config false;
  description
    "The number of time ticks (hundredths of a second) since the
    the state of the session with the peer changed to
    OPERATIONAL.";
}

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
            "The number of address messages sent or received.";
    }
    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 notification messages sent or received.";
    }
} // statistics-peer-received-sent

/*
 * Configuration data and operational state data nodes
 */
```

```
augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol" {
    when "derived-from-or-self(rt:type, 'ldp:mpls-ldp')" {
      description
        "This augmentation is only valid for a control-plane
        protocol instance of LDP (type 'mpls-ldp').";
    }
    description
      "LDP augmentation to routing control-plane protocol
      configuration and state.";

    container mpls-ldp {
      must "not (../../rt:control-plane-protocol"
        + "[derived-from-or-self(rt:type, 'ldp:mpls-ldp')]"
        + "[rt:name!=current()/../../rt:name])"
      {
        description "Only one LDP instance is allowed.";
      }

      description
        "Containing configuration and operational data for the LDP
        protocol.";

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

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

        container graceful-restart {
          description
            "Attributes for graceful restart.";
          leaf enabled {
            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.";
}
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.";
}
} // graceful-restart

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 'enabled'
            leaf is set to 'false'";
    }
}
```

```
description
  "Containing data related to the IPv4 address family.";

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

leaf label-distribution-control-mode {
  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;
  }

  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
         or advertised to a peer.";
    }
    uses binding-label-state-attributes;
  }
} // bindings
} // ipv4
} // address-families
} // global

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 {
      refine "hello-holdtime" {
        default 15;
      }
      refine "hello-interval" {
        default 5;
      }
    }
  }

  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 'enabled'
    leaf is set to 'false'";
  description
    "IPv4 address family.";

  leaf enabled {
    type boolean;
    default true;
    description
      "Set to false to disable the address family on
      the interface.";
  }

  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-from-interface;
    }
  } // ipv4
} // address-families
} // 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 enabled {
      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-from-target;
}

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

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

  leaf enabled {
    type boolean;
    default true;
    description
      "'true' to 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 {
      refine session-ka-holdtime {
        default 180;
      }
      refine session-ka-interval {
        default 60;
      }
    }

    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 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.";
    }
  }
} // 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-absolute {
      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 targetttted adjacency. If this is not
        provided then all Hello adjacencies are cleared";
      leaf protocol-name {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
            + "rt:control-plane-protocol/rt:name";
        }
        description
          "The name of the LDP protocol instance.";
      }
      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";
            }
          }
        }
        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/"
          + "rt:control-plane-protocol/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.";
    }
  }
} // hello-adjacency-type
} // hello-adjacency
} // input
} // mpls-ldp-clear-hello-adjacency

rpc mpls-ldp-clear-peer-statistics {
  description
    "Clears protocol statistics (e.g. sent and received
    counters).";
  input {
    uses ldp-peer-ref-absolute {
      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.";
    }
    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-absolute;
    }
}

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.";
    }
    leaf protocol-name {
        type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
                + "rt:control-plane-protocol/rt:name";
        }
        description
            "The name of the LDP protocol instance.";
    }
    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.";
                }
            }
        }
    }
}
```

```
    case link {
      container link {
        description
          "Link adjacency through LDP basic discovery.";
        leaf next-hop-interface {
          type if:interface-ref;
          description
            "The interface connecting to the adjacent next hop.";
        }
        leaf next-hop-address {
          type inet:ip-address;
          must "../next-hop-interface" {
            description
              "Applicable when interface is specified.";
          }
          description
            "IP address of the next hop. This can be IPv4 or IPv6
            address.";
        }
      }
    } // hello-adjacency-type
  } // mpls-ldp-hello-adjacency-event

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 protocol-name {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rt:name";
    }
    description
      "The name of the LDP protocol instance.";
  }
  leaf fec {
    type inet:ip-prefix;
    description
      "The address prefix element of the FEC whose status
      has changed.";
  }
}
}
```


<CODE ENDS>

Figure 10: LDP base module

9.2. Extended

This YANG module imports types defined in [RFC6991], [RFC8349], [RFC8177], and [RFC8343].

```
<CODE BEGINS> file "ietf-mpls-ldp-extended@2020-02-25.yang"

// RFC Editor: replace the above date 2020-02-25 with the date of
// publication and remove this note.

module ietf-mpls-ldp-extended {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp-extended";
  prefix "ldp-ext";

  import ietf-inet-types {
    prefix "inet";
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management (NMDA
      version)";
  }

  import ietf-key-chain {
    prefix "key-chain";
    reference "RFC 8177: YANG Data Model for Key Chains";
  }

  import ietf-mpls-ldp {
    prefix "ldp";
    reference "RFC XXXX: YANG Data Model for MPLS LDP";
    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
  }
}
```

```
import ietf-interfaces {
  prefix "if";
  reference "RFC 8343: A YANG Data Model for Interface Management";
}

import ietf-routing-policy {
  prefix rt-pol;
  reference
    "I-D.ietf-rtgwg-policy-model: A YANG Data Model for Routing
    Policy Management";
}

organization
  "IETF MPLS Working Group";
contact
  "WG Web:  <http://tools.ietf.org/wg/mppls/>
  WG List:  <mailto:mppls@ietf.org>

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

  Editor:    Rajiv Asati
             <mailto:rajiva@cisco.com>

  Editor:    Xufeng Liu
             <mailto:xufeng.liu.ietf@gmail.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 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) 2020 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.";

// RFC Editor: replace XXXX with actual RFC number and remove
// this note

```
revision 2020-02-25 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for MPLS LDP.";

    // RFC Editor: replace XXXX with actual RFC number and remove
    // this note
}

/*
 * Features
 */
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. We call an LSR to be operating in GR
      helper mode when it advertises 0 as its FT Reconnect Timeout
      in the FT Session TLV.
      Please refer RFC3478 section 2 for details.";
  }

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

  feature peers-dual-stack-transport-preference {
    description
      "This feature indicates that the system allows to configure
      the transport connection preference in a dual-stack setup
      for peers.";
  }

  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 leafref {
            path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                + "rt-pol:neighbor-sets/rt-pol:neighbor-set/rt-pol:name";
        }
        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.";
        reference
            "I-D.ietf-rtgwg-policy-model: A YANG Data Model for Routing
            Policy Management";
    }

    typedef prefix-list-ref {
        type leafref {
            path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                + "rt-pol:prefix-sets/rt-pol:prefix-set/rt-pol:name";
        }
        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.";
    reference
        "I-D.ietf-rtgwg-policy-model: A YANG Data Model for Routing
        Policy Management";
}

typedef peer-list-ref {
    type leafref {
        path "/rt-pol:routing-policy/rt-pol:defined-sets/"
            + "rt-pol:neighbor-sets/rt-pol:neighbor-set/rt-pol:name";
    }
    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.";
    reference
        "I-D.ietf-rtgwg-policy-model: A YANG Data Model for Routing
        Policy Management";
}

/*
 * 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.";
    }
}

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.";
    }
  }

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

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

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

    container upstream-label-assignment {
      if-feature capability-upstream-label-assignment;
      description
        "Configure upstream label assignment capability.";
      leaf enabled {
        type boolean;
        default false;
        description
          "'true' to 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.";
}

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
        "Containing a list of interfaces on which forwarding can be
        disabled.";

      list interface {
        key "name";
        description
          "List of LDP interfaces on which forwarding can be
          disabled.";
        uses ldp:ldp-interface-ref;
        list address-family {
          key "afi";
          description
            "Per-vrf per-af params.";
          leaf afi {
            type identityref {
              base rt:address-family;
            }
            description
              "Address family type value.";
          }
          leaf ldp-disable {
            type boolean;
          }
        }
      }
    }
  }
}
```



```
        default false;
        description
            "'true' to disable LDP forwarding on the interface.";
    }
} // interface
} // interfaces
} // forwarding-nexthop
} // global-forwarding-nexthop-augment

grouping graceful-restart-augment {
    description "Augmentation to graceful restart.";

    leaf helper-enabled {
        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.";
}

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;
    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.
       This leaf may be configured at the per-interface level or
       the global level, with precedence given to the value at the
       per-interface level. If the leaf is not configured at
       either level, the default value at the global level is
       used.";
  }
}

grouping peer-af-policy-container {
  description
```

```
    "LDP policy attribute container under peer address-family.";
  container label-policy {
    description
      "Label policy attributes.";
    container advertise {
      description
        "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.";
      }
    }
    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.";
      }
    }
  }
} // 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;
  }
}
```

```
    }  
  }  
  
  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 enabled {  
        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-preference {  
    if-feature peers-dual-stack-transport-preference;  
    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 preferred 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 enabled {
            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.";
    }
}
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.";
    }
}
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).";
        }
    }
}
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.";
    }
}
} // assign
} // label-policy
} // policy-container

/*
 * Configuration and state data nodes
 */
// Forwarding nexthop augmentation to the global tree
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:global" {
```

```
    description "Forwarding nexthop augmentation.";
    uses global-forwarding-nexthop-augment;
}

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

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

        leaf enabled {
            type boolean;
            default true;
            description
                "'false' to disable 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-control-mode {
            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;
    }

    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;
    }
  } // bindings
} // ipv6

// discovery/interfaces/interface/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/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 'enabled'
```



```
        leaf is set to 'false'";
    description
        "IPv6 address family.";

    leaf enabled {
        type boolean;
        default true;
        description
            "'false' to disable the address family on the interface.";
    }

    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-from-interface;
        }
    }
} // ipv6
}

// discovery/targeted/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:discovery/"
+ "ldp:targeted/ldp:address-families" {
    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-from-target;
}

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

    leaf adjacent-address {
        type inet:ipv6-address;
        description
            "Configures a remote LDP neighbor for the
             extended LDP discovery.";
    }
    leaf enabled {
        type boolean;
        default true;
        description
            "'true' to 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/"
+ "rt:control-plane-protocol/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.";
      }
    }
  }
} // ipv6
}
```

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

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/ldp:mpls-ldp/ldp:global/"
+ "ldp:capability" {
  description "Capability augmentation.";
  uses capability-augment;
}

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

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

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/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/"
+ "rt:control-plane-protocol/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/"
+ "rt:control-plane-protocol/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
       any source are accepted.";
  }
}

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

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

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

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

```
        uses authentication-keychain-augment;
    }
}

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

<CODE ENDS>
```

Figure 11: LDP extended module

10. Security Considerations

This specification inherits the security considerations captured in [RFC5920] and the LDP protocol specification documents, namely base LDP [RFC5036], LDP IPv6 [RFC7552], LDP Capabilities [RFC5561], Typed Wildcard FEC [RFC5918], LDP End-of-LIB [RFC5919], and LDP Upstream Label Assignment [RFC6389].

10.1. YANG model

The YANG modules specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or

RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

10.1.1. Writable nodes

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

For LDP, the ability to modify MPLS LDP configuration may allow the entire MPLS LDP domain to be compromised including forming LDP adjacencies and/or peer sessions with unauthorized routers to mount a massive Denial-of-Service (DoS) attack. In particular, following are the subtrees and data nodes that are sensitive and vulnerable:

- * /mpls-ldp/discovery/interfaces/interface: Adding LDP on any unprotected interface could allow an LDP hello adjacency to be formed with an unauthorized and malicious neighbor. Once an hello adjacency is formed, a peer session could progress with this neighbor.
- * /mpls-ldp/discovery/targeted/hello-accept: Allowing acceptance of targeted-hellos could open LDP to DoS attacks related to incoming targeted hellos from malicious sources.
- * /mpls-ldp/peers/authentication: Allowing a peer session establishment is typically controlled via LDP authentication where a proper and secure authentication password/key management is warranted.
- * /mpls-ldp/peers/peer/authentication: Same as above.

10.1.2. Readable nodes

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

The exposure of LDP databases (such as hello adjacencies, peers, address bindings, and fec-label bindings) beyond the scope of the LDP admin domain may be undesirable. The relevant subtrees and data nodes are as follows:

- * /mpls-ldp/global/address-families/ipv4/bindings/address
- * /mpls-ldp/global/address-families/ipv6/bindings/address
- * /mpls-ldp/global/address-families/ipv4/bindings/fec-label
- * /mpls-ldp/global/address-families/ipv6/bindings/fec-label
- * /mpls-ldp/discovery/interfaces/interface/address-families/ipv4/hello-adjacencies
- * /mpls-ldp/discovery/interfaces/interface/address-families/ipv6/hello-adjacencies
- * /mpls-ldp/discovery/targeted/address-families/ipv4/hello-adjacencies
- * /mpls-ldp/discovery/targeted/address-families/ipv6/hello-adjacencies
- * /mpls-ldp/peers

The configuration for LDP peer authentication is supported via the specification of key-chain [RFC8040], or via direct specification of a key associated with a crypto algorithm (such as MD5). The relevant subtrees and data nodes are as follows:

- * /mpls-ldp/peers/authentication
- * /mpls-ldp/peers/peer/authentication

The actual authentication key data (whether locally specified or part of a key-chain) is sensitive and needs to be kept secret from unauthorized parties. For key-chain based authentication, this model inherits the security considerations of [RFC8040] (that includes the considerations with respect to the local storage and handling of authentication keys). A similar procedure for storage and access to direct key is warranted.

10.1.3. RPC operations

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations otherwise control plane flaps, network outages, and DoS attacks are possible. The RPC operations are:

- * mpls-ldp-clear-peer

* mpls-ldp-clear-hello-adjacency

10.1.4. Notifications

The model describes several notifications. The implementations must rate-limit the generation of these notifications to avoid creating significant notification load and possible side effects on the system stability.

11. IANA Considerations

This document requests the registration of the following URIs in the IETF "XML registry" [RFC3688]:

URI	Registrant	XML
urn:ietf:params:xml:ns:yang:ietf-mpls-ldp	The IESG	N/A
urn:ietf:params:xml:ns:yang:ietf-mpls-ldp-extended	The IESG	N/A

Table 1: URIs

This document requests the registration of the following YANG modules in the "YANG Module Names" registry [RFC6020]:

Name	Namespace	Prefix	Reference
ietf-mpls-ldp	urn:ietf:params:xml:ns:yang: :ietf-mpls-ldp	ldp	This document
ietf-mpls-ldp-extended	urn:ietf:params:xml:ns:yang: :ietf-mpls-ldp-extended	ldp- ext	This document

Table 2: YANG Modules

-- RFC Editor: Replace "this document" with the document RFC number at time of publication, and remove this note.

12. Acknowledgments

The authors would like to acknowledge Eddie Chami, Nagendra Kumar, Mannan Venkatesan, and Pavan Beeram for their contribution to this document.

We also acknowledge Ladislav Lhotka, Jan Lindblad, Tom Petch, Yingzhen Qu, and Benjamin Kaduk for their detailed review of the model during WG and IESG.

13. Contributors

Danial Johari
Cisco Systems
Email: dajohari@cisco.com

Loa Andersson
Huawei Technologies
Email: loa@pi.nu

Jeff Tantsura
Apstra
Email: jefftant.ietf@gmail.com

Matthew Bocci
Nokia
Email: matthew.bocci@nokia.com

Reshad Rahman
Cisco Systems
Email: rrahman@cisco.com

Stephane Litkowski
Cisco Systems
Email: slitkows@cisco.com

14. Normative References

- [I-D.ietf-rtgwg-policy-model]
Qu, Y., Tantsura, J., Lindem, A., and X. Liu, "A YANG Data Model for Routing Policy Management", Work in Progress, Internet-Draft, draft-ietf-rtgwg-policy-model-09, 4 March 2020, <<https://tools.ietf.org/html/draft-ietf-rtgwg-policy-model-09>>.
- [RFC3478] Leelanivas, M., Rekhter, Y., and R. Aggarwal, "Graceful Restart Mechanism for Label Distribution Protocol", RFC 3478, DOI 10.17487/RFC3478, February 2003, <<https://www.rfc-editor.org/info/rfc3478>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,

- DOI 10.17487/RFC3688, January 2004,
<<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed.,
"LDP Specification", RFC 5036, DOI 10.17487/RFC5036,
October 2007, <<https://www.rfc-editor.org/info/rfc5036>>.
- [RFC5331] Aggarwal, R., Rekhter, Y., and E. Rosen, "MPLS Upstream
Label Assignment and Context-Specific Label Space",
RFC 5331, DOI 10.17487/RFC5331, August 2008,
<<https://www.rfc-editor.org/info/rfc5331>>.
- [RFC5443] Jork, M., Atlas, A., and L. Fang, "LDP IGP
Synchronization", RFC 5443, DOI 10.17487/RFC5443, March
2009, <<https://www.rfc-editor.org/info/rfc5443>>.
- [RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL.
Le Roux, "LDP Capabilities", RFC 5561,
DOI 10.17487/RFC5561, July 2009,
<<https://www.rfc-editor.org/info/rfc5561>>.
- [RFC5918] Asati, R., Minei, I., and B. Thomas, "Label Distribution
Protocol (LDP) 'Typed Wildcard' Forward Equivalence Class
(FEC)", RFC 5918, DOI 10.17487/RFC5918, August 2010,
<<https://www.rfc-editor.org/info/rfc5918>>.
- [RFC5919] Asati, R., Mohapatra, P., Chen, E., and B. Thomas,
"Signaling LDP Label Advertisement Completion", RFC 5919,
DOI 10.17487/RFC5919, August 2010,
<<https://www.rfc-editor.org/info/rfc5919>>.
- [RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS
Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010,
<<https://www.rfc-editor.org/info/rfc5920>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
the Network Configuration Protocol (NETCONF)", RFC 6020,
DOI 10.17487/RFC6020, October 2010,
<<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
and A. Bierman, Ed., "Network Configuration Protocol
(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
<<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure
Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011,
<<https://www.rfc-editor.org/info/rfc6242>>.

- [RFC6389] Aggarwal, R. and JL. Le Roux, "MPLS Upstream Label Assignment for LDP", RFC 6389, DOI 10.17487/RFC6389, November 2011, <<https://www.rfc-editor.org/info/rfc6389>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <<https://www.rfc-editor.org/info/rfc7277>>.
- [RFC7552] Asati, R., Pignataro, C., Raza, K., Manral, V., and R. Papneja, "Updates to LDP for IPv6", RFC 7552, DOI 10.17487/RFC7552, June 2015, <<https://www.rfc-editor.org/info/rfc7552>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <<https://www.rfc-editor.org/info/rfc8177>>.
- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <<https://www.rfc-editor.org/info/rfc8294>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.

- [RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <<https://www.rfc-editor.org/info/rfc8344>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <<https://www.rfc-editor.org/info/rfc8529>>.

15. Informative References

- [I-D.ietf-mpls-mldp-yang] Raza, K., Liu, X., Esale, S., Andersson, L., Tantsura, J., and S. Krishnaswamy, "YANG Data Model for MPLS mLDP", Work in Progress, Internet-Draft, draft-ietf-mpls-mldp-yang-06, 31 May 2019, <<https://tools.ietf.org/html/draft-ietf-mpls-mldp-yang-06>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC7307] Zhao, Q., Raza, K., Zhou, C., Fang, L., Li, L., and D. King, "LDP Extensions for Multi-Topology", RFC 7307, DOI 10.17487/RFC7307, July 2014, <<https://www.rfc-editor.org/info/rfc7307>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/info/rfc7951>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

Appendix A. Data Tree Example

This section contains an example of an instance data tree in the JSON encoding [RFC7951], containing both configuration and state data.

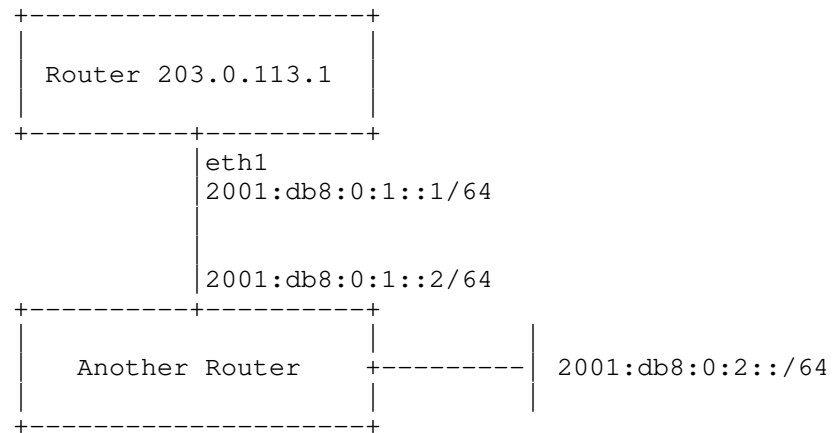


Figure 12: Example topology

The configuration instance data tree for Router 203.0.113.1 in the above figure could be as follows:

```

{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth1",
        "description": "An interface with LDP enabled.",
        "type": "iana-if-type:ethernetCsmacd",
        "ietf-ip:ipv6": {
          "address": [
            {
              "ip": "2001:db8:0:1::1",
              "prefix-length": 64
            }
          ]
        },
        "forwarding": true
      }
    ]
  },
  "ietf-routing:routing": {
    "router-id": "203.0.113.1",
    "control-plane-protocols": {

```

```

"control-plane-protocol": [
  {
    "type": "ietf-mpls-ldp:mpls-ldp",
    "name": "ldp-1",
    "ietf-mpls-ldp:mpls-ldp": {
      "global": {
        "address-families": {
          "ietf-mpls-ldp-extended:ipv6": {
            "enabled": true,
            "transport-address": "2001:db8:0:1::1"
          }
        }
      },
      "discovery": {
        "interfaces": {
          "interface": [
            {
              "name": "eth1",
              "address-families": {
                "ietf-mpls-ldp-extended:ipv6": {
                  "enabled": true
                }
              }
            }
          ]
        }
      }
    }
  }
]
}

```

Figure 13: Example Configuration data in JSON

The corresponding operational state data for Router 203.0.113.1 could be as follows:

```

{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth1",
        "description": "An interface with LDP enabled.",
        "type": "iana-if-type:ethernetCsmacd",
        "phys-address": "00:00:5e:00:53:01",

```

```
    "oper-status": "up",
    "statistics": {
      "discontinuity-time": "2018-09-10T15:16:27-05:00"
    },
    "ietf-ip:ipv6": {
      "forwarding": true,
      "mtu": 1500,
      "address": [
        {
          "ip": "2001:db8:0:1::1",
          "prefix-length": 64,
          "origin": "static",
          "status": "preferred"
        },
        {
          "ip": "fe80::200:5eff:fe00:5301",
          "prefix-length": 64,
          "origin": "link-layer",
          "status": "preferred"
        }
      ],
      "neighbor": [
        {
          "ip": "2001:db8:0:1::2",
          "link-layer-address": "00:00:5e:00:53:02",
          "origin": "dynamic",
          "is-router": [null],
          "state": "reachable"
        },
        {
          "ip": "fe80::200:5eff:fe00:5302",
          "link-layer-address": "00:00:5e:00:53:02",
          "origin": "dynamic",
          "is-router": [null],
          "state": "reachable"
        }
      ]
    }
  ],
  "ietf-routing:routing": {
    "router-id": "203.0.113.1",
    "interfaces": {
      "interface": [
        "eth1"
      ]
    }
  },
}
```



```

"control-plane-protocols": {
  "control-plane-protocol": [
    {
      "type": "ietf-mpls-ldp:mpls-ldp",
      "name": "ldp-1",
      "ietf-mpls-ldp:mpls-ldp": {
        "global": {
          "address-families": {
            "ietf-mpls-ldp-extended:ipv6": {
              "enabled": true,
              "transport-address": "2001:db8:0:1::1"
            }
          }
        },
        "discovery": {
          "interfaces": {
            "interface": [
              {
                "name": "eth1",
                "address-families": {
                  "ietf-mpls-ldp-extended:ipv6": {
                    "enabled": true,
                    "hello-adjacencies": {
                      "hello-adjacency": [
                        {
                          "adjacent-address":
                            "fe80::200:5eff:fe00:5302",
                          "flag": ["adjacency-flag-active"],
                          "hello-holdtime": {
                            "adjacent": 15,
                            "negotiated": 15,
                            "remaining": 9
                          },
                          "next-hello": 3,
                          "statistics": {
                            "discontinuity-time":
                              "2018-09-10T15:16:27-05:00"
                          },
                          "peer": {
                            "lsr-id": "203.0.113.2",
                            "label-space-id": 0
                          }
                        }
                      ]
                    }
                  }
                }
              ]
            }
          }
        }
      }
    }
  ]
}

```

```
]
}
},
"peers": {
  "peer": [
    {
      "lsr-id": "203.0.113.2",
      "label-space-id": 0,
      "label-advertisement-mode": {
        "local": "downstream-unsolicited",
        "peer": "downstream-unsolicited",
        "negotiated": "downstream-unsolicited"
      },
      "next-keep-alive": 5,
      "session-holdtime": {
        "peer": 180,
        "negotiated": 180,
        "remaining": 78
      },
      "session-state": "operational",
      "tcp-connection": {
        "local-address": "fe80::200:5eff:fe00:5301",
        "local-port": 646,
        "remote-address": "fe80::200:5eff:fe00:5302",
        "remote-port": 646
      },
      "up-time": 3438100,
      "statistics": {
        "discontinuity-time": "2018-09-10T15:16:27-05:00"
      }
    }
  ]
}
}
```

Figure 14: Example Operational data in JSON

Authors' Addresses

Kamran Raza (editor)
Cisco Systems

Canada
Email: skraza@cisco.com

Rajiv Asati
Cisco Systems
United States of America
Email: rajiva@cisco.com

Xufeng Liu
Volta Networks
United States of America
Email: xufeng.liu.ietf@gmail.com

Santosh Esale
Juniper Networks
United States of America
Email: sesale@juniper.net

Xia Chen
Huawei Technologies
China
Email: jescia.chenxia@huawei.com

Himanshu Shah
Ciena Corporation
United States of America
Email: hshah@ciena.com

Internet Engineering Task Force
Internet-Draft
Updates: 8029 (if approved)
Intended status: Standards Track
Expires: October 6, 2019

N. Akiya
Big Switch Networks
G. Swallow
Cisco Systems
S. Litkowski
B. Decraene
Orange
J. Drake
Juniper Networks
M. Chen
Huawei
April 04, 2019

Label Switched Path (LSP) Ping/Trace Multipath Support for
Link Aggregation Group (LAG) Interfaces
draft-ietf-mpls-lsp-ping-lag-multipath-08

Abstract

This document defines extensions to the MPLS Label Switched Path (LSP) Ping and Traceroute mechanisms as specified in RFC 8029. The extensions allow the MPLS LSP Ping and Traceroute mechanisms to discover and exercise specific paths of Layer 2 (L2) Equal-Cost Multipath (ECMP) over Link Aggregation Group (LAG) interfaces. Additionally, a mechanism is defined to enable determination of the capabilities of an LSR supported.

This document updates RFC8029.

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 BCP14 [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 <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 October 6, 2019.

Copyright Notice

Copyright (c) 2019 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. Terminology	3
1.2. Background	4
2. Overview of Solution	4
3. LSR Capability Discovery	6
3.1. Initiator LSR Procedures	7
3.2. Responder LSR Procedures	7
4. Mechanism to Discover L2 ECMP Multipath	7
4.1. Initiator LSR Procedures	7
4.2. Responder LSR Procedures	8
4.3. Additional Initiator LSR Procedures	10
5. Mechanism to Validate L2 ECMP Traversal	11
5.1. Incoming LAG Member Links Verification	11
5.1.1. Initiator LSR Procedures	11
5.1.2. Responder LSR Procedures	12
5.1.3. Additional Initiator LSR Procedures	12
5.2. Individual End-to-End Path Verification	14
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	20
11. Rate Limiting On Echo Request/Reply Messages	21
12. Security Considerations	21
13. IANA Considerations	21
13.1. LSR Capability TLV	21
13.1.1. LSR Capability Flags	22
13.2. Local Interface Index Sub-TLV	22
13.2.1. Interface Index Flags	22
13.3. Remote Interface Index Sub-TLV	23
13.4. Detailed Interface and Label Stack TLV	23
13.4.1. Sub-TLVs for TLV Type TBD4	23
13.4.2. Interface and Label Stack Address Types	24
13.5. DS Flags	24
14. Acknowledgements	24
15. References	25
15.1. Normative References	25
15.2. Informative References	25
Appendix A. LAG with intermediate L2 Switch Issues	26
A.1. Equal Numbers of LAG Members	26
A.2. Deviating Numbers of LAG Members	26
A.3. LAG Only on Right	27
A.4. LAG Only on Left	27
Authors' Addresses	27

1. Introduction

1.1. Terminology

The following acronyms/terms are used in this document:

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

1.2. Background

The MPLS Label Switched Path (LSP) Ping and Traceroute mechanisms [RFC8029] are powerful tools designed to diagnose all available Layer 3 (L3) paths of LSPs, including diagnostic coverage of L3 Equal-Cost Multipath (ECMP). In many MPLS networks, Link Aggregation Group (LAG) as defined in [IEEE802.1AX], which provides Layer 2 (L2) ECMP, is often used for various reasons. MPLS LSP Ping and Traceroute tools were not designed to discover and exercise specific paths of L2 ECMP. This raises a limitation for the following scenario when an LSP traverses over a LAG:

- o Label switching over some member links of the LAG is successful, but fails over other member links of the LAG.
- o MPLS echo request for the LSP over the LAG is load balanced on one of the member links 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 the 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 a problematic LAG interface. It is, thus, desirable to extend the MPLS LSP Ping and Traceroute to have deterministic diagnostic coverage of LAG interfaces.

The need for a solution of this problem was motivated by issues encountered in live networks.

2. Overview of Solution

This document defines a new TLV to discover the capabilities of a responder LSR and extensions for use with the MPLS LSP Ping and Traceroute mechanisms to describe Multipath Information for individual LAG member links, thus allowing MPLS LSP Ping and Traceroute to discover and exercise specific paths of L2 ECMP over LAG interfaces. The reader is expected to be familiar with mechanics Downstream Detailed Mapping TLV (DDMAP) described in Section 3.4 of [RFC8029].

The solution consists of the MPLS echo request containing a DDMAP TLV and the new LSR Capability TLV to indicate that separate load balancing information for each L2 nexthop over LAG is desired in the MPLS echo reply. The Responder LSR places the same LSR Capability TLV in the MPLS echo reply to provide acknowledgement back to the initiator LSR. It also adds, for each downstream LAG member, load balance information (i.e., multipath information and interface

index). This mechanism is applicable to all types of LSPs which can traverse over LAG interfaces. Many LAGs are built from p2p links, with router X and router X+1 having direct connectivity and the same number of LAG members. It is possible to build LAGs asymmetrically by using Ethernet switches between two routers. Appendix A lists some use cases for which the mechanisms defined in this document may not be applicable. Note that the mechanisms described in this document do 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).

The following figure and description provides an example using an LDP network.

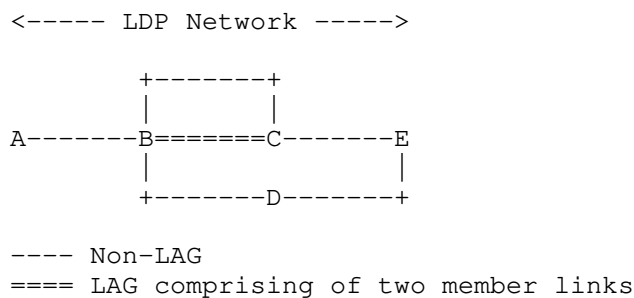


Figure 1: Example LDP Network

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:

- o In Section 3, a mechanism to discover capabilities of responder LSRs;
- o In Section 4, a mechanism to discover L2 ECMP multipath information;
- o In Section 5, a mechanism to validate L2 ECMP traversal;

- o In Section 6, the LSR Capability TLV;
- o In Section 7, the LAG Description Indicator flag;
- o In Section 8, the Local Interface Index Sub-TLV;
- o In Section 9, the Remote Interface Index Sub-TLV;
- o In Section 10, the Detailed Interface and Label Stack TLV;

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 have been many extensions to the MPLS Ping and Traceroute mechanisms over the years. Thus it is often useful, and sometimes necessary, for the initiator LSR to deterministically disambiguate the differences between:

- o The responder LSR sent the MPLS echo reply message with contents C because it has feature X, Y and Z implemented.
- o 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.
- o 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, if it knows the LSR Capability TLV, then it 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. Otherwise, an MPLS echo reply must be sent back to the initiator LSR with the return code set to "One or more of the TLVs was not understood", according to the rules as defined Section 3 of [RFC8029]. Then the initiator LSR can send another MPLS echo request without including the LSR Capability TLV.

It is RECOMMENDED that implementations supporting the LAG Multipath extensions defined in this document include the LSR Capability TLV in MPLS echo request messages.

3.1. Initiator LSR Procedures

If an initiator LSR does not know what capabilities a responder LSR can support, it can send an MPLS echo request message and carry the LSR Capability TLV to the responder to discover the capabilities that the responder LSR can support.

3.2. Responder LSR Procedures

When a responder LSR received an MPLS echo request message that carries the LSR Capability TLV, the following procedures are used:

If the responder knows how to process the LSR Capability TLV, the following procedures are used:

- o The responder LSR MUST include the LSR Capability TLV in the MPLS echo reply message.
- o If the responder LSR understands the "LAG Description Indicator flag":
 - * Set the "Downstream LAG Info Accommodation flag" if the responder LSR is capable of describing outgoing LAG member links separately; otherwise, clear the "Downstream LAG Info Accommodation flag".
 - * Set the "Upstream LAG Info Accommodation flag" if responder LSR is capable of describing incoming LAG member links separately; otherwise, clear the "Upstream LAG Info Accommodation flag".

4. Mechanism to Discover L2 ECMP Multipath

4.1. Initiator LSR Procedures

Through the "LSR Capability Discovery" as defined in Section 3, the initiator LSR can understand whether the responder LSR can describe incoming/outgoing LAG member links separately in the DDMAP TLV.

Once the initiator LSR knows that a responder can support this mechanism, then it sends an MPLS echo request carrying a DDMAP TLV with the "LAG Description Indicator flag" (G) set to the responder LSR. The "LAG Description Indicator flag" (G) indicates that separate load balancing information for each L2 nexthop over a LAG is

desired in the MPLS echo reply. The new "LAG Description Indicator flag" is described in Section 7.

4.2. Responder LSR Procedures

When a responder LSR received an MPLS echo request message with the "LAG Description Indicator flag" set in the DDMAP TLV, if the responder LSR understands the "LAG Description Indicator flag" and is capable of describing outgoing LAG member links separately, the following procedures are used, regardless of whether or not outgoing interfaces include LAG interfaces:

- o For each downstream that is a LAG interface:
 - * The responder LSR MUST include a DDMAP TLV when sending the MPLS echo reply. There is a single DDMAP TLV for the LAG interface, with member links described using sub-TLVs.
 - * The responder LSR MUST set the "LAG Description Indicator flag" in the DS Flags field of the DDMAP TLV.
 - * In the DDMAP TLV, the Local Interface Index Sub-TLV, Remote Interface Index Sub-TLV and Multipath Data Sub-TLV are used to describe each LAG member link. All other fields of the DDMAP TLV are used to describe the LAG interface.
 - * For each LAG member link of the 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 the received DDMAP TLV 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 TLV. The order of the Sub-TLVs in the DDMAP TLV for a LAG member link MUST be Local Interface Index Sub-TLV immediately followed by Multipath Data Sub-TLV except as follows. 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 TLV 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. The block of local interface index, (optional remote interface index) and multipath data sub-TLVs for each member link MUST appear adjacent to each other in order of increasing local interface index.

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

```

      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 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
~      DDMAP fields describing LAG interface with DS Flags G set      ~
+-----+-----+-----+-----+-----+-----+-----+-----+
| Local Interface Index Sub-TLV of LAG member link #1                |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Remote Interface Index Sub-TLV of LAG member link #1                |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Multipath Data Sub-TLV LAG member link #1                          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Local Interface Index Sub-TLV of LAG member link #2                |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Remote Interface Index Sub-TLV of LAG member link #2                |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Multipath Data Sub-TLV LAG member link #2                          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Label Stack Sub-TLV                    |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 2: Example of DDMAP in MPLS Echo Reply

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 TLV. The value of Multipath Length field of the Multipath Data Sub-TLV is set to zero.

4.3. Additional Initiator LSR Procedures

The procedures above allow an initiator LSR to:

- o Identify whether or not the responder LSR can describe outgoing LAG member links separately, by looking at the LSR Capability TLV.
- o Utilize the value of the "LAG Description Indicator flag" in DS Flags to identify whether each received DDMAP TLV describes a LAG interface or a non-LAG interface.
- o 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:

- o The Local Interface Index Sub-TLVs MUST be removed in the sending DDMAP.
- o 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.
- o The Multipath Data Sub-TLVs MUST be updated to include just one Multipath Data Sub-TLV. The initiator LSR MAY just keep 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.
- o All other fields of the DDMAP are to comply with procedures described in [RFC8029].

Figure 3 is an example that shows how to use the DDMAP TLV to notify which member link (link #1 in the example) will be chosen to send the MPLS echo request message to the next downstream LSR:

```

      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 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
~   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 construct a DDMAP for a downstream LAG. The Remote Interface Index Sub-TLVs that describes the incoming LAG member links of the downstream LSR is optional, because 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, 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

An MPLS echo request carrying a DDMAP TLV with the "Interface and Label Stack Object Request flag" and "LAG Description Indicator flag" set is sent 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

When received an echo request with the "LAG Description Indicator flag" set, a responder LSR that understands the "LAG Description Indicator flag" and is capable of describing incoming LAG member link SHOULD use the following procedures, regardless of whether or not incoming interface was a LAG interface:

- o When the "I" flag ("Interface and Label Stack Object Request flag") of the DDMAP TLV in the received MPLS echo request is set:
 - * The responder LSR MUST add the Detailed Interface and Label Stack TLV (described in Section 10) in the MPLS echo reply.
 - * If the incoming interface is 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:

- o Utilize the Incoming Interface Index Sub-TLV in the Detailed Interface and Label Stack TLV to derive, if the incoming interface is 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:

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

Depending on the LAG traffic division algorithm, the messages may or may not traverse different member links. The 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 entropies examined. 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 logic, the initiator LSR can perform the following checks in a scenario where the initiator LSR 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 member links.

The initiator LSR sends two MPLS echo request messages to traverse the two LAG member links at TTL=n+1:

o Success case:

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

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

o Error case:

- * One MPLS echo request message reaches TTL=n+1 on an LAG member link 1.
- * The other MPLS echo request message also reaches TTL=n+1 on an LAG member link 1.
- * One or both MPLS echo request messages cannot reach the immediate downstream LSR on whichever link.

One or two MPLS echo request messages sent by the initiator LSR cannot reach the immediate downstream LSR, or the two MPLS echo request messages reach at the immediate downstream LSR from the same LAG member link.

Note that the above defined procedures will provide a deterministic result for LAG interfaces that are back-to-back connected between LSRs (i.e. no L2 switch in between). If there is a L2 switch between the LSR at TTL=n and the LSR at TTL=n+1, there is no guarantee that traversal of every LAG member link at TTL=n will result in reaching from different interface 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 for the downstream responder LSR are then updated to include the comparison of the incoming LAG member link to the interface index described in the Remote Interface Index Sub-TLV in the DDMAP TLV. Failure of this comparison results in the return code being set to "Downstream Mapping Mismatch (5)".

6. LSR Capability TLV

This document defines a new TLV which is referred to as the "LSR Capability TLV". It 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 TLV. The presence of an LSR Capability TLV in an MPLS echo request message is a request that a responder LSR includes an LSR Capability TLV in the MPLS echo reply message, with the LSR Capability TLV describing features and extensions that the responder LSR supports.

The format of the LSR Capability TLV is as below:

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

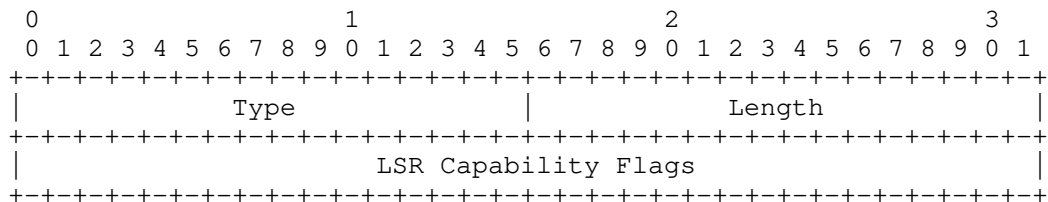


Figure 4: LSR Capability TLV

Where:

The Type field is 2 octets in length and the value is TBD1.

The Length field is 2 octets in length, and the value is 4.

The "LSR Capability Flags" field is 4 octets in length, this document defines the following flags:

```

      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 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Reserved (Must Be Zero)                               |U|D|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

This document defines two flags. The unallocated flags MUST be set to zero when sending and ignored on receipt. Both the U and the D flag MUST be cleared in the MPLS echo request message when sending, and ignored on receipt. Neither, either or both the U and the D flag MAY be set in the MPLS echo reply message.

Flag	Name and Meaning
U	Upstream LAG Info Accommodation
D	Downstream LAG Info Accommodation

An LSR sets this flag when the LSR is capable of describing a LAG member link in the Incoming Interface Index Sub-TLV in the Detailed Interface and Label Stack TLV.

An LSR sets this flag when the LSR 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

This document defines a new flag, the "G" flag (LAG Description Indicator), in the DS Flags field of the DDMAP TLV.

The "G" flag 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.

The "G" flag is defined as below:

The Bit Number is TBD5.

```

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

```

RFC-Editor-Note: Please update above figure to place the G flag in the bit number TBD5.

Flag	Name and Meaning
G	LAG Description Indicator

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

8. Local Interface Index Sub-TLV

The Local Interface Index Sub-TLV describes the interface index assigned by the local LSR to an egress interface. One or more Local Interface Index sub-TLVs MAY appear in a DDMAP TLV.

The format of the Local Interface Index Sub-TLV is below:

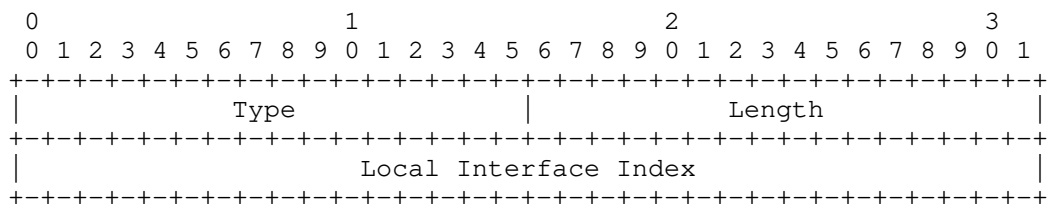


Figure 5: Local Interface Index Sub-TLV

Where:

- o The "Type" field is 2 octets in length, the value is TBD2.
- o The "Length" field is 2 octets in length, and the value is 4.
- o The "Local Interface Index" field is 4 octets in length, it is an interface index assigned by a local LSR to an egress interface. It's normally an unsigned integer and in network byte order.

9. Remote Interface Index Sub-TLV

The Remote Interface Index Sub-TLV is an optional TLV, it describes the interface index assigned by a downstream LSR to an ingress interface. One or more Remote Interface Index sub-TLVs MAY appear in a DDMAP TLV.

The format of the Remote Interface Index Sub-TLV is as below:

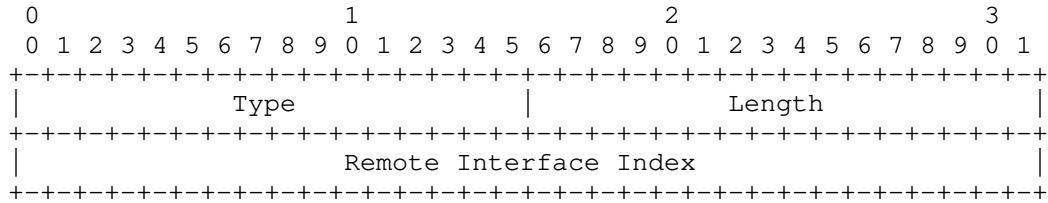


Figure 6: Remote Interface Index Sub-TLV

Where:

- o The "Type" field is 2 octets in length, and the value is TBD3.
- o The "Length" field is 2 octets in length, and the value is 4.
- o The "Remote Interface Index" is 4 octets in length, it is an interface index assigned by a downstream LSR to an ingress interface. It's normally an unsigned integer and in network byte order.

10. Detailed Interface and Label Stack TLV

The "Detailed Interface and Label Stack" object is a TLV that MAY be included in an 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 into the MPLS echo reply message. 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:

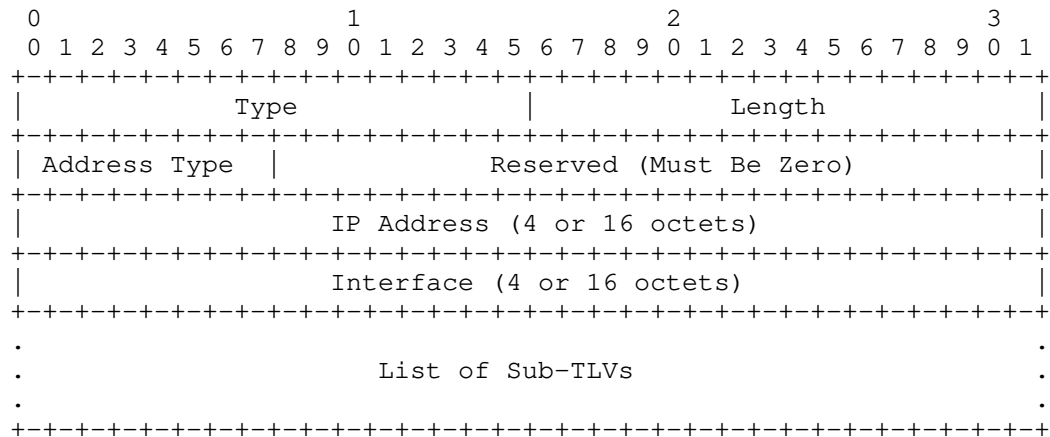


Figure 7: Detailed Interface and Label Stack TLV

The Detailed Interface and Label Stack TLV format is derived from the Interface and Label Stack TLV format (from [RFC8029]). Two changes are introduced. The first is that the label stack is converted into a sub-TLV. The second is that a new sub-TLV is added to describe an interface index. The other fields of Detailed Interface and Label Stack TLV have the same use and meaning as in [RFC8029]. A summary of these fields 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 length of the initial part of the TLV is listed as "K Octets". The Address Type is set to one of the following values:

Type #	Address Type	K Octets
-----	-----	-----
1	IPv4 Numbered	16
2	IPv4 Unnumbered	16
3	IPv6 Numbered	40
4	IPv6 Unnumbered	28

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 and 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. Two sub-TLVs are defined:

Sub-Type	Sub-TLV Name
1	Incoming Label stack
2	Incoming Interface Index

10.1.1. Incoming Label Stack Sub-TLV

The Incoming Label Stack sub-TLV contains the label stack as received by an 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 its format is as below:

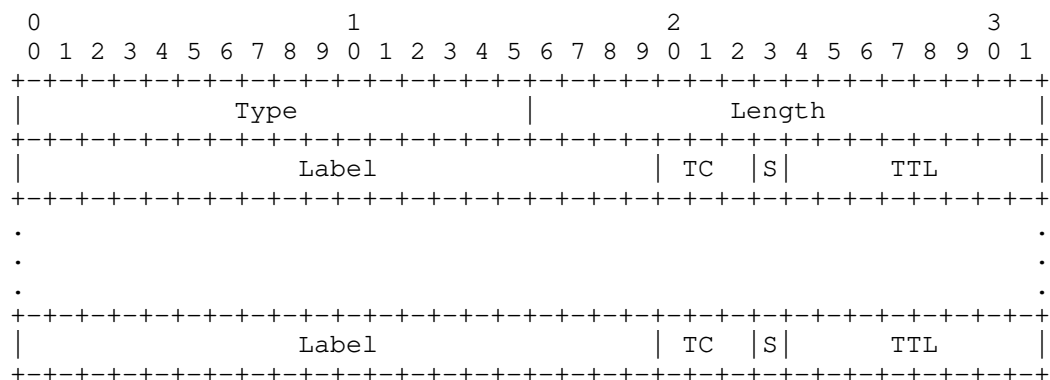


Figure 8: Incoming Label Stack Sub-TLV

10.1.1.2. 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 a local LSR to the interface which received the MPLS echo request message.

Incoming Interface Index Sub-TLV Type is 2. Length is 8, and its format is as below:

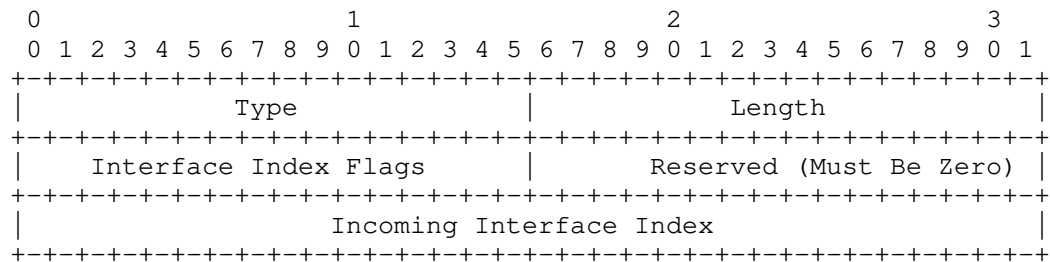
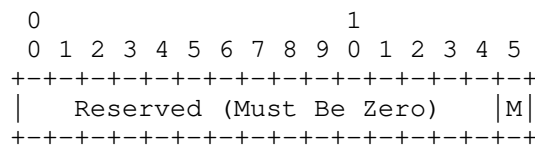


Figure 9: Incoming Interface Index Sub-TLV

Interface Index Flags

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



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

Flag	Name and Meaning
M	LAG Member Link Indicator

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. It's normally an unsigned integer and in network byte order.

11. Rate Limiting On Echo Request/Reply Messages

For an LSP path, it may be over several LAGs. Each LAG may have many member links. To exercise all the links, many Echo Request/Reply messages will be sent in a short period. It's possible that those messages may traverse a common path as a burst. Under some circumstances this might cause congestion at the common path. To avoid potential congestion, it is RECOMMENDED that implementations to randomly delay the Echo Request and Reply messages at the Initiating LSRs and Responder LSRs. Rate limiting of ping traffic is further specified in [RFC8029] (Section 5) and [RFC6425] (Section 4.1) which apply to this document as well.

12. Security Considerations

This document extends LSP Traceroute mechanism [RFC8029] to discover and exercise L2 ECMP paths to determine problematic member link(s) of a LAG. These on-demand diagnostic mechanisms are used by an operator within an MPLS control domain.

[RFC8029] reviews the possible attacks and approaches to mitigate possible threats when using these mechanisms.

To prevent leakage of vital information to untrusted users, a responder LSR MUST only accept MPLS echo request messages from designated trusted sources via filtering source IP address field of received MPLS echo request messages. As noted in [RFC8029], spoofing attacks only have a small window of opportunity. If these messages are indeed hijacked (non-delivery) by an intermediate node, the use of these mechanisms will determine the data plane is not working (as it should). Hijacking of a responder node such that it provides a legitimate reply would involve compromising the node itself and the MPLS control domain. [RFC5920] provides additional MPLS network-wide operation recommendations to avoid attacks and recommendations to follow. Please note that source IP address filtering provides only a weak form of access control and is not, in general, a reliable security mechanism. Nonetheless, it is required here in the absence of any more robust mechanisms that might be used.

13. IANA Considerations

13.1. LSR Capability TLV

The IANA is requested to assign new value TBD1 (from the range 4-16383) for LSR Capability TLV from the "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry.

Value -----	Meaning -----	Reference -----
TBD1	LSR Capability TLV	this document

13.1.1. 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

Bit number	Name -----	Reference -----
31	D: Downstream LAG Info Accommodation	this document
30	U: Upstream LAG Info Accommodation	this document
0-29	Unassigned	

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

13.2. Local Interface Index Sub-TLV

The IANA is requested to assign new value TBD2 (from the range 4-16383) 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.

Value -----	Meaning -----	Reference -----
TBD2	Local Interface Index Sub-TLV	this document

13.2.1. 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

Bit number	Name -----	Reference -----
15	M: LAG Member Link Indicator	this document
0-14	Unassigned	

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:

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

13.3. 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.

Value	Meaning	Reference
-----	-----	-----
TBD3	Remote Interface Index Sub-TLV	this document

13.4. Detailed Interface and Label Stack TLV

The IANA is requested to assign new value TBD4 (from the range 4-16383) 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]).

Value	Meaning	Reference
-----	-----	-----
TBD4	Detailed Interface and Label Stack TLV	this document

13.4.1. 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.

Sub-Type	Name	Reference
-----	-----	-----
1	Incoming Label Stack	this document
2	Incoming Interface Index	this document
3-16383	Unassigned (mandatory TLVs)	
16384-31743	Specification Required	
32768-49161	Unassigned (optional TLVs)	
49162-64511	Specification Required	

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

13.4.2. Interface and Label Stack Address Types

Since the "Detailed Interface and Label Stack TLV" shares the "Interface and Label Stack Address Types" with the "Interface and Label Stack TLV". IANA is requested to update the "Interface and Label Stack Address Types" registry to reflect this.

For example, change the registry name to "Interface and Label Stack and Detailed Interface and Label Stack Address Types", and add a reference to this document.

13.5. 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].

Bit number	Name	Reference
-----	-----	-----
TBD5	G: LAG Description Indicator	this document

14. Acknowledgements

The authors would like to thank Nagendra Kumar, 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 mechanisms. The suggestion by Yimin Shen to create two separate validation procedures had a big impact to the contents of this document.

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

15.2. Informative References

- [IANA-MPLS-LSP-PING] IANA, "Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters", <<http://www.iana.org/assignments/mpls-lsp-ping-parameters/mpls-lsp-ping-parameters.xhtml>>.
- [IEEE802.1AX] IEEE Std. 802.1AX, "IEEE Standard for Local and metropolitan area networks - Link Aggregation", November 2008.
- [RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010, <<https://www.rfc-editor.org/info/rfc5920>>.

- [RFC6425] Saxena, S., Ed., Swallow, G., Ali, Z., Farrel, A., Yasukawa, S., and T. Nadeau, "Detecting Data-Plane Failures in Point-to-Multipoint MPLS - Extensions to LSP Ping", RFC 6425, DOI 10.17487/RFC6425, November 2011, <<https://www.rfc-editor.org/info/rfc6425>>.
- [RFC7439] George, W., Ed. and C. Pignataro, Ed., "Gap Analysis for Operating IPv6-Only MPLS Networks", RFC 7439, DOI 10.17487/RFC7439, January 2015, <<https://www.rfc-editor.org/info/rfc7439>>.

Appendix A. LAG with intermediate L2 Switch Issues

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

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 Router 1 (R1) to intermediate L2 switch (S1) can get load balanced by S1 towards Router 2 (R2). Therefore, MPLS echo request messages traversing a specific LAG member from R1 to S1 can actually reach R2 via any of the LAG members, and the sender of MPLS echo request messages has 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. (Notes, AFAICT there's not a better option than "try a bunch of entropy labels and see what responses you can get back" and that's the same remedy in all the described topologies.)

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.decraene@orange.com

John E. Drake
Juniper Networks

Email: jdrake@juniper.net

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

MPLS WG
Internet-Draft
Intended status: Experimental
Expires: August 18, 2021

K. Kompella
Juniper Networks, Inc.
L. Contreras
Telefonica
February 14, 2021

Resilient MPLS Rings
draft-ietf-mpls-rmr-14

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][RFC8174].

This document is classified as an Experimental RFC. The parameters of this experiment have yet to be defined: how long the experiment runs, what criteria determine that the experiment is over -- does the doc then become Standards Track or Historical, etc. A future update will document these parameters.

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 18, 2021.

Copyright Notice

Copyright (c) 2021 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. Definitions	3
1.2. Changes from -12 in response to reviews	5
2. Motivation	6
3. Theory of Operation	6
3.1. Provisioning	7
3.2. Ring Nodes	7
3.3. Ring Links and Directions	8
3.3.1. Express Links	9
3.4. Ring LSPs	9
3.5. Installing Primary LFIB Entries	9
3.6. Protection	10
3.7. Installing FRR LFIB Entries	11
4. Autodiscovery	11
4.1. Overview	11
4.2. Ring Announcement Phase	13
4.3. Mastership Phase	14
4.4. Ring Identification Phase	14
4.5. Ring Changes	15
5. Ring OAM	16
6. Advanced Topics	16
6.1. Beyond the Ring	16
6.2. Half-rings	18
6.3. Hub Node Resilience	18
7. Security Considerations	18
8. Acknowledgments	19
9. IANA Considerations	19
10. References	19
10.1. Normative References	19
10.2. Informative References	19
Authors' Addresses	20

1. Introduction

Rings are a very common topology either at infrastructure level (e.g., physical ring fiber deployments in Layer 1 networks) or node interconnection structures (e.g., loops created in bridged interconnected infrastructures [IEEE.802.1D_2004]). 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, 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]. RMR LSPs can also be signaled with IGP [RFC8402]; that will be described in a future document.

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 with the purpose of using them for protection.

In some other networking situations (e.g., interconnection of bridges), those rings could create loops making the network inoperable, and thus needing from signaling mechanisms (such the Spanning Tree Protocol) for preventing and eliminating such loops [IEEE.802.1D_2004]. Here it is followed a dual approach where the signaling methods are precisely created for automatically identifying and defining rings where efficiently create LSPs adapted to the formed ring topology.

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 $\{(R_i, R_{i+1}) \text{ and } (R_{i+1}, R_i), 0 \leq i < n-1\}$ 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.

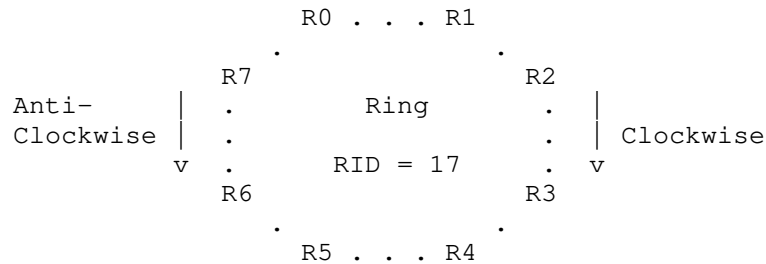


Figure 1: Ring with 8 nodes

The following terminology is used for ring LSPs:

Ring ID (RID): A non-negative number. When the RID identifies a ring, it must be positive and unique in some scope of a Service Provider's network. An RID of zero, when assigned to a node, indicates that the node must behave in "promiscuous mode" (see Section 3.2). 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.

1.2. Changes from -12 in response to reviews

[Note to RFC Editor: this (sub-)section to be removed prior to publication.]

Reqs Lang: updated (response to Gen-ART review [Gen])

Section 1: updated "transport networks" to "Layer 1 networks" (response to Transport Area review [TAR])

Sec 1: replaced SPRING with IGP (response to OPS directorate [OPS])

Sec 1: rephrased last sentence [TAR]

Sec 2: added para on control plane resilience [TAR]

Sec 3.1: typo fixed [Gen]

Sec 3.2: added figure, caveats for promiscuous mode (response to Security Area Directorate review [SAD])

Sec 3.5: updated reference [OPS]

Sec 3.6: updated text on node protection, TTL [OPS]

Sec 4.1: changed Ring Neighbor TLV/flags to Ring Link TLV/flags; changed SPRING to IGP [OPS]

Sec 4.1: clean up [Gen]

Sec 4.2: updated text on timers T1, T2 [SAD]

Sec 4.3, 4.4: rewrote sections on Mastership, Ring Identification Phases for clarity [OPS]

Sec 4.5: removed "and" [Gen]

Sec 5: updated text on timers [TAR]

New Sec 6.1: added text on traffic transiting a ring [OPS]

Sec Cons: added text on compromised nodes [SAD]

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.

Note that this document addresses data plane resilience. Control plane resilience, and robustness of protocol messaging, is managed by the protocols being used here (IS-IS, OSPF, LDP and RSVP-TE) and not described in this document.

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 policy 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 .

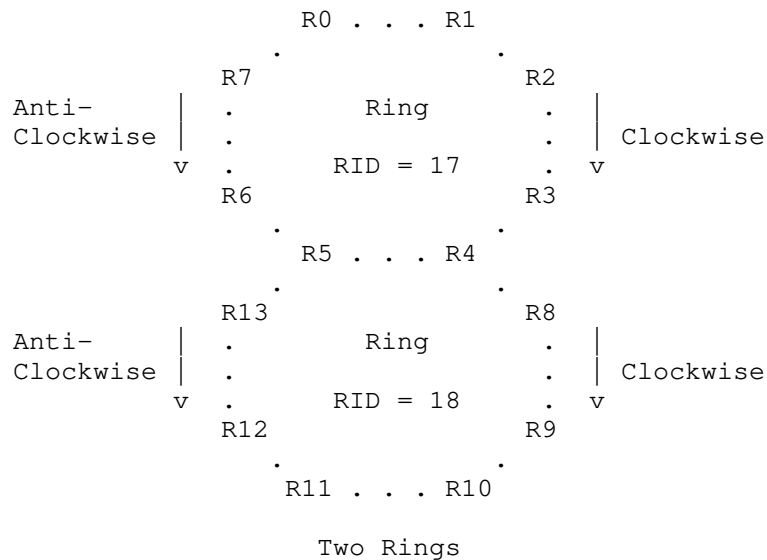
3.1. Provisioning

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 overridden 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. However, this feature should be used with care. Consider the following:



If R3 and R6 are configured with RID 17, R8 and R13 with RID 18, and all other nodes with RID 0, this will end up as two rings with R4 and R5 in both. However, other permutations of RID configurations could easily end up with all nodes being in both rings 17 and 18, whereupon the maximal ring will consist of R0 to R4, R8 to R13, R5 to R7 (and the link from R4 to R5 will be an express link). In cases such as these, one should eschew promiscuous mode in favor of simply configuring all nodes with the appropriate RIDs.

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 "auto-bundled" 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 may 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-neighbor ring nodes; often, this may be the result of optically bypassing ring nodes.

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 .

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 ([RFC8029]), 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 Ultimate Hop Popping, 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. 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.

Approaches 1 and 2 work for traffic that remains on the ring or terminates on a ring node (see Section 6.1); for traffic transiting the ring, playing with TTL may affect forwarding beyond the ring. Approach 3 is the most general and is the one we advocate; however, this will require the allocation and definition of a new special purpose label.

3.7. 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.

Say R1 receives label L42 from R2 to reach R4 in the clockwise direction, and receives label L40 from R0 to reach R4 in the anti-clockwise direction. Say R1 also receives label L52 from R2 to reach R5 in the clockwise direction, and receives label L50 from R0 to reach R5 in the anti-clockwise direction. R1 makes the following LFIB entries:

Dest	CW/NH	CW FRR/NH	AC/NH	AC FRR/NH
...				
R4	L42/R2	L40/R0	L40/R0	L42/R2
R5	L52/R2	L50/R0	L50/R0	L52/R2
...				

R1's LFIB

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.

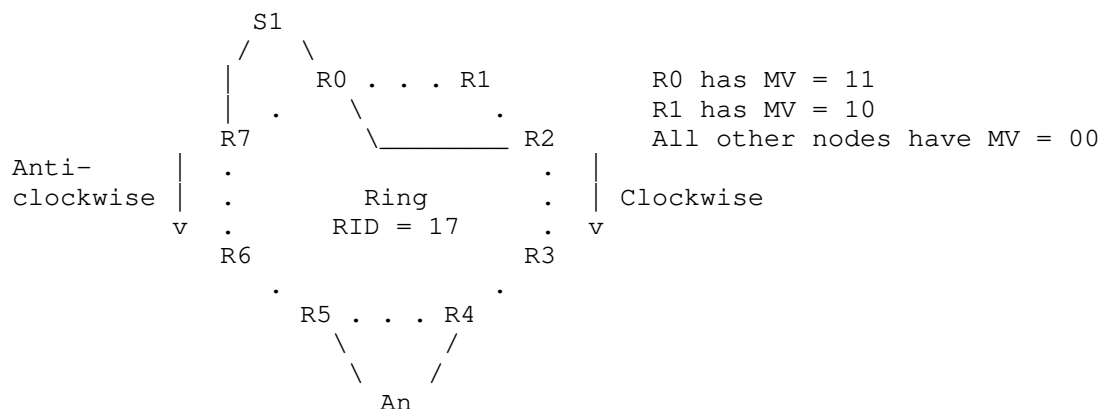


Figure 2: Ring with non-ring nodes and links

We use three concepts below:

ring nodes: all nodes that announce ring node TLVs with a given RID.

IGP neighbors: all nodes which are IGP neighbors of a given node.

ring neighbors: ring nodes that are IGP neighbors of a given node. Exactly one is the CW neighbor and one is the AC neighbor; all other ring neighbors are express neighbors.

In Figure 2, R0 through R7 are ring nodes belonging to ring 17. R0 has IGP neighbors R1, R2, R7 and S1. R0 has ring neighbors R1 (CW), R2 (express) and R7 (AC). Autodiscovery aims to identify ring nodes of a given ring, ring neighbors of each ring node, and the CW and AC node for each ring node.

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

[My Intf Inx][Rem Intf Inx][RID 1][Flags for RID 1]
[RID 2][Flags for RID 2]...

Ring Link Sub-TLV Format

```

0                                     1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|MV | SS | SO | MBZ | SU | M |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
MV: Mastership Value
SS: Supported Signaling Protocols
    (100 = RSVP-TE; 010 = LDP; 001 = IGP)
MBZ: Must be zero
SO: Supported OAM Protocols (100 = BFD; 010 = CFM; 001 = EFM)
SU: Signaling Protocol to Use (00: none; 01: LDP; 10: RSVP-TE;
    11: IGP)
M : Elected Master (0 = no, 1 = yes)

```

Flags for a Ring Node TLV

```

0                                     1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|RD |OAM| MBZ |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
RD: Ring Direction (00 = none; 01 = CW; 10 = AC; 11 = express)
OAM: OAM Protocol to use (00 = none; 01 = BFD; 10 = CFM; 11 = EFM)
MBZ: Must be zero

```

Flags for a Ring Link TLV

4.2. Ring Announcement Phase

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; this timer is to allow each node time to hear from all other nodes in the ring. [The settings for timers T1 and T2 (below) are particular to the specific IGP used for signaling; they will be discussed in the IGP document that defines the ring node/link TLVs.] The settings for timers T1 and T2 (below) will be discussed in the IGP document that defines the ring node/link TLVs.]

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.

4.3. Mastership Phase

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 as follows. N examines the MV value of all ring nodes and selects those with the highest MV value. Among these nodes, N finds the node with the lowest loopback address. If that node is N, N declares itself master to the entire ring by readvertising 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.

Barring software bugs or malicious code, the principal reason for multiple nodes for setting their M bit is late-arriving ring announcements. Say nodes N1 and N2 have the highest mastership values, and N1 has the lowest loopback address, while N2 has the second lowest loopback address. If N1 makes its ring announcement just as N2's T1 timer fires, both N1 and N2 will think they are the master (since N2 will not have heard N1's announcement in time). However, in the next round, N2 will realize that N1 is indeed the master. In the worst case, the mastership phase will occur as many times as there are nodes in the ring.

4.4. Ring Identification Phase

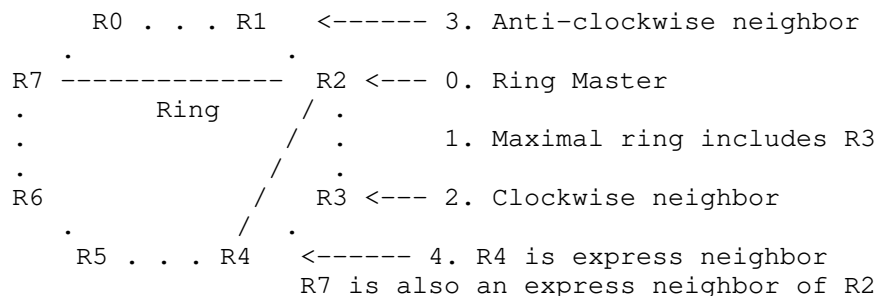


Figure 3: Ring Identification

When there is exactly one ring master M (here, R2), 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.

The Ring Identification Phase proceeds as follows:

1. M identifies all ring nodes for ring RID, i.e., those that have announced ring node TLVs with the ring ID = RID.
2. M computes a maximal ring among these nodes.
3. Based on that, M picks a CW neighbor and an AC neighbor.
4. M then inserts ring link TLVs with ring direction CW for each link to its CW neighbor; M also inserts a ring link TLV with direction AC for each link to its AC neighbor. (Note that there may be multiple links from M to each of its neighbors.)
5. Finally, M determines its express links. These are links to IGP neighbors that are ring nodes but neither the CW or AC neighbor. M advertises ring link TLVs for express links by setting the link direction to "express link".

This process passes on to the CW neighbor X as follows:

1. Each node Y listens for ring link TLVs. The set of nodes S consists of those that have announced ring link TLVs.
2. If a node Z announces a ring link TLV with Y as the CW neighbor, then Y is next.

X follows the same procedure as M with two small changes:

1. when X computes a maximal ring, it MUST include all nodes in S.
2. X knows its AC neighbor (Z above), and doesn't have to pick it.

Here, R2 (the master) knows R0 through R7 are ring nodes (Step 1). R1, R3, R4 and R7 are its ring neighbors. R2 computes a maximal ring (Step 2). It then picks R3 as its CW neighbor and R1 as its AC neighbor (Step 3). Finally, it declares the links to R4 and R7 as express links (Step 5).

4.5. Ring Changes

The main changes to a ring are:

ring link addition;

ring link deletion;
ring node addition;
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.

The addition of a new ring node can also be handled incrementally.

5. 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 that of the protocol chosen.

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. Each node chooses the hello interval, the choice of which should be based on the size of the ring (as each node would have to send out twice that many hello messages every interval) and the desired failure detection time.

6. Advanced Topics

6.1. Beyond the Ring

The discourse above discusses traffic that originates and terminates on a ring. However, in many cases, traffic may come originate on a ring node and terminate at a non-ring node; other traffic may originate on a non-ring node and terminate on a ring node; and in yet other cases, traffic may transit a ring, i.e., originate on a non-ring node, arrive at a ring node, traverse the ring, and leave for a

non-ring destination. This section discusses these cases, and how traffic traversing a ring can profit from ring protection.

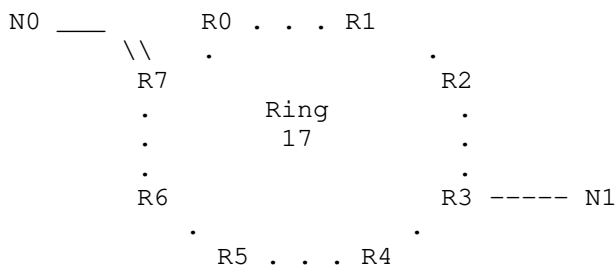


Figure 4: Beyond the Ring

In all these cases, the "end-to-end" path needs to be either stitched with, or overlaid on, the ring path. The latter approach is recommended, using hierarchy in both the control and data planes. In the figure above, traffic from N0 to N1 (both non-ring nodes) traverses Ring 17. If nodes outside Ring 17 use LDP to signal LSPs, here's one way to accomplish this: R7 and R3 have targeted LDP sessions to exchange labels. The following LDP label exchanges occur (among others):

1. N1 sends an "egress label" L0 for its loopback N1 to R3 and inserts a "pop L0 and forward" entry in its LFIB.
2. R3 sends a label L1 for N1 to R7 over the targeted LDP session and inserts a "swap L1 with L0" in its LFIB.
3. R7 sends label L2 for N1 to N0 and inserts a "swap L2 with L1" entry in its LFIB.
4. N0 inserts a "push L2" entry in its LFIB for traffic destined to N1.

In parallel, nodes in Ring 17 exchange labels for traffic within the ring.

To send a packet to N1, N0 pushes label L2. When this reaches R7, R7 swaps L2 with L1 and additionally pushes a ring label to reach R3. Ring forwarding occurs between R7 and R3. R3 pops the ring label, swaps L2 with L1 and forwards the packet to N1. If a failure occurs on the ring, ring protection kicks in. A failure of R7, R3 or any non-ring node will be dealt with by the non-ring label distribution protocol (in this case, LDP).

6.2. 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.

6.3. 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.

7. Security Considerations

This document proposes extensions to IS-IS, OSPF, LDP and RSVP-TE, all of which have mechanisms to secure them. The extensions proposed do not represent per se a compromise to network security when the control plane is secured, since any manipulation of the content of the messages or even the control plane misinterpretation of the semantics are avoided.

A compromised or otherwise misbehaving node can foil the autodiscovery process Section 4, leading to a ring never transitioning to a usable state.

8. 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.

9. IANA Considerations

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

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

10.2. Informative References

- [IEEE.802.1D_2004] IEEE, "IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges", IEEE 802.1D-2004, DOI 10.1109/ieeestd.2004.94569, July 2004, <<http://ieeexplore.ieee.org/servlet/opac?punumber=9155>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<https://www.rfc-editor.org/info/rfc3630>>.
- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <<https://www.rfc-editor.org/info/rfc5036>>.
- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <<https://www.rfc-editor.org/info/rfc5305>>.

- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Authors' Addresses

Kireeti Kompella
Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, CA 94089
USA

Email: kireeti.ietf@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://lmcontreras.com>

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 28, 2022

T. Saad
Juniper Networks
R. Gandhi
Cisco Systems, Inc.
X. Liu
Volta Networks
V. Beeram
Juniper Networks
I. Bryskin
Huawei Technologies
July 27, 2021

A YANG Data Model for MPLS Static LSPs
draft-ietf-mpls-static-yang-13

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 Label Edge Router(s) LER(s) and Label Switched Router(s) 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 January 28, 2022.

Copyright Notice

Copyright (c) 2021 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
1.1. Terminology	3
1.2. Acronyms and Abbreviations	3
2. MPLS Static LSP Model	3
2.1. Model Organization	4
2.2. Model Tree Diagram	4
2.3. Model Overview	6
2.4. Model YANG Module(s)	7
3. IANA Considerations	14
4. Security Considerations	15
5. Contributors	15
6. References	16
6.1. Normative References	16
6.2. Informative References	17
Authors' Addresses	18

1. Introduction

This document describes a YANG [RFC7950] data model for configuring and managing the Multiprotocol Label Switching (MPLS) [RFC3031] Static LSPs. 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 LER and 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 broken into two modules "ietf-mpls-static" and "ietf-mpls-static-extended". The "ietf-mpls-static" module covers basic features for the configuration and management of unidirectional Static LSP(s), while "ietf-mpls-static-extended" covers extended features like the configuration and management of bidirectional Static LSP(s) and LSP admission control.

The module "ietf-mpls-static" augments the MPLS Base YANG model defined in module "ietf-mpls" in [I-D.ietf-mpls-base-yang].

1.1. Terminology

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.

The terminology for describing YANG data models is found in [RFC7950].

1.2. Acronyms and Abbreviations

MPLS: Multiprotocol Label Switching

LSP: Label Switched Path

LSR: Label Switching Router

LER: Label Edge Router

FEC: Forwarding Equivalence Class

NHLFE: Next Hop Label Forwarding Entry

ILM: Incoming Label Map

2. MPLS Static LSP Model

2.1. 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.

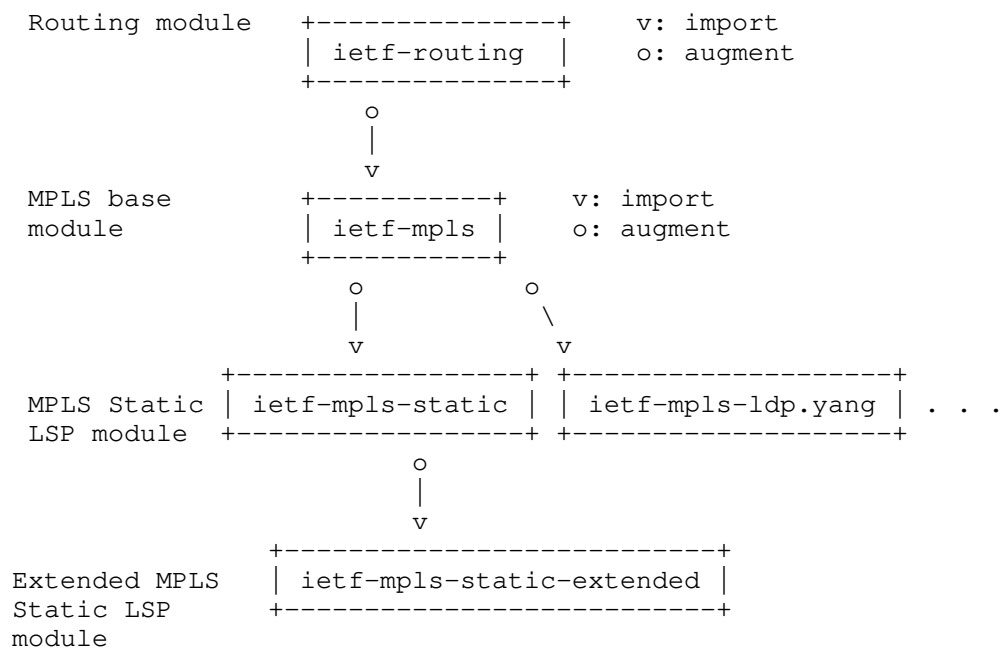


Figure 1: Relationship between MPLS modules

2.2. Model Tree Diagram

The MPLS Static and extended LSP tree diagram as per [RFC8340] is shown in Figure 2.

```

module: ietf-mpls-static
  augment /rt:routing/mpls:mpls:
    +--rw static-lsps
      +--rw static-lsp* [name]
        |   +--rw name          string
        |   +--rw operation?    mpls:mpls-operations-type

```

```

+--rw in-segment
|   +--rw fec
|   |   +--rw (type)?
|   |   |   +--:(ip-prefix)
|   |   |   |   +--rw ip-prefix?          inet:ip-prefix
|   |   |   +--:(mpls-label)
|   |   |   |   +--rw incoming-label?      rt-types:mpls-label
|   |   +--rw incoming-interface?        if:interface-ref
|   +--rw out-segment
|   |   +--rw (out-segment)?
|   |   +--:(nhlfe-single)
|   |   |   +--rw nhlfe-single
|   |   |   |   +--rw mpls-label-stack
|   |   |   |   |   +--rw entry* [id]
|   |   |   |   |   |   +--rw id              uint8
|   |   |   |   |   |   +--rw label?          rt-types:mpls-label
|   |   |   |   |   |   +--rw ttl?            uint8
|   |   |   |   |   |   +--rw traffic-class?   uint8
|   |   |   |   +--rw outgoing-interface?    if:interface-ref
|   |   +--:(nhlfe-multiple)
|   |   |   +--rw nhlfe-multiple
|   |   |   |   +--rw nhlfe* [index]
|   |   |   |   |   +--rw index                string
|   |   |   |   |   +--rw backup-index?        string
|   |   |   |   |   +--rw loadshare?           uint16
|   |   |   |   |   +--rw role?               nhlfe-role
|   |   |   |   |   +--rw mpls-label-stack
|   |   |   |   |   |   +--rw entry* [id]
|   |   |   |   |   |   |   +--rw id              uint8
|   |   |   |   |   |   |   +--rw label?
|   |   |   |   |   |   |   |   rt-types:mpls-label
|   |   |   |   |   |   |   +--rw ttl?            uint8
|   |   |   |   |   |   |   +--rw traffic-class?   uint8
|   |   |   |   +--rw outgoing-interface?    if:interface-ref
+--rw mpls-static-ext:bandwidth?      uint32
+--rw mpls-static-ext:lsp-priority-setup?  uint8
+--rw mpls-static-ext:lsp-priority-hold?   uint8

module: ietf-mpls-static-extended
augment /rt:routing/mpls:mpls:
+--rw bidir-static-lsps
|   +--rw bidir-static-lsp* [name]
|   |   +--rw name                string
|   |   +--rw forward-lsp?        mpls-static:static-lsp-ref
|   |   +--rw reverse-lsp?        mpls-static:static-lsp-ref

```

Figure 2: MPLS Static LSP tree diagram

2.3. Model Overview

This document defines two YANG modules for MPLS Static LSP(s) configuration and management: `ietf-mpls-static.yang` and `ietf-mpls-static-extended.yang`.

The `ietf-mpls-static` module contains the following high-level types and groupings:

`static-lsp-ref`:

A YANG reference type for a static LSP that can be used by data models to reference a configured static LSP.

`in-segment`:

A YANG grouping that describes parameters of an incoming class of FEC associated with a specific LSP as described in the MPLS architecture document [RFC3031]. The model allows the following types of traffic to be mapped onto the static LSP on an ingress LER:

- o Unlabeled traffic destined to a specific prefix
- o Labeled traffic arriving with a specific label

`out-segment`:

A YANG grouping that describes parameters for the forwarding path(s) and their associated attributes for an LSP. The model allows for the following cases:

- o single forwarding path or NHLFE
- o multiple forwarding path(s) or NHLFE(s), each of which can serve a primary, backup or both role(s).

The `ietf-mpls-static-extended` module contains the following high-level types and groupings:

`bidir-static-lsp`:

A YANG grouping that describes list of static bidirectional LSPs

The `ietf-mpls-static-extended` augments the `ietf-mpls-static` model with additional parameters to configure and manage:

- o Bidirectional Static LSP(s)
- o Defining Static LSP bandwidth allocation

- o Defining Static LSP preemption priorities

2.4. Model YANG Module(s)

Configuring LSPs through an LSR/LER involves the following steps:

- o Enabling MPLS on MPLS capable interfaces.
- o Configuring in-segments and out-segments on LER(s) and LSR(s) traversed by the LSP.
- o Setting up the cross-connect per LSP to associate segments and/or to indicate connection origination and termination.
- o Optionally specifying label stack actions.
- o Optionally specifying segment traffic parameters.

The objects covered by this model are derived from the Incoming Label Map (ILM) and Next Hop Label Forwarding Entry (NHLFE) as specified in the MPLS architecture document [RFC3031].

The ietf-mpls-static module imports the following modules:

- o ietf-inet-types defined in [RFC6991]
- o ietf-routing defined in [RFC8349]
- o ietf-routing-types defined in [RFC8294]
- o ietf-interfaces defined in [RFC8343]
- o ietf-mpls defined in [I-D.ietf-mpls-base-yang]
- o ietf-te defined in [I-D.ietf-teas-yang-te]

The ietf-mpls-static module is shown below:

```
<CODE BEGINS> file "ietf-mpls-static@2019-09-12.yang"
module ietf-mpls-static {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-static";
  prefix "mpls-static";

  import ietf-mpls {
    prefix "mpls";
    reference "draft-ietf-mpls-base-yang: MPLS Base YANG Data Model";
  }
}
```

```
import ietf-routing {
  prefix "rt";
  reference "RFC8349: A YANG Data Model for Routing Management";
}

import ietf-routing-types {
  prefix "rt-types";
  reference "RFC8294: Common YANG Data Types for the Routing Area";
}

import ietf-inet-types {
  prefix inet;
  reference "RFC6991: Common YANG Data Types";
}

import ietf-interfaces {
  prefix "if";
  reference "RFC7223: A YANG Data Model for Interface Management";
}

organization "IETF MPLS Working Group";

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

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

  Editor:    Tarek Saad
             <mailto:tsaad@juniper.net>

  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:    Igor Bryskin
             <mailto:Igor.Bryskin@huawei.com>";

description
  "This YANG module augments the 'ietf-routing' module with basic
  configuration and operational state data for MPLS static
  The model fully conforms to the Network Management Datastore
  Architecture (NMDA)."
```

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

(<https://trustee.ietf.org/license-info>).

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

```
// RFC Ed.: replace XXXX with actual RFC number and remove this  
// note.
```

```
// RFC Ed.: update the date below with the date of RFC publication  
// and remove this note.
```

```
revision "2019-09-12" {  
  description  
    "Latest revision of MPLS Static LSP YANG module";  
  reference "RFC XXXX: A YANG Data Model for MPLS Static LSPs";  
}
```

```
typedef static-lsp-ref {  
  type leafref {  
    path "/rt:routing/mpls:mpls/mpls-static:static-lsps/" +  
      "mpls-static:static-lsp/mpls-static:name";  
  }  
  description  
    "This type is used by data models that need to reference  
    configured static LSP.";  
}
```

```
grouping in-segment {  
  description "In-segment grouping";  
  container in-segment {  
    description "MPLS incoming segment";  
    container fec {  
      description "Forwarding Equivalence Class grouping";  
      choice type {  
        description "FEC type choices";  
        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";
      }
    }
  }
  leaf incoming-interface {
    type if:interface-ref;
    description
      "Optional incoming interface if FEC is restricted
       to traffic incoming on a specific interface";
  }
}

grouping out-segment {
  description "Out-segment grouping";
  container out-segment {
    description "MPLS outgoing segment";
    choice out-segment {
      description "The MPLS out-segment type choice";
      case nhlfe-single {
        container nhlfe-single {
          description "Container for single NHLFE entry";
          uses mpls:nhlfe-single-contents;
          leaf outgoing-interface {
            type if:interface-ref;
            description
              "The outgoing interface";
          }
        }
      }
      case nhlfe-multiple {
        container nhlfe-multiple {
          description "Container for multiple NHLFE entries";
          list nhlfe {
            key index;
            description "MPLS NHLFE entry";
            uses mpls:nhlfe-multiple-contents;
            leaf outgoing-interface {
              type if:interface-ref;
              description
                "The outgoing interface";
            }
          }
        }
      }
    }
  }
}
```

```

    }
  }
}

augment "/rt:routing/mpls:mpls" {
  description "Augmentations for MPLS Static LSPs";
  container static-lsps {
    description
      "Statically configured LSPs, without dynamic signaling";
    list static-lsp {
      key name;
      description "list of defined static LSPs";
      leaf name {
        type string;
        description "name to identify the LSP";
      }
      leaf operation {
        type mpls:mpls-operations-type;
        description
          "The MPLS operation to be executed on the incoming packet";
      }
      uses in-segment;
      uses out-segment;
    }
  }
}
}
<CODE ENDS>

```

The ietf-mpls-static-extended module imports the following modules:

- o ietf-mpls defined in [I-D.ietf-mpls-base-yang]
- o ietf-mpls-static defined in this document
- o ietf-routing defined in [RFC8349]

The ietf-mpls-static-extended module is shown below:

```

<CODE BEGINS> file "ietf-mpls-static-extended@2019-09-12.yang"
module ietf-mpls-static-extended {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-static-extended";
  prefix "mpls-static-ext";

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

```

```
    reference "draft-ietf-mpls-base-yang: MPLS Base YANG Data Model";
  }

  import ietf-routing {
    prefix "rt";
    reference "RFC8349: A YANG Data Model for Routing Management";
  }

  import ietf-routing-types {
    prefix "rt-types";
    reference "RFC8294: Common YANG Data Types for the Routing Area";
  }

  import ietf-mpls-static {
    prefix "mpls-static";
    reference "RFC XXXX: A YANG Data Model for MPLS Static LSPs";
  }

  organization "IETF MPLS Working Group";

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

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

    Editor:    Tarek Saad
               <mailto:tσαad@juniper.net>

    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:    Igor Bryskin
               <mailto: Igor.Bryskin@huawei.com>";

  description
    "This YANG module contains the Extended MPLS Static LSP YANG
    data model. The model fully conforms to the Network Management
    Datastore Architecture (NMDA).

    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
(https://trustee.ietf.org/license-info).
This version of this YANG module is part of RFC XXXX; see
the RFC itself for full legal notices.";

// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.

// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.

revision "2019-09-12" {
  description
    "Latest revision of MPLS Static LSP Extended YANG module";
  reference "RFC XXXX: A YANG Data Model for MPLS Static LSPs";
}

grouping bidir-static-lsp {
  description
    "grouping for top level list of static bidirectional 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";
  }
}

augment "/rt:routing/mppls:mppls/mppls-static:static-lsps" {
  description
    "Augmentation for static MPLS LSPs";

  leaf bandwidth {
    type rt-types:bandwidth-ieee-float32;
    units "Bytes per second";
    description
      "Bandwidth 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";
}
}

augment "/rt:routing/mpls:mpls" {
    description "Augmentations for MPLS Static LSPs";
    container bidir-static-lsps {
        description
            "Statically configured bidirectional LSPs";
        list bidir-static-lsp {
            key name;
            description "List of static bidirectional LSPs";

            leaf name {
                type string;
                description "Name that identifies the bidirectional LSP";
            }
            uses bidir-static-lsp;
        }
    }
}
}
<CODE ENDS>
```

3. 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.

URI: urn:ietf:params:xml:ns:yang:ietf-mpls-static
Registrant Contact: The MPLS WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-mpls-static-extended
Registrant Contact: The MPLS WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

This document registers two YANG modules in the YANG Module Names registry [RFC6020].

```
name:      ietf-mpls-static
namespace: urn:ietf:params:xml:ns:yang:ietf-mpls-static
prefix:    ietf-mpls-static
// RFC Ed.: replace XXXX with RFC number and remove this note
reference: RFCXXXX

name:      ietf-mpls-static-extended
namespace: urn:ietf:params:xml:ns:yang:ietf-mpls-static-extended
prefix:    ietf-mpls-static-extended
// RFC Ed.: replace XXXX with RFC number and remove this note
reference: RFCXXXX
```

4. Security Considerations

The YANG modules specified in this document define schemas for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default) 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. These are the subtrees and data nodes and their sensitivity/vulnerability:

o /ietf-routing:routing/ietf-mpls:mpls:/ietf-mpls:static-lsps: This entire subtree is related to security.

An administrator needs to restrict write access to all configurable objects within this data model.

5. Contributors

Himanshu Shah
Ciena
email: hshah@ciena.com

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

6. References

6.1. Normative References

- [I-D.ietf-mpls-base-yang]
Saad, T., Raza, K., Gandhi, R., Liu, X., and V. P. Beeram,
"A YANG Data Model for MPLS Base", draft-ietf-mpls-base-
yang-17 (work in progress), October 2020.
- [I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V. P., Bryskin, I.,
and O. G. D. Dios, "A YANG Data Model for Traffic
Engineering Tunnels, Label Switched Paths and Interfaces",
draft-ietf-teas-yang-te-27 (work in progress), July 2021.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol
Label Switching Architecture", RFC 3031,
DOI 10.17487/RFC3031, January 2001,
<<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
DOI 10.17487/RFC3688, January 2004,
<<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
the Network Configuration Protocol (NETCONF)", RFC 6020,
DOI 10.17487/RFC6020, October 2010,
<<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
and A. Bierman, Ed., "Network Configuration Protocol
(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
<<https://www.rfc-editor.org/info/rfc6241>>.

- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <<https://www.rfc-editor.org/info/rfc8294>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.

6.2. Informative References

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

Authors' Addresses

Tarek Saad
Juniper Networks

Email: tsaad.net@gmail.com

Rakesh Gandhi
Cisco Systems, Inc.

Email: rgandhi@cisco.com

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Igor Bryskin
Huawei Technologies

Email: Igor.Bryskin@huawei.com

BFD Working Group
Internet-Draft
Intended status: Informational
Expires: 8 September 2022

G. Mirsky
Ericsson
7 March 2022

BFD in Demand Mode over Point-to-Point MPLS LSP
draft-mirsky-bfd-mpls-demand-11

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 8 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Conventions used in this document	2
2.1. Terminology	2
3. Use of the BFD Demand Mode	2
3.1. The Applicability of BFD for Multipoint Networks	4
4. IANA Considerations	4
5. Security Considerations	4
6. Normative References	4
7. Informative References	5
Appendix A. Acknowledgements	5
Author's Address	6

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

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. If the BFD session is configured to use the Demand mode, once the BFD session is in Up state the ingress LER switches to the Demand mode as defined in Section 6.6 [RFC5880]. The egress LER also follows procedures defined in Section 6.6 [RFC5880] and ceases further transmission of periodic BFD control packets to the ingress LER.

In this state BFD peers remains as long as the egress LER is in Up state. The ingress LER can periodically check continuity of a bidirectional path between the ingress and egress LERs by using the Poll Sequence, as described in Section 6.6 [RFC5880]. An implementation that supports using the Poll Sequence as the mechanism for bidirectional path continuity check must control the interval between consecutive Poll Sequences. The Rdefault value could be selected as 1 second.

If the Detection timer at the egress LER expires, the BFD system on LER sends BFD Control packet to the ingress LER with the Poll (P) bit set, Status (Sta) field set to the Down value, and the Diagnostic (Diag) field set to Control Detection Time Expired value. The egress LER periodically transmits these Control packets to the ingress LER 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. An implementation that supports this specification provides control of the interval between consecutive Poll messages signaling the expiration of the Detection timer. The default value of the interval can be selected as 1 second.

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 is 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 is set;
- * Demand (D) flag in BFD control packet is 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.

3.1. The Applicability of BFD for Multipoint Networks

[RFC8562] defines the use of BFD in multipoint networks. This specification analyzes the case of p2p LSP. In that scenario, the ingress of the LSP acts as the MultipointHead, and the egress - as MultipointTail. The BFD state machines for MultipointHead, MultipointClient, and MultipointTail don't use the three-way handshakes for session establishment and teardown. As a result, the Init state is absent, and the session transitions to the Up state once the BFD session is administratively enabled. Hence, a BFD session over a p2p LSP, using principles of [RFC8562] or [RFC8563], can be established faster if the MultipointTail has been provisioned with the value of My Discriminator used by the MultipointHead for that BFD session. That value can be provided to the MultipointTail using different mechanisms, e.g., an extension to IGP. Description of mechanism to provide the value of My Discriminator used by the MultipointHead for the particular BFD session is outside the scope of this specification.

Unsolicited notification of the detected failure by the MultipointTail to the MultipointClient performs as described above for the case when the ingress BFD system switches the remote peer into 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

- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883, June 2010, <<https://www.rfc-editor.org/info/rfc5883>>.
- [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010, <<https://www.rfc-editor.org/info/rfc5884>>.
- [RFC6425] Saxena, S., Ed., Swallow, G., Ali, Z., Farrel, A., Yasukawa, S., and T. Nadeau, "Detecting Data-Plane Failures in Point-to-Multipoint MPLS - Extensions to LSP Ping", RFC 6425, DOI 10.17487/RFC6425, November 2011, <<https://www.rfc-editor.org/info/rfc6425>>.
- [RFC7726] Govindan, V., Rajaraman, K., Mirsky, G., Akiya, N., and S. Aldrin, "Clarifying Procedures for Establishing BFD Sessions for MPLS Label Switched Paths (LSPs)", RFC 7726, DOI 10.17487/RFC7726, January 2016, <<https://www.rfc-editor.org/info/rfc7726>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8562] Katz, D., Ward, D., Pallagatti, S., Ed., and G. Mirsky, Ed., "Bidirectional Forwarding Detection (BFD) for Multipoint Networks", RFC 8562, DOI 10.17487/RFC8562, April 2019, <<https://www.rfc-editor.org/info/rfc8562>>.

7. Informative References

- [RFC8563] Katz, D., Ward, D., Pallagatti, S., Ed., and G. Mirsky, Ed., "Bidirectional Forwarding Detection (BFD) Multipoint Active Tails", RFC 8563, DOI 10.17487/RFC8563, April 2019, <<https://www.rfc-editor.org/info/rfc8563>>.

Appendix A. Acknowledgements

TBD

Author's Address

Greg Mirsky
Ericsson
Email: gregimirsky@gmail.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 March 2022

G. Mirsky
Ericsson
G. Mishra
Verizon Inc.
D. Eastlake
Futurewei Technologies
19 September 2021

BFD for Multipoint Networks over Point-to-Multi-Point MPLS LSP
draft-mirsky-mpls-p2mp-bfd-15

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) and Segment Routing (SR) point-to-multipoint policies with SR-MPLS data plane.

It also describes the applicability of LSP Ping, as in-band, and the control plane, as out-band, solutions to bootstrap a BFD session.

It also describes the behavior of the active tail for head notification.

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 23 March 2022.

Copyright Notice

Copyright (c) 2021 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	3
2.1. Terminology	3
2.2. Requirements Language	3
3. Multipoint BFD Encapsulation	3
3.1. IP Encapsulation of Multipoint BFD	4
3.2. Non-IP Encapsulation of Multipoint BFD	4
4. Bootstrapping Multipoint BFD	5
4.1. LSP Ping	5
4.2. Control Plane	6
5. Operation of Multipoint BFD with Active Tail over P2MP MPLS LSP	6
6. Security Considerations	7
7. IANA Considerations	8
8. Acknowledgements	8
9. References	8
9.1. Normative References	8
9.2. Informative References	10
Authors' Addresses	10

1. Introduction

[RFC8562] 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.

[RFC8562] added two BFD session types - MultipointHead and MultipointTail. Throughout this document, MultipointHead and MultipointTail refer to the value of the `bfd.SessionType` is set on a BFD endpoint.

This document describes procedures for using such modes of BFD protocol to detect data plane failures in Multiprotocol Label Switching (MPLS) point-to-multipoint (p2mp) Label Switched Paths (LSPs) and Segment Routing (SR) point-to-multipoint policies with SR-MPLS data plane

The document also describes the applicability of out-band solutions to bootstrap a BFD session in this environment.

It also describes the behavior of the active tail for head notification.

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

LSR: Label Switching Router

SR: Segment Routing

SR-MPLS: SR with MPLS data plane

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

[RFC8562] uses BFD in the Demand mode from the very start of a point-to-multipoint (p2mp) BFD session. Because the head doesn't receive any BFD Control packet from a tail, the head of the p2mp BFD session transmits all BFD Control packets with the value of Your Discriminator field set to zero. As a result, a tail cannot demultiplex BFD sessions using Your Discriminator, as defined in

[RFC5880]. [RFC8562] requires that to demultiplex BFD sessions, the tail uses the source IP address, My Discriminator, and the identity of the multipoint tree from which the BFD Control packet was received. If the BFD Control packet is encapsulated in IP/UDP, then the source IP address MUST be used to demultiplex the received BFD Control packet as described in Section 3.1. The non-IP encapsulation case is described in Section 3.2.

3.1. IP Encapsulation of Multipoint BFD

[RFC8562] defines IP/UDP encapsulation for multipoint BFD over p2mp MPLS LSP:

- * UDP destination port MUST be set to 3784;
- * destination IP address MUST be set to the loopback address 127.0.0.1/32 for IPv4, or the loopback address ::1/128 for IPv6 [RFC4291]. Note that that is different from how the destination IP address selection is defined in Section 7 [RFC5884]. Firstly, because only one loopback address ::1/128 is defined in IPv6. And also, it is recommended to use the Entropy Label [RFC6790] to discover multiple alternate paths in an MPLS network. Using a single loopback address both for IPv4 and IPv6 encapsulation makes it consistent and more straightforward for an implementation.

The Motivation section [RFC6790] lists several advantages of generating the entropy value by an ingress Label Switching Router (LSR) compared to when a transit LSR infers entropy using the information in the MPLS label stack or payload. Thus 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 those particular alternative paths;

or the MultipointHead MAY use the UDP port number as discovered by LSP Ping traceroute [RFC8029] as the source UDP port number to possibly exercise those particular alternate paths.

3.2. Non-IP Encapsulation of Multipoint BFD

In some environments, the overhead of extra IP/UDP encapsulations may be considered burdensome, making the use of more compact G-ACh encapsulation attractive. Also, the validation of the IP/UDP encapsulation of a BFD Control packet in a p2mp BFD session may fail because of a problem related to neither the MPLS label stack nor to BFD. Avoiding unnecessary encapsulation of p2mp BFD over an MPLS LSP

improves the accuracy of the correlation of the detected failure and defect in MPLS LSP. Non-IP encapsulation for multipoint BFD over p2mp MPLS LSP MUST use Generic Associated Channel (G-ACH) Label (GAL) (see [RFC5586]) at the bottom of the label stack followed by an Associated Channel Header (ACH). If a BFD Control packet in PW-ACH encapsulation (without IP/UDP Headers) is to be used in ACH, an implementation would not be able to verify the identity of the MultipointHead and, as a result, will not properly demultiplex BFD packets. Hence, a new channel type value is needed. The Channel Type field in ACH MUST be set to TBA1 value Section 7. To provide the identity of the MultipointHead for the particular multipoint BFD session, a Source Address TLV [RFC7212] MUST immediately follow a BFD Control message.

4. Bootstrapping Multipoint BFD

4.1. LSP Ping

LSP Ping is the part of the on-demand OAM toolset used to detect and localize defects in the data plane and verify the control plane against the data plane by ensuring that the LSP is mapped to the same FEC at both egress and ingress endpoints.

LSP Ping, as defined in [RFC6425], MAY be used to bootstrap MultipointTail. If LSP Ping is used, it MUST include the Target FEC TLV and the BFD Discriminator TLV defined in [RFC5884]. For the case of p2mp MPLS LSP, the Target FEC TLV MUST use sub-TLVs defined in Section 3.1 [RFC6425]. For the case of p2mp SR policy with SR-MPLS data plane, an implementation of this specification MUST follow procedures defined in [RFC8287]. Setting the value of Reply Mode field to "Do not reply" [RFC8029] for the LSP Ping to bootstrap MultipointTail of the p2mp BFD session is RECOMMENDED. Indeed, because BFD over a multipoint network uses BFD Demand mode, the LSP echo reply from a tail has no useful information to convey to the head, unlike in the case of the BFD over a p2p MPLS LSP [RFC5884]. A MultipointTail that receives an LSP Ping that includes the BFD Discriminator TLV:

- * MUST validate the LSP Ping;
- * MUST associate the received BFD Discriminator value with the p2mp LSP;
- * MUST create a p2mp BFD session and set `bfd.SessionType = MultipointTail` as described in [RFC8562];

- * MUST use the source IP address of LSP Ping, the value of BFD Discriminator from the BFD Discriminator TLV, and the identity of the p2mp LSP to properly demultiplex BFD sessions.

Besides bootstrapping a BFD session over a p2mp LSP, LSP Ping SHOULD be used to verify the control plane against the data plane periodically by checking that the p2mp LSP is mapped to the same FEC at the MultipointHead and all active MultipointTails. 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 because LSP Ping requires more processing to validate the consistency between the data plane and the control plane. An implementation MAY provide configuration options to control the rate of generation of the periodic LSP Ping Echo request messages.

4.2. Control Plane

The BGP-BFD Attribute [RFC9026] MAY be used to bootstrap multipoint BFD session on a tail.

5. Operation of Multipoint BFD with Active Tail over P2MP MPLS LSP

[RFC8562] defined how the BFD Demand mode can be used in multipoint networks. When applied in MPLS, procedures specified in [RFC8562] allow an egress LSR to detect a failure of the part of the MPLS p2mp LSP from the ingress LSR. The ingress LSR is not aware of the state of the p2mp LSP. [RFC8563], using mechanisms defined in [RFC8562], defined an "active tail" behavior. An active tail might notify the head of the detected failure and responds to a poll sequence initiated by the head. The first method, referred to as Head Notification without Polling, is mentioned in Section 5.2.1 [RFC8563], is the simplest of all described in [RFC8563]. The use of this method in BFD over MPLS p2mp LSP is discussed in this document. Analysis of other methods of a head learning of the state of an MPLS p2mp LSP is outside the scope of this document.

As specified in [RFC8563] for the active tail mode, BFD variables MUST be as follows:

On an ingress LSR:

- * bfd.SessionType is MultipointHead;
- * bfd.RequiredMinRxInterval is set to nonzero, allowing egress LSRs to send BFD Control packets.

On an egress LSR:

- * bfd.SessionType is MultipointTail;
- * bfd.SilentTail is set to zero.

In Section 5.2.1 [RFC8563] is noted that "the tail sends unsolicited BFD packets in response to the detection of a multipoint path failure" but without the specifics on the information in the packet and frequency of transmissions. This document defines below the procedure of an active tail with unsolicited notifications for p2mp MPLS LSP.

Upon detecting the failure of the p2mp MPLS LSP, an egress LSR sends BFD Control packet with the following settings:

- * the Poll (P) bit is set;
- * the Status (Sta) field set to Down value;
- * the Diagnostic (Diag) field set to Control Detection Time Expired value;
- * the value of the Your Discriminator field is set to the value the egress LSR has been using to demultiplex that BFD multipoint session;
- * BFD Control packet MAY be encapsulated in IP/UDP with the destination IP address of the ingress LSR and the UDP destination port number set to 4784 per [RFC5883]. If non-IP encapsulation is used, then a BFD Control packet is encapsulated using PW-ACH encapsulation (without IP/UDP Headers) (0x0007) [RFC5885];
- * these BFD Control packets are transmitted at the rate of one per second until either it receives a control packet valid for this BFD session with the Final (F) bit set from the ingress LSR or the defect condition clears; however to improve the likelihood of notifying the ingress LSR of the failure of the p2mp MPLS LSP, the egress LSR SHOULD initially transmit three BFD Control packets defined above in short succession.

An ingress LSR that has received the BFD Control packet, as described above, sends the unicast IP/UDP encapsulated BFD Control packet with the Final (F) bit set to the egress LSR.

6. Security Considerations

This document does not introduce new security aspects but inherits all security considerations from [RFC5880], [RFC5884], [RFC7726], [RFC8562], [RFC8029], and [RFC6425].

Also, BFD for p2mp MPLS LSP MUST follow the requirements listed in section 4.1 [RFC4687] to avoid congestion in the control plane or the data plane caused by the rate of generating BFD Control packets. An operator SHOULD consider the amount of extra traffic generated by p2mp BFD when selecting the interval at which the MultipointHead will transmit BFD Control packets. The operator MAY consider the size of the packet the MultipointHead transmits periodically as using IP/UDP encapsulation, which adds up to 28 octets, more than 50% of the BFD Control packet length, comparing to G-ACh encapsulation.

7. IANA Considerations

IANA is requested to allocate value (TBA1) from its MPLS Generalized Associated Channel (G-ACh) Types registry.

Value	Description	Reference
TBA1	Multipoint BFD Session	This document

Table 1: Multipoint BFD Session G-ACh Type

8. Acknowledgements

The authors sincerely appreciate the comments received from Andrew Malis, Italo Busi, Shraddha Hegde, and thought stimulating questions from Carlos Pignataro.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <<https://www.rfc-editor.org/info/rfc5586>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.

- [RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883, June 2010, <<https://www.rfc-editor.org/info/rfc5883>>.
- [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010, <<https://www.rfc-editor.org/info/rfc5884>>.
- [RFC5885] Nadeau, T., Ed. and C. Pignataro, Ed., "Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)", RFC 5885, DOI 10.17487/RFC5885, June 2010, <<https://www.rfc-editor.org/info/rfc5885>>.
- [RFC6425] Saxena, S., Ed., Swallow, G., Ali, Z., Farrel, A., Yasukawa, S., and T. Nadeau, "Detecting Data-Plane Failures in Point-to-Multipoint MPLS - Extensions to LSP Ping", RFC 6425, DOI 10.17487/RFC6425, November 2011, <<https://www.rfc-editor.org/info/rfc6425>>.
- [RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <<https://www.rfc-editor.org/info/rfc6790>>.
- [RFC7212] Frost, D., Bryant, S., and M. Bocci, "MPLS Generic Associated Channel (G-ACh) Advertisement Protocol", RFC 7212, DOI 10.17487/RFC7212, June 2014, <<https://www.rfc-editor.org/info/rfc7212>>.
- [RFC7726] Govindan, V., Rajaraman, K., Mirsky, G., Akiya, N., and S. Aldrin, "Clarifying Procedures for Establishing BFD Sessions for MPLS Label Switched Paths (LSPs)", RFC 7726, DOI 10.17487/RFC7726, January 2016, <<https://www.rfc-editor.org/info/rfc7726>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8287] Kumar, N., Ed., Pignataro, C., Ed., Swallow, G., Akiya, N., Kini, S., and M. Chen, "Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes", RFC 8287, DOI 10.17487/RFC8287, December 2017, <<https://www.rfc-editor.org/info/rfc8287>>.
- [RFC8562] Katz, D., Ward, D., Pallagatti, S., Ed., and G. Mirsky, Ed., "Bidirectional Forwarding Detection (BFD) for Multipoint Networks", RFC 8562, DOI 10.17487/RFC8562, April 2019, <<https://www.rfc-editor.org/info/rfc8562>>.
- [RFC8563] Katz, D., Ward, D., Pallagatti, S., Ed., and G. Mirsky, Ed., "Bidirectional Forwarding Detection (BFD) Multipoint Active Tails", RFC 8563, DOI 10.17487/RFC8563, April 2019, <<https://www.rfc-editor.org/info/rfc8563>>.

9.2. Informative References

- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4687] Yasukawa, S., Farrel, A., King, D., and T. Nadeau, "Operations and Management (OAM) Requirements for Point-to-Multipoint MPLS Networks", RFC 4687, DOI 10.17487/RFC4687, September 2006, <<https://www.rfc-editor.org/info/rfc4687>>.
- [RFC9026] Morin, T., Ed., Kebler, R., Ed., and G. Mirsky, Ed., "Multicast VPN Fast Upstream Failover", RFC 9026, DOI 10.17487/RFC9026, April 2021, <<https://www.rfc-editor.org/info/rfc9026>>.

Authors' Addresses

Greg Mirsky
Ericsson

Email: gregimirsky@gmail.com

Gyan Mishra
Verizon Inc.

Email: gyan.s.mishra@verizon.com

Donald Eastlake, 3rd
Futurewei Technologies
2386 Panoramic Circle
Apopka, FL 32703
United States of America

Email: d3e3e3@gmail.com

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 28, 2020

G. Mirsky
ZTE Corp.
J. Tantsura
Apstra, Inc.
I. Varlashkin
Google
M. Chen
Huawei
J. Wenying
CMCC
April 26, 2020

Bidirectional Forwarding Detection (BFD) in Segment Routing Networks
Using MPLS Dataplane
draft-mirsky-spring-bfd-10

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 an SR Policy in the reverse direction of the SR-MPLS tunnel, and applicability of BFD Demand mode in the SR-MPLS domain. Also, the document describes the use of BFD Echo with BFD Control packet payload.

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 October 28, 2020.

Copyright Notice

Copyright (c) 2020 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
1.1. Conventions	3
1.1.1. Terminology	3
1.1.2. Requirements Language	3
2. Bootstrapping BFD Session over Segment Routed Tunnel with MPLS Data Plane	4
3. Use BFD Reverse Path TLV over Segment Routed MPLS Tunnel	5
4. Use Non-FEC Path TLV	5
5. BFD Reverse Path TLV over Segment Routed MPLS Tunnel with Dynamic Control Plane	7
6. Applicability of BFD Demand Mode in SR-MPLS Domain	7
7. Using BFD to Monitor Point-to-Multipoint SR Policy	7
8. Use of Echo BFD in SR-MPLS	8
9. IANA Considerations	8
9.1. Non-FEC Path TLV	8
9.2. Return Code	9
10. Implementation Status	10
11. Security Considerations	10
12. Contributors	11
13. Acknowledgments	11
14. References	11
14.1. Normative References	11
14.2. Informative References	13
Authors' Addresses	13

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 remote BFD system, which is at the egress Label Edge Router (LER), will use the shortest path route regardless of the path the BFD system at the ingress LER uses to send BFD Control packets towards it. Throughout this document, references to ingress LER and egress LER are used, respectively, as a shortened version of the "BFD system at the ingress/egress LER".

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

BSID: Binding Segment Identifier

FEC: Forwarding Equivalence Class

MPLS: Multiprotocol Label Switching

SR-MPLS Segment Routing with MPLS data plane

LSP: Label Switched Path

LER Label Edge Router

p2p Point-to-point

p2mp Point-to-multipoint

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 LER 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 [RFC8660] 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 an SR-MPLS domain if there are no other means to bootstrap the BFD session, e.g., using an extension to a dynamic routing protocol as described in [I-D.ietf-bess-mvpn-fast-failover] and [I-D.ietf-pim-bfd-p2mp-use-case].

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 the 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 LER, 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 LER 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 the ingress LER is aware of the SR-based explicit route, the egress LER 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 bootstrapping 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 LER to use a 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."

This document defines a new optional Non-FEC Path TLV. The format of the Non-FEC Path TLV is presented in Figure 1

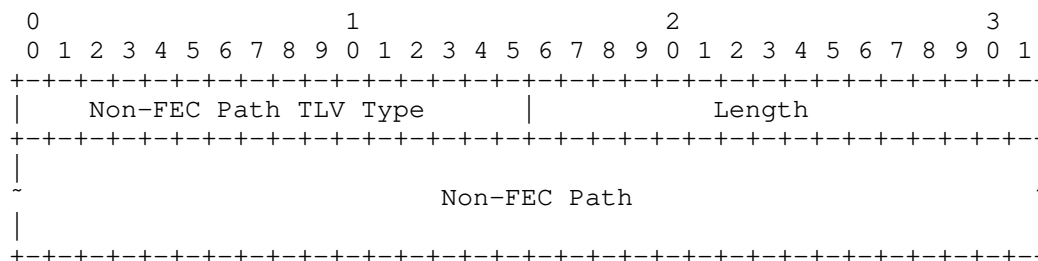


Figure 1: Non-FEC Path TLV Format

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 9.1).

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 LER 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 the Segment 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.

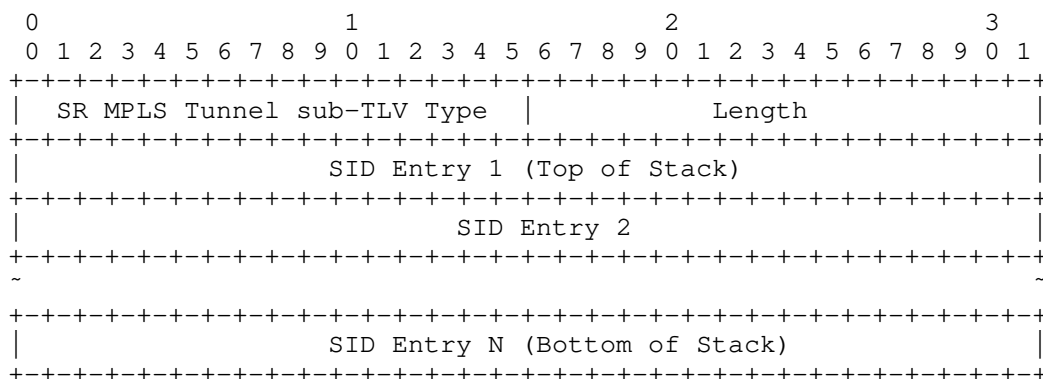


Figure 2: Segment Routing MPLS Tunnel sub-TLV

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 9.1).

The egress LER 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:

- o an ingress SR node bootstraps BFD session over SR-MPLS in Async BFD mode;
- o once BFD session is Up, the ingress SR node switches the egress LER into the Demand mode by setting D field in BFD Control packet it transmits;
- o if the egress LER detects the failure of the BFD session, it sends its BFD Control packet to the ingress SR node over the IP network with a Poll sequence;
- o if the ingress SR 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 SR node 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. Using BFD to Monitor Point-to-Multipoint SR Policy

[I-D.voyer-spring-sr-p2mp-policy] defined variants of SR Policy to deliver point-to-multipoint (p2mp) services. For the given P2MP segment [RFC8562] can be used if, for example, leaves have an alternative source of the multicast service flow to select. In such a scenario, a leaf may switch to using the alternative flow after p2mp BFD detects the failure in the working multicast path. For scenarios where it is required for the root to monitor the state of the multicast tree [RFC8563] can be used. The root may use the detection of the failure of the multicast tree to the particular leaf to restore the path for that leaf or re-instantiate the whole multicast tree.

An essential part of using p2mp BFD is the bootstrapping the BFD session at all the leaves. The root, acting as the MultipointHead, MAY use LSP Ping with the BFD Discriminator TLV. Alternatively, extensions to routing protocols, e.g., BGP, or management plane, e.g., PCEP, MAY be used to associate the particular P2MP segment with MultipointHead's Discriminator. Extensions for routing protocols and management plane are for further study.

8. Use of Echo BFD in SR-MPLS

Echo-BFD [RFC5880] can be used to monitor an SR Policy between the local and the remote BFD peers. As defined in [RFC5880], the remote BFD system does not process the payload of an Echo BFD. Thus it is the local system that demultiplexes the Echo BFD packet matching it to the appropriate BFD session and detects missing Echo BFD packets. A BFD Control packet MAY be used as the payload of Echo BFD. This specification defines the use of Echo BFD in SR-MPLS network with BFD Control packet as the payload. The use of other types of Echo BFD payload is outside the scope of this document. Because the remote BFD system does not process Echo BFD, the value of the Your Discriminator field MUST be set to the discriminator the local BFD system assigned to the given BFD session. My Discriminator field MUST be zeroed. Authentication MUST be set according to the configuration of the BFD session. To ensure that the Echo BFD packet is returned to the sender without being processed, the sender MAY use a Binding SID (BSID) [RFC8402] that has been bound with the SR Policy that ensures the return of a packet to that particular node. A BSID MAY be associated with the SR Policy that is the reverse to the SR Policy programmed onto the BFD Echo packet by the sender.

9. IANA Considerations

9.1. 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.

Range	Registration Procedures	Note
0-16383	Standards Action	This range is for mandatory TLVs or for optional TLVs that require an error message if not recognized. Experimental RFC needed
16384-31743	Specification Required	
32768-49161	Standards Action	This range is for optional TLVs that can be silently dropped if not recognized. Experimental RFC needed
49162-64511	Specification Required	
64512-65535	Private Use	

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.

Value	Description	Reference
0	Reserved	This document
TBD2	Segment Routing MPLS Tunnel sub-TLV	This document
65535	Reserved	This document

Table 3: New Segment Routing Tunnel sub-TLV

9.2. 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.

Value	Description	Reference
X TBD3	Too Many TLVs Detected.	This document

Table 4: New Return Code

10. Implementation Status

- The organization responsible for the implementation: ZTE Corporation.
- The implementation's name ROSng SW empowers traditional routers, e.g., ZXCTN 6000.
- A brief general description: A list of SIDs can be specified as the Return Path for an SR-MPLS tunnel.
- The implementation's level of maturity: production.
- Coverage: complete
- Version compatibility: draft-mirsky-spring-bfd-06.
- Licensing: proprietary.
- Implementation experience: Appreciate Early Allocation of values for Non-FEC TLV and Segment Routing MPLS Tunnel sub-TLV (using Private Use code points).
- Contact information: Qian Xin qian.xin2@zte.com.cn
- The date when information about this particular implementation was last updated: 12/16/2019

Note to RFC Editor: This section MUST be removed before publication of the document.

11. Security Considerations

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

12. Contributors

Xiao Min
ZTE Corp.
Email: xiao.min2@zte.com.cn

13. Acknowledgments

Authors greatly appreciate the help of Qian Xin, who provided the information about the implementation of this specification.

14. References

14.1. Normative References

- [I-D.ietf-mpls-bfd-directed]
Mirsky, G., Tantsura, J., Varlashkin, I., and M. Chen,
"Bidirectional Forwarding Detection (BFD) Directed Return
Path", draft-ietf-mpls-bfd-directed-13 (work in progress),
December 2019.
- [I-D.mirsky-bfd-mpls-demand]
Mirsky, G., "BFD in Demand Mode over Point-to-Point MPLS
LSP", draft-mirsky-bfd-mpls-demand-06 (work in progress),
December 2019.
- [I-D.voyer-spring-sr-p2mp-policy]
daniel.voyer@bell.ca, d., Filsfils, C., Parekh, R.,
Bidgoli, H., and Z. Zhang, "SR Replication Policy for P2MP
Service Delivery", draft-voyer-spring-sr-p2mp-policy-03
(work in progress), July 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
(BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,
<<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
(BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881,
DOI 10.17487/RFC5881, June 2010,
<<https://www.rfc-editor.org/info/rfc5881>>.

- [RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883, June 2010, <<https://www.rfc-editor.org/info/rfc5883>>.
- [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884, June 2010, <<https://www.rfc-editor.org/info/rfc5884>>.
- [RFC6428] Allan, D., Ed., Swallow, G., Ed., and J. Drake, Ed., "Proactive Connectivity Verification, Continuity Check, and Remote Defect Indication for the MPLS Transport Profile", RFC 6428, DOI 10.17487/RFC6428, November 2011, <<https://www.rfc-editor.org/info/rfc6428>>.
- [RFC7726] Govindan, V., Rajaraman, K., Mirsky, G., Akiya, N., and S. Aldrin, "Clarifying Procedures for Establishing BFD Sessions for MPLS Label Switched Paths (LSPs)", RFC 7726, DOI 10.17487/RFC7726, January 2016, <<https://www.rfc-editor.org/info/rfc7726>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8287] Kumar, N., Ed., Pignataro, C., Ed., Swallow, G., Akiya, N., Kini, S., and M. Chen, "Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes", RFC 8287, DOI 10.17487/RFC8287, December 2017, <<https://www.rfc-editor.org/info/rfc8287>>.
- [RFC8402] Filss, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8562] Katz, D., Ward, D., Pallagatti, S., Ed., and G. Mirsky, Ed., "Bidirectional Forwarding Detection (BFD) for Multipoint Networks", RFC 8562, DOI 10.17487/RFC8562, April 2019, <<https://www.rfc-editor.org/info/rfc8562>>.

- [RFC8563] Katz, D., Ward, D., Pallagatti, S., Ed., and G. Mirsky, Ed., "Bidirectional Forwarding Detection (BFD) Multipoint Active Tails", RFC 8563, DOI 10.17487/RFC8563, April 2019, <<https://www.rfc-editor.org/info/rfc8563>>.
- [RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <<https://www.rfc-editor.org/info/rfc8660>>.

14.2. Informative References

- [I-D.ietf-bess-mvpn-fast-failover]
Morin, T., Kebler, R., and G. Mirsky, "Multicast VPN fast upstream failover", draft-ietf-bess-mvpn-fast-failover-10 (work in progress), February 2020.
- [I-D.ietf-pim-bfd-p2mp-use-case]
Mirsky, G. and J. Xiaoli, "Bidirectional Forwarding Detection (BFD) for Multi-point Networks and Protocol Independent Multicast - Sparse Mode (PIM-SM) Use Case", draft-ietf-pim-bfd-p2mp-use-case-03 (work in progress), January 2020.
- [I-D.ietf-spring-mpls-anycast-segments]
Sarkar, P., Gredler, H., Filsfils, C., Previdi, S., Decraene, B., and M. Horneffer, "Anycast Segments in MPLS based Segment Routing", draft-ietf-spring-mpls-anycast-segments-02 (work in progress), January 2018.
- [RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <<https://www.rfc-editor.org/info/rfc6790>>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Jeff Tantsura
Apstra, Inc.

Email: jefftant.ietf@gmail.com

Ilya Varlashkin
Google

Email: Ilya@nobulus.com

Mach (Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Jiang Wenying
CMCC

Email: jiangwenying@chinamobile.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 12, 2018

H. Sitaraman
V. Beeram
Juniper Networks
T. Parikh
Verizon
T. Saad
Cisco Systems
December 9, 2017

Signaling RSVP-TE tunnels on a shared MPLS forwarding plane
draft-sitaraman-mpls-rsvp-shared-labels-03.txt

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.

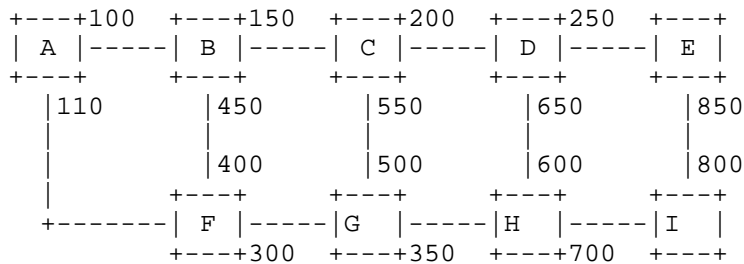


Figure 1: Sample Topology - TE Link Labels

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.

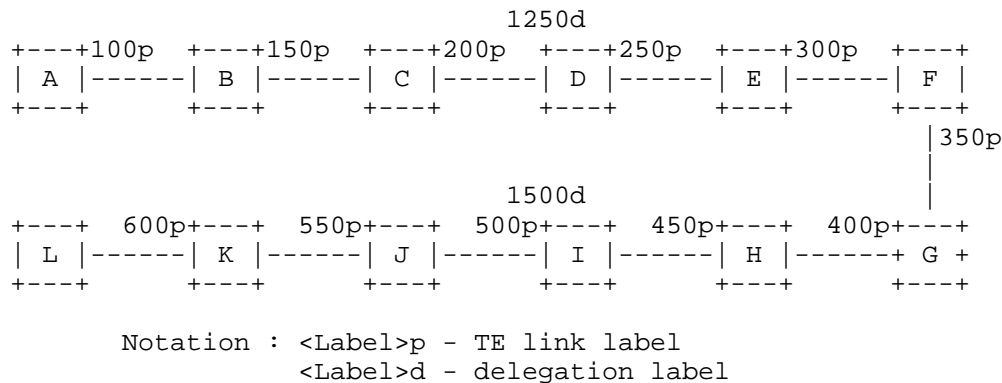


Figure 2: Delegating Label Stack Imposition

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}.

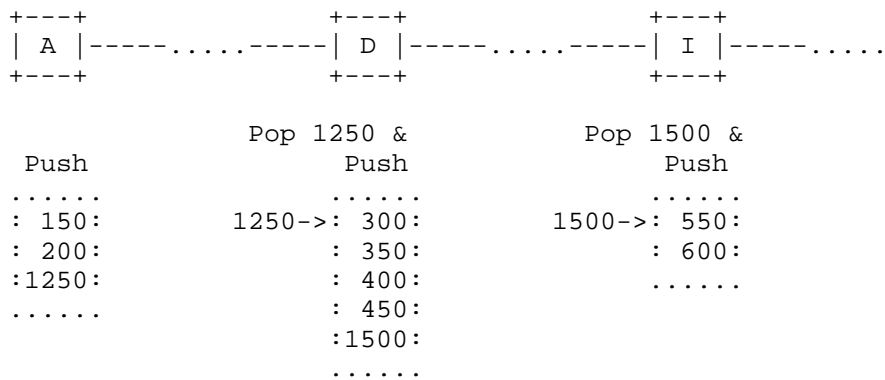


Figure 3: Stack to Reach Delegation Hop

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.

5.1.1.2. 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}.

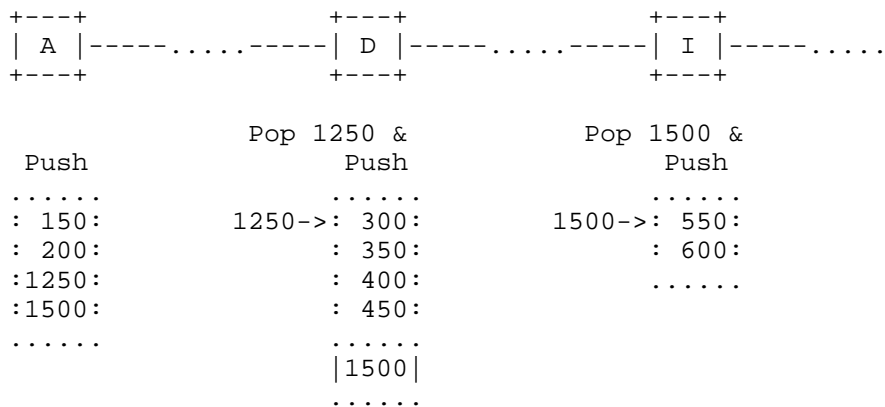


Figure 4: Stack to reach egress

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.

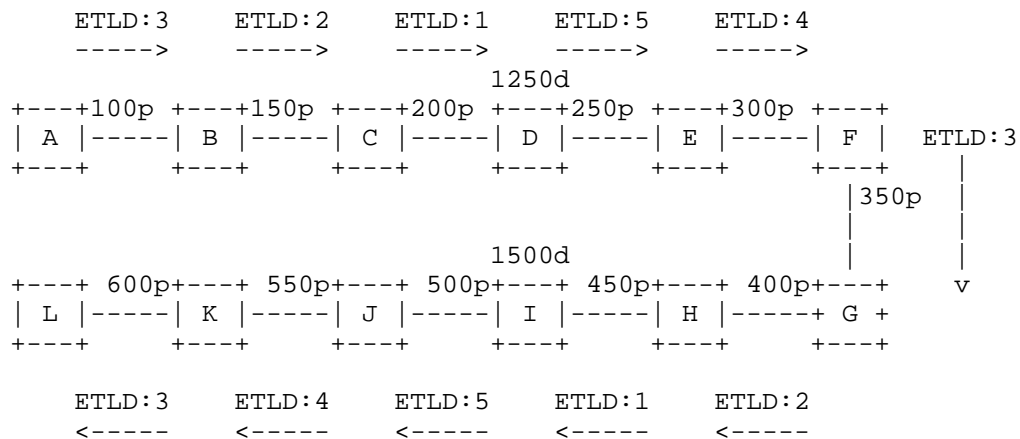
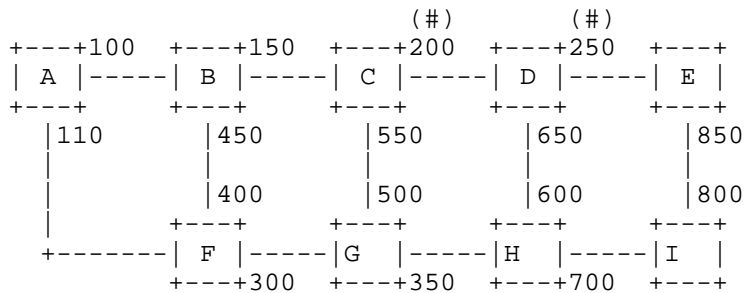


Figure 5: ETLD

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.



Notation : (#) denotes regular labels
Other labels are TE link labels

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

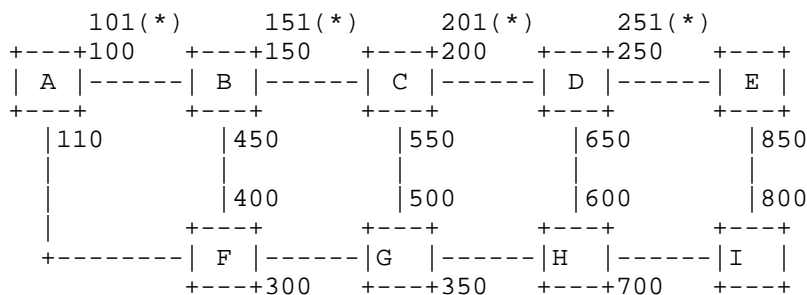
The following section describe how link and node protection works with facility backup protection [RFC4090] for the Segment Routed RSVP-TE tunnels.

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.



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:

- o For an LSR having X neighbors reachable across Y interfaces, the number of unprotected TE link labels is X.
- o For a PLR having X neighbors reachable across Y interfaces, the number of link protected TE link labels is X.
- o 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_OF_ALL(Nx).

The total number of TE link labels is given by:

Unprotected TE link labels +
link protected TE link labels +
node protected TE link labels = $2X + \text{SUM_OF_ALL}(Nx)$

10. Protocol Extensions

10.1. Requirements

The functionality discussed in this document imposes the following requirements on the signaling protocol.

- o 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.
- o 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.

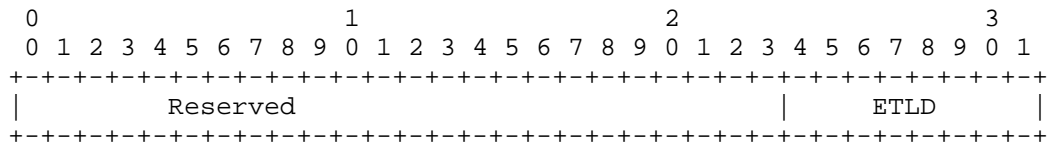


Figure 8: The ETLD Attributes TLV

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 ETLID 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

IANA manages the 'Attribute 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 document introduces three new Attribute Flags.

Bit No.	Name	Attribute Flags	Attribute Path	Attribute Resv	RRO	ERO	Reference
TBD1	TE Link Label	Yes	No	No	No	No	This document (Section 11.2)
TBD3	LSI-D	Yes	No	No	No	Yes	This document (Section 11.4)
TBD5	LSI-D-S2E	Yes	No	No	No	No	This document (Section 11.6)

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.

Type	Name	Allowed on LSP ATTRIBUTES	Allowed on LSP REQUIRED ATTRIBUTES	Allowed on LSP Hop Attributes	Reference
TBD6	ETLD	No	No	Yes	This document (Section 11.7)

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.

Flag	Name	Reference
0x1	Global Label	RFC 3209
TBD2	TE Link Label	This document (Section 11.3)
TBD4	Delegation Label	This document (Section 11.5)

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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <<https://www.rfc-editor.org/info/rfc2205>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC4090] Pan, P., Ed., Swallow, G., Ed., and A. Atlas, Ed., "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, DOI 10.17487/RFC4090, May 2005, <<https://www.rfc-editor.org/info/rfc4090>>.
- [RFC7570] Margaria, C., Ed., Martinelli, G., Balls, S., and B. Wright, "Label Switched Path (LSP) Attribute in the Explicit Route Object (ERO)", RFC 7570, DOI 10.17487/RFC7570, July 2015, <<https://www.rfc-editor.org/info/rfc7570>>.

- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

16.2. Informative 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-11 (work in progress), October 2017.
- [I-D.ietf-teas-rsvp-te-scaling-rec]
Beeram, V., Minei, I., Shakir, R., Pacella, D., and T. Saad, "Techniques to Improve the Scalability of RSVP Traffic Engineering Deployments", draft-ietf-teas-rsvp-te-scaling-rec-08 (work in progress), October 2017.
- [I-D.ietf-teas-sr-rsvp-coexistence-rec]
Sitaraman, H., Beeram, V., Minei, I., and S. Sivabalan, "Recommendations for RSVP-TE and Segment Routing LSP co-existence", draft-ietf-teas-sr-rsvp-coexistence-rec-01 (work in progress), June 2017.
- [RFC2961] Berger, L., Gan, D., Swallow, G., Pan, P., Tommasi, F., and S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, DOI 10.17487/RFC2961, April 2001, <<https://www.rfc-editor.org/info/rfc2961>>.
- [RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010, <<https://www.rfc-editor.org/info/rfc5920>>.

Authors' Addresses

Harish Sitaraman
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: hsitaraman@juniper.net

Vishnu Pavan Beeram
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
US

Email: vbeeram@juniper.net

Tejal Parikh
Verizon
400 International Parkway
Richardson, TX 75081
US

Email: tejal.parikh@verizon.com

Tarek Saad
Cisco Systems
2000 Innovation Drive
Kanata, Ontario K2K 3E8
Canada

Email: tsaad@cisco.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2017

X. Xu
S. Bryant
Huawei
H. Assarpour
Broadcom
H. Shah
Ciena
L. Contreras
Telefonica I+D
D. Bernier
Bell Canada
J. Tantsura
Individual
S. Ma
Juniper
M. Vigoureux
Nokia
June 29, 2017

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

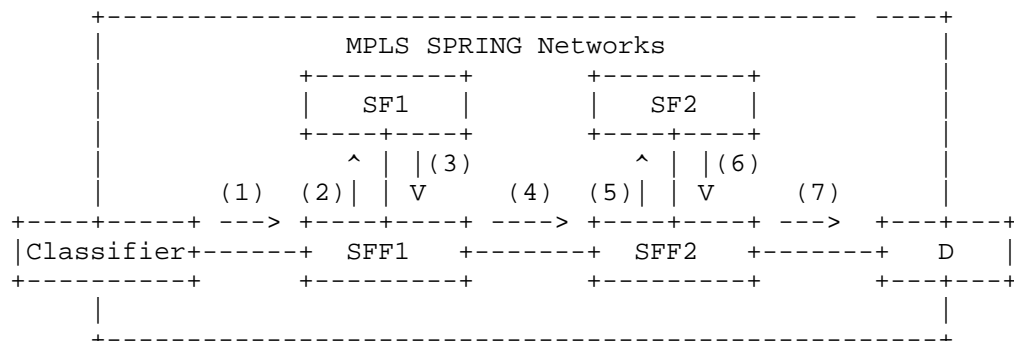


Figure 1: Service Function Chaining in MPLS-SPRING Networks

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

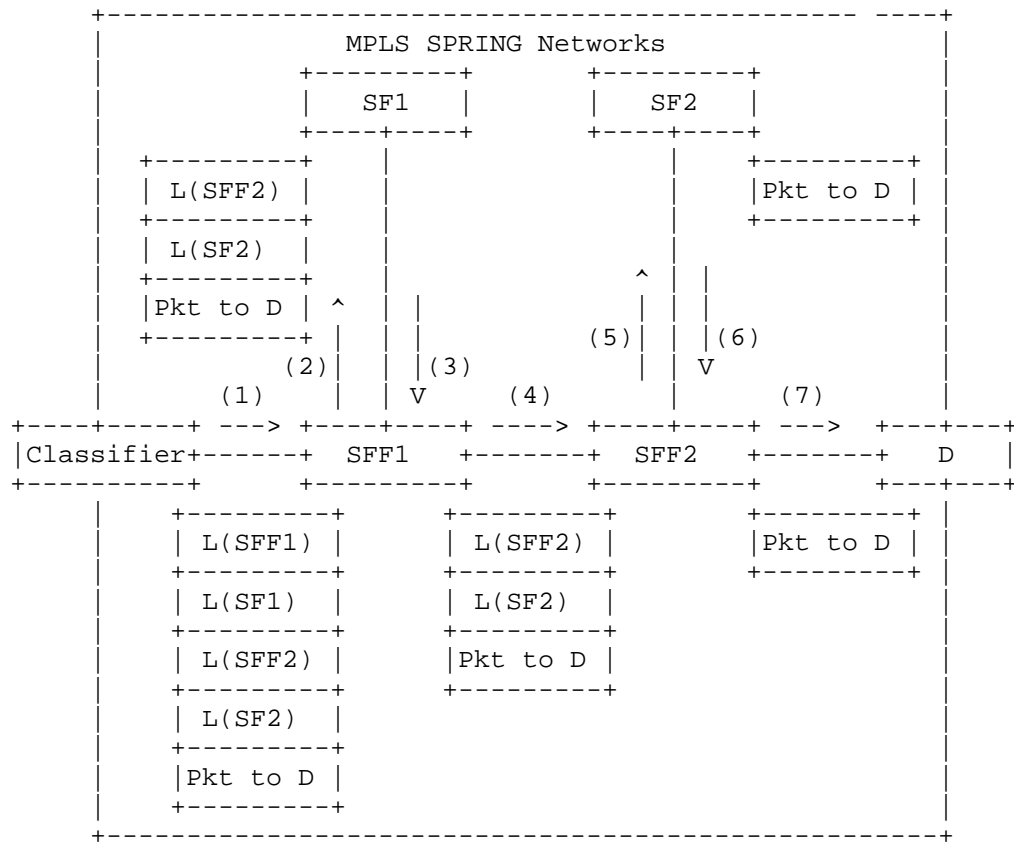


Figure 2: Packet Walk in MPLS underlay

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 SFF 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].

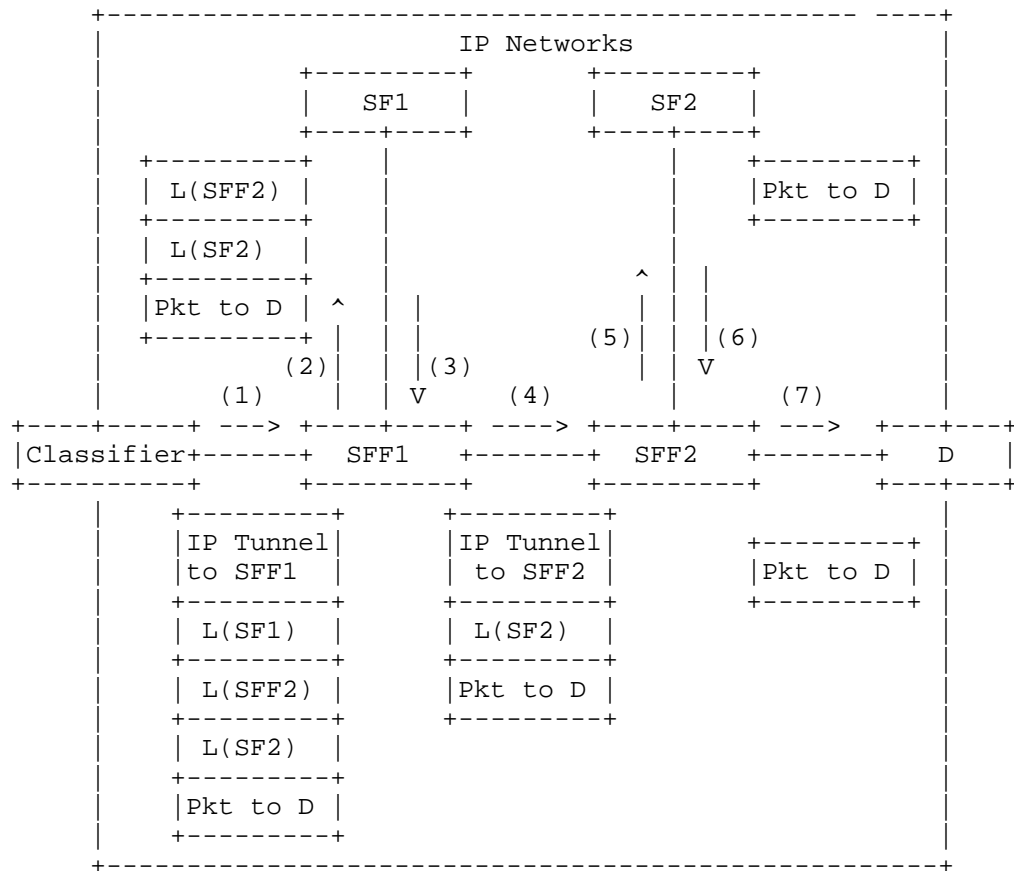


Figure 3: Packet Walk in IP underlay

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 SFF1 (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.,

L(SF1)). 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].

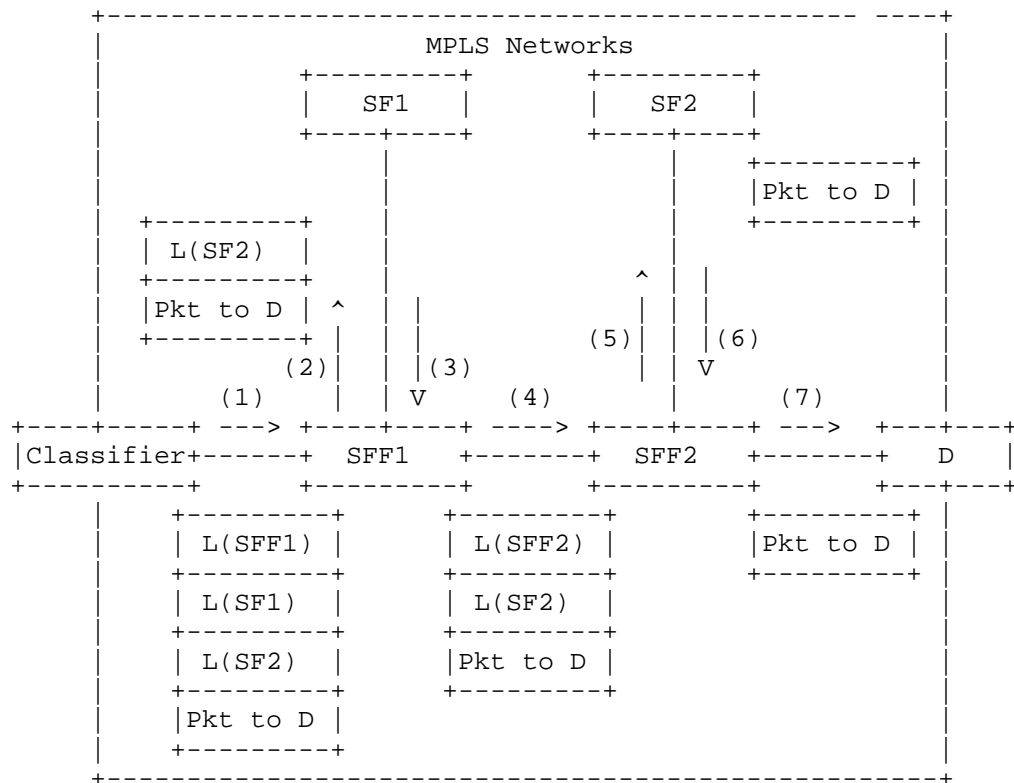


Figure 4: Packet Walk in MPLS underlay

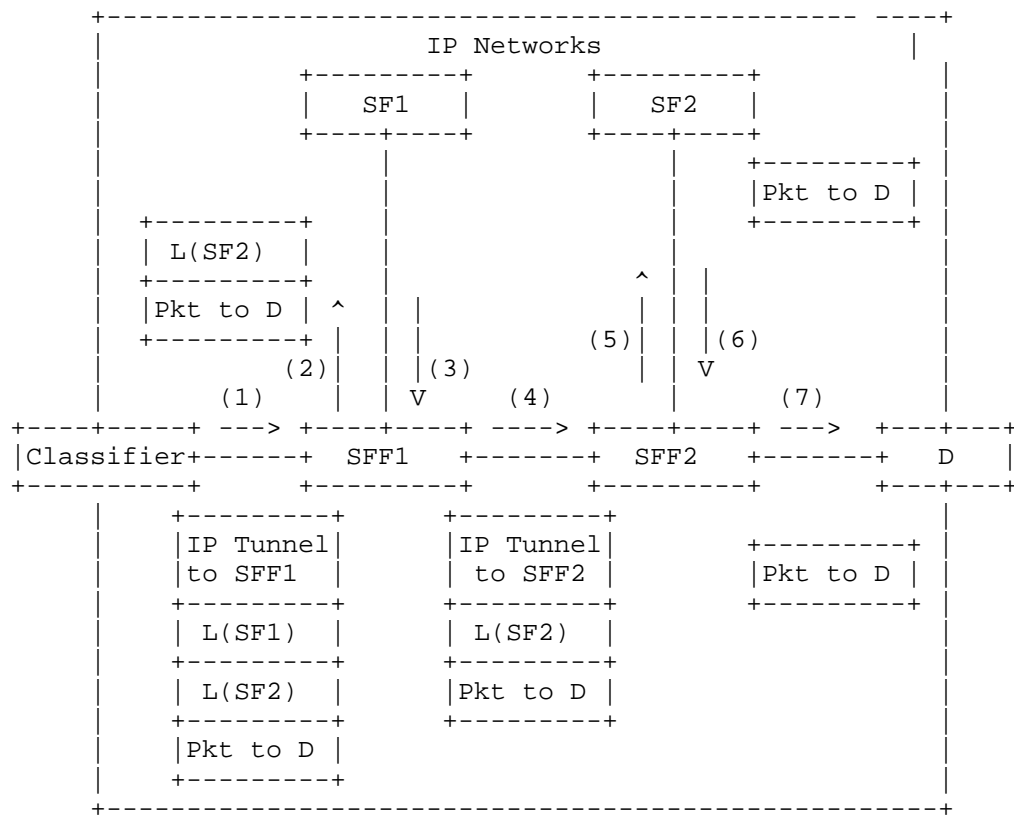


Figure 5: Packet Walk in IP underlay

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.

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

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

7.2. Informative References

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-12 (work in progress), February 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.song-sfc-legacy-sf-mapping]
Song, H., You, J., Yong, L., Jiang, Y., Dunbar, L.,
Bouthors, N., and D. Dolson, "SFC Header Mapping for
Legacy SF", draft-song-sfc-legacy-sf-mapping-08 (work in
progress), September 2016.
- [I-D.xu-mpls-payload-protocol-identifier]
Xu, X., "MPLS Payload Protocol Identifier", draft-xu-mpls-
payload-protocol-identifier-02 (work in progress),
December 2016.
- [I-D.xu-mpls-unified-source-routing-instruction]
Xu, X., Bryant, S., Raszuk, R., Chunduri, U., Contreras,
L., Jalil, L., Assarpour, H., Velde, G., Tantsura, J., and
S. Ma, "Unified Source Routing Instruction using MPLS
Label Stack", draft-xu-mpls-unified-source-routing-
instruction-02 (work in progress), June 2017.
- [RFC4023] Worster, T., Rekhter, Y., and E. Rosen, Ed.,
"Encapsulating MPLS in IP or Generic Routing Encapsulation
(GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005,
<<http://www.rfc-editor.org/info/rfc4023>>.
- [RFC4817] Townsley, M., Pignataro, C., Wainner, S., Seely, T., and
J. Young, "Encapsulation of MPLS over Layer 2 Tunneling
Protocol Version 3", RFC 4817, DOI 10.17487/RFC4817, March
2007, <<http://www.rfc-editor.org/info/rfc4817>>.
- [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black,
"Encapsulating MPLS in UDP", RFC 7510,
DOI 10.17487/RFC7510, April 2015,
<<http://www.rfc-editor.org/info/rfc7510>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function
Chaining (SFC) Architecture", RFC 7665,
DOI 10.17487/RFC7665, October 2015,
<<http://www.rfc-editor.org/info/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

Internet-Draft

June 2017

Martin Vigoureux
Nokia

Email: martin.vigoureux@nokia.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 1, 2018

X. Xu
Huawei
A. Bashandy
Cisco
H. Assarpour
Broadcom
S. Ma
Juniper
W. Henderickx
Nokia
J. Tantsura
Individual
September 28, 2017

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.

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	2
1.1. Requirements Language	3
2. Terminology	3
3. Use Cases	3
4. Packet Forwarding Procedures	4
4.1. Forwarding Entry Construction	5
4.2. Packet Forwarding Procedures	6
5. Contributors	9
6. Acknowledgements	10
7. IANA Considerations	10
8. Security Considerations	10
9. References	10
9.1. Normative References	10
9.2. Informative References	10
Authors' Addresses	13

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:

- o 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).
- o 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-elc] [I-D.ietf-isis-mpls-elc] [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-ls-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.

- o 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.
- o A new IPv4 source routing mechanism which has overcome the security vulnerability issues associated with the traditional IPv4 source routing mechanism.
- o 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.
- o 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 *i*th 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:

- o 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.
- o 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] .
- o 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" .

- o 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 "NH_i".

4.2. Packet Forwarding Procedures

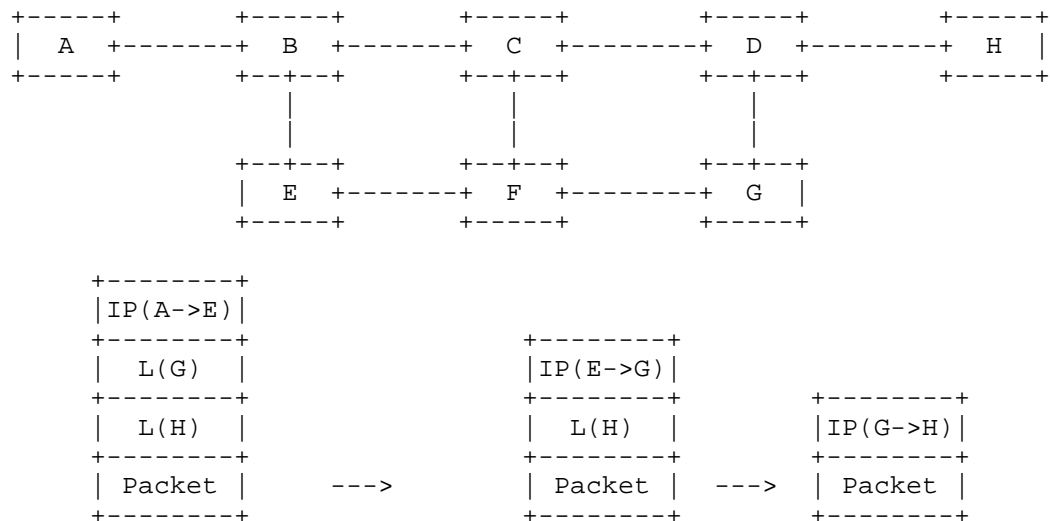


Figure 1

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).

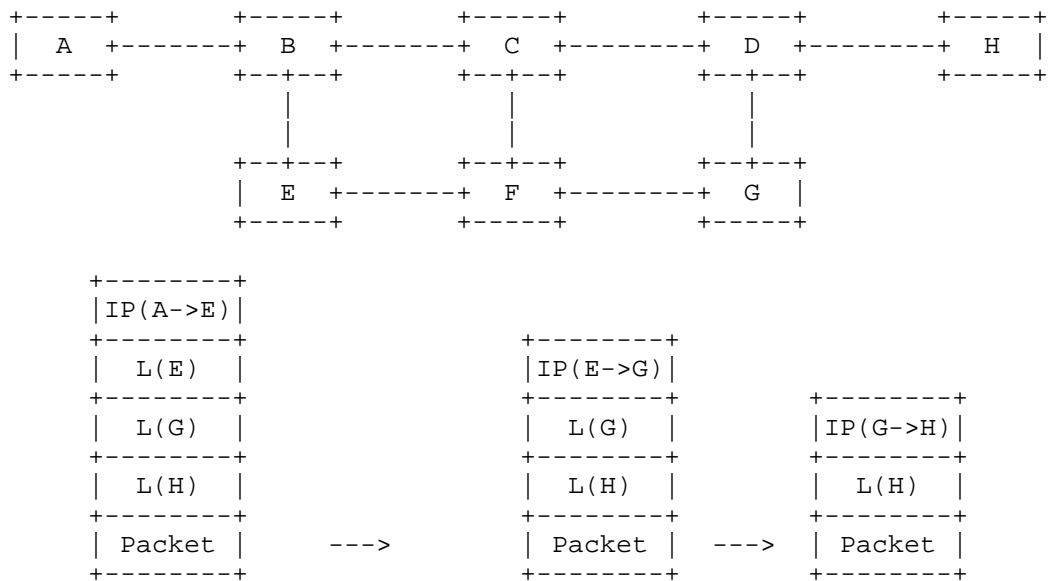


Figure 2

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 an 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: cfilsfil@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

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

9.2. 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-segment-routing-header-07 (work in progress), July 2017.

[I-D.ietf-idr-bgp-ls-segment-routing-msd]
Tantsura, J., Chunduri, U., Mirsky, G., and S. Sivabalan, "Signaling Maximum SID Depth using Border Gateway Protocol Link-State", draft-ietf-idr-bgp-ls-segment-routing-msd-00 (work in progress), July 2017.

[I-D.ietf-idr-bgp-ls-segment-routing-rld]
Velde, G., Henderickx, W., Bocci, M., and K. Patel, "Signalling ERLD using BGP-LS", draft-ietf-idr-bgp-ls-segment-routing-rld-00 (work in progress), July 2017.

[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-mpls-elc]

Xu, X., Kini, S., Sivabalan, S., Filsfils, C., and S. Litkowski, "Signaling Entropy Label Capability Using IS-IS", draft-ietf-isis-mpls-elc-02 (work in progress), October 2016.

[I-D.ietf-isis-segment-routing-extensions]

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

[I-D.ietf-isis-segment-routing-msd]

Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling MSD (Maximum SID Depth) using IS-IS", draft-ietf-isis-segment-routing-msd-04 (work in progress), June 2017.

[I-D.ietf-mpls-spring-entropy-label]

Kini, S., Kompella, K., Sivabalan, S., Litkowski, S., Shakir, R., and j. jeffrant@gmail.com, "Entropy label for SPRING tunnels", draft-ietf-mpls-spring-entropy-label-06 (work in progress), May 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-08 (work in progress), September 2017.

[I-D.ietf-ospf-mpls-elc]

Xu, X., Kini, S., Sivabalan, S., Filsfils, C., and S. Litkowski, "Signaling Entropy Label Capability Using OSPF", draft-ietf-ospf-mpls-elc-04 (work in progress), November 2016.

[I-D.ietf-ospf-segment-routing-extensions]

Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions-19 (work in progress), August 2017.

[I-D.ietf-ospf-segment-routing-msd]

Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling MSD (Maximum SID Depth) using OSPF", draft-ietf-ospf-segment-routing-msd-05 (work in progress), June 2017.

- [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, <<https://www.rfc-editor.org/info/rfc2784>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC4817] Townsley, M., Pignataro, C., Wainner, S., Seely, T., and J. Young, "Encapsulation of MPLS over Layer 2 Tunneling Protocol Version 3", RFC 4817, DOI 10.17487/RFC4817, March 2007, <<https://www.rfc-editor.org/info/rfc4817>>.

- [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black,
"Encapsulating MPLS in UDP", RFC 7510,
DOI 10.17487/RFC7510, April 2015,
<<https://www.rfc-editor.org/info/rfc7510>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function
Chaining (SFC) Architecture", RFC 7665,
DOI 10.17487/RFC7665, October 2015,
<<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC7684] Psenak, P., Gredler, H., Shakir, R., Henderickx, W.,
Tantsura, J., and A. Lindem, "OSPFv2 Prefix/Link Attribute
Advertisement", RFC 7684, DOI 10.17487/RFC7684, November
2015, <<https://www.rfc-editor.org/info/rfc7684>>.
- [RFC7794] Ginsberg, L., Ed., Decraene, B., Previdi, S., Xu, X., and
U. Chunduri, "IS-IS Prefix Attributes for Extended IPv4
and IPv6 Reachability", RFC 7794, DOI 10.17487/RFC7794,
March 2016, <<https://www.rfc-editor.org/info/rfc7794>>.
- [RFC7981] Ginsberg, L., Previdi, S., and M. Chen, "IS-IS Extensions
for Advertising Router Information", RFC 7981,
DOI 10.17487/RFC7981, October 2016,
<<https://www.rfc-editor.org/info/rfc7981>>.

Authors' Addresses

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Ahmed Bashandy
Cisco

Email: bashandy@cisco.com

Hamid Assarpour
Broadcom

Email: hamid.assarpour@broadcom.com

Shaowen Ma
Juniper

Email: mashao@juniper.net

Wim Henderickx
Nokia

Email: wim.henderickx@nokia.com

Jeff Tantsura
Individual

Email: jefftant@gmail.com