MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 19, 2015

R. Torvi
R. Bonica
Juniper Networks
I. Minei
Google, Inc.
M. Conn
D. Pacella
L. Tomotaki
M. Wygant
Verizon
May 18, 2015

LSP Self-Ping
draft-bonica-mpls-self-ping-06

Abstract

When certain RSVP-TE optimizations are implemented, ingress LSRs can receive RSVP RESV messages before forwarding state has been installed on all downstream nodes. According to the RSVP-TE specification, the ingress LSR can forward traffic through an LSP as soon as it receives a RESV message. However, if the ingress LSR forwards traffic through the LSP before forwarding state has been installed on all downstream nodes, traffic can be lost.

This memo describes LSP Self-ping. When an ingress LSR receives an RESV message, it can invoke LSP Self-ping procedures to ensure that forwarding state has been installed on all downstream nodes.

LSP Self-ping is an extremely light-weight mechanism. It does not consume control plane resources on transit or egress LSRs.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on November 19, 2015.

Copyright Notice

Copyright (c) 2015 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

Ingress Label Switching Routers (LSR) use RSVP-TE [RFC3209] to establish MPLS Label Switched Paths. The following paragraphs describe RSVP-TE procedures.

The ingress LSR calculates path between itself and an egress LSR. The calculated path can be either strictly or loosely routed. Having calculated a path, the ingress LSR constructs an RSVP PATH message.

The PATH message includes an Explicit Route Object (ERO) that represents the path between the ingress and egress LSRs.

The ingress LSR forwards the PATH message towards the egress LSR, following the path defined by the ERO. Each transit LSR that receives the PATH message executes admission control procedures. If the transit LSR admits the LSP, it sends the PATH message downstream, to the next node in the ERO.

When the egress LSR receives the PATH message, it binds a label to the LSP. The label can be implicit null, explicit null, or non-null. The egress LSR then installs forwarding state (if necessary), and constructs an RSVP RESV message. The RESV message contains a Label Object that includes the label that has been bound to the LSP.

The egress LSR sends the RESV message upstream towards the ingress LSR. The RESV message visits the same transit LSRs that the PATH message visited, in reverse order. Each transit LSR binds a label to the LSP, updates its forwarding state and updates the RESV message. As a result, the Label Object in the RESV message contains the label that has been bound to the LSP most recently. Finally, the transit LSR sends the RESV message upstream, along the reverse path of the LSP.

When the ingress LSR receives the RESV message, it installs forwarding state. Once the ingress LSR installs forwarding state it can forward traffic through the LSP.

Some implementations optimize the above-described procedure by allowing LSRs to send RESV messages before installing forwarding state. This optimization is desirable, because it allows LSRs to install forwarding state in parallel, thus accelerating the process of LSP signaling and setup. However, this optimization creates a race condition. When the ingress LSR receives a RESV message, some downstream LSRs may not have installed forwarding state yet. If the ingress LSR forwards traffic through the LSP before forwarding state has been installed on all downstream nodes, traffic can be lost.

This memo describes LSP Self-ping. When an ingress LSR receives an RESV message, it can invoke LSP Self-ping procedures to verify that forwarding state has been installed on all downstream nodes. By verifying the installation of downstream forwarding state, the ingress LSR eliminates this particular cause of traffic loss.

LSP Self-ping is an extremely light-weight mechanism. It does not consume control plane resources on transit or egress LSRs.

Although LSP Ping and LSP Self-ping are named similarly, each is a unique protocol. Each protocol listens on its own UDP port and executes is own unique procedures.

2. Applicability

LSP Self-ping is applicable in the following scenario:

- o The ingress LSR receives a RESV message
- o The RESV message indicates that all downstream nodes have begun the process of forwarding state installation
- o The RESV message does not guarantee that all downstream nodes have completed the process of forwarding state installation
- o The ingress LSR needs to confirm that all downstream nodes have completed the process for forwarding state installation
- o The ingress LSR does not need to confirm the correctness of downstream forwarding state, because there is a very high likelihood that downstream forwarding state is correct
- o Control plane resources on the egress LSR may be scarce
- o The need to conserve control plane resources on the egress LSR outweighs the need to determine whether downstream forwarding state is correct

Unlike LSP Ping [RFC4379] and S-BFD [I-D.akiya-bfd-seamless-base], LSP Self-ping is not a general purpose MPLS OAM mechanism. It cannot reliably determine whether downstream forwarding state is correct. For example, if a downstream LSR installs a forwarding state that causes an LSP to terminate at the wrong node, LSP Self-ping will not detect an error. Furthermore, LSP Self-ping fails when either of the following conditions are true:

- o The LSP under test is signaled by the Label Distribution Protocol (LDP) Independent Mode [RFC5036]
- o Reverse Path Forwarding (RPF) [RFC3704] filters are enabled on links that connect the ingress LSR to the egress LSR

While LSP Ping and S-BFD are general purpose OAM mechanisms, they are not applicable in the above described scenario because:

- o LSP Ping consumes control plane resources on the egress LSR

- o An S-BFD implementation either consumes control plane resources on the egress LSR or requires special support for S-BFD on the forwarding plane.

By contrast, LSP Self-ping requires nothing from the egress LSR beyond the ability to forward an IP datagram.

LSP Self-ping’s purpose is to determine whether forwarding state has been installed on all downstream LSRs. Its primary constraint is to minimize its impact on egress LSR performance. This functionality is required during network convergence events that impact a large number of LSPs.

Therefore, LSP Self-ping is applicable in the scenario described above, where the LSP is signaled by RSVP, RPF is not enabled, and the need to conserve control plane resources on the egress LSR outweighs the need to determine whether downstream forwarding state is correct.

3. The LSP Self-ping Message

The LSP Self-ping Message is a User Datagram Protocol (UDP) [RFC0768] packet. If the RSVP messages used to establish the LSP under test were delivered over IPv4 [RFC0791], the UDP datagram MUST be encapsulated in an IPv4 header. If the RSVP messages used to establish the LSP were delivered over IPv6 [RFC2460], the UDP datagram MUST be encapsulated in an IPv6 header.

In either case:

- o The IP Source Address MAY be configurable. By default, it MUST be the address of the egress LSR
- o The IP Destination Address MUST be the address of the ingress LSR
- o The IP Time to Live (TTL) / Hop Count MAY be configurable. By default, it MUST be 255
- o The IP DSCP MAY be configurable. By default, it MUST be CS6 (0x48) [RFC4594]
- o The UDP Source Port MUST be selected from the dynamic range (49152-65535) [RFC6335]
- o The UDP Destination Port MUST be LSP Self-ping. (Value to be assigned by IANA. See Section 7)

UDP packet contents have the following format:

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

LSP Self-ping Message

The Session-ID is a 64-bit field that associates an LSP Self-ping message with an LSP Self-ping session.

4. LSP Self Ping Procedures

In order to verify that an LSP is ready to carry traffic, the ingress LSR creates a short-lived LSP Self-ping session. All session state is maintained locally on the ingress LSR. Session state includes the following information:

- o Session-ID: A 64-bit number that identifies the LSP Self-ping session

- o Retry Counter: The maximum number of times that the ingress LSR probes the LSP before terminating the LSP Self-ping session. The initial value of this variable is determined by configuration.

- o Retry Timer: The number of milliseconds that the LSR waits after probing the LSP. The initial value of this variable is determined by configuration.

- o Status: A boolean variable indicating the completion status of the LSP Self-ping session. The initial value of this variable is FALSE.

Implementation MAY represent the above mentioned information in any format that is convenient to them.

The ingress LSR executes the following procedure until Status equals TRUE or Retry Counter equals zero:

- o Format a LSP Self-ping message.

- o Set the Session-ID in the LSP Self-ping message to the Session-ID mentioned above

- o Send the LSP Self-ping message through the LSP under test

- o Set a timer to expire in Retry Timer milliseconds

- o Wait until either an LSP Self-ping message associated with the session returns or the timer expires. If an LSP Self-ping message associated with the session returns, set Status to TRUE. Otherwise, decrement the Retry Counter. Optionally, increase the value of Retry Timer according to an appropriate back off algorithm.

In the process described above, the ingress LSR addresses an LSP Self-ping message to itself and forwards that message through the LSP under test. If forwarding state has been installed on all downstream LSRs, the egress LSR receives the LSP Self-ping message and determines that it is addressed to the ingress LSR. So, the egress LSR forwards LSP Self-ping message back to the ingress LSR, exactly as it would forward any other IP packet.

The LSP Self-ping message can arrive at the egress LSR with or without an MPLS header, depending on whether the LSP under test executes penultimate hop-popping procedures. If the LSP Self-ping message arrives at the egress LSR with an MPLS header, the egress LSR removes that header.

If the egress LSR’s most preferred route to the ingress LSR is through an LSP, the egress LSR forwards the LSP Self-ping message through that LSP. However, if the egress LSR’s most preferred route to the ingress LSR is not through an LSP, the egress LSR forwards the LSP Self-ping message without MPLS encapsulation.

When an LSP Self-ping session terminates, it returns its completion status to the invoking protocol. For example, if RSVP-TE invokes LSP Self-ping as part of the LSP set-up procedure, LSP Self-ping returns its completion status to RSVP-TE.

5. Bidirectional LSP Procedures

A bidirectional LSP has an active side and a passive side. The active side calculates the ERO and signals the LSP in the forward direction. The passive side reverses the ERO and signals the LSP in the reverse direction.

When LSP Self-ping is applied to a bidirectional LSP:

- o The active side calculates ERO, signals LSP and runs LSP Self-ping

- o The Passive side reverses ERO, signals LSP and runs another instance of LSP Self-ping

- o Neither side forwards traffic through the LSP until local LSP Self-ping returns TRUE

The two LSP Self-ping sessions, mentioned above, are independent of one another. They are not required to have the same Session-ID. Each endpoint can forward traffic through the LSP as soon as the its local LSP Self-ping returns TRUE. Endpoints are not required to wait until both LSP Self-ping sessions have returned TRUE.

6. Rejected Approaches

In a rejected approach, the ingress LSR uses LSP-Ping to verify LSP readiness. This approach was rejected for the following reasons.

While an ingress LSR can control its control plane overhead due to LSP Ping, an egress LSR has no such control. This is because each ingress LSR can, on its own, control the rate of the LSP Ping originated by the LSR, while an egress LSR must respond to all the LSP Pings originated by various ingresses. Furthermore, when an MPLS Echo Request reaches an egress LSR it is sent to the control plane of the egress LSR, which makes egress LSR processing overhead of LSP Ping well above the overhead of its data plane (MPLS/IP forwarding). These factors make LSP Ping problematic as a tool for detecting LSP readiness to carry traffic when dealing with a large number of LSPs.

By contrast, LSP Self-ping does not consume any control plane resources at the egress LSR, and relies solely on the data plane of the egress LSR, making it more suitable as a tool for checking LSP readiness when dealing with a large number of LSPs.

In another rejected approach, the ingress LSR does not verify LSP readiness. Alternatively, it sets a timer when it receives an RSVP RESV message and does not forward traffic through the LSP until the timer expires. This approach was rejected because it is impossible to determine the optimal setting for this timer. If the timer value is set too low, it does not prevent black-holing. If the timer value is set too high, it slows down the process of LSP signalling and setup.

Moreover, the above-mentioned timer is configured on a per-router basis. However, its optimum value is determined by a network-wide behavior. Therefore, changes in the network could require changes to the value of the timer, making the optimal setting of this timer a moving target.

7. IANA Considerations

This memo request that IANA assign a UDP port from the user range (1024-49151) for LSP Self-ping.

8. Security Considerations

LSP Self-ping messages are easily forged. Therefore, an attacker can send the ingress LSR a forged LSP Self-ping message, causing the ingress LSR to terminate the LSP Self-ping session prematurely. In order to mitigate these threats, implementations SHOULD NOT assign Session-ID’s in a predictable manner. Furthermore, operators SHOULD filter LSP Self-ping packets at network ingress points.

9. Acknowledgements

Thanks to Yakov Rekhter, Ravi Singh, Eric Rosen, Eric Osborne, Greg Mirsky and Nobo Akiya for their contributions to this document.

10. References

10.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, March 2004.

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

[RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

10.2. Informative References

[I-D.akiya-bfd-seamless-base] Akiya, N., Pignataro, C., Ward, D., Bhatia, M., and J. Networks, "Seamless Bidirectional Forwarding Detection (S-BFD)", draft-akiya-bfd-seamless-base-03 (work in progress), April 2014.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, August 2006.

Authors' Addresses

Ravi Torvi
Juniper Networks

Email: rtorvi@juniper.net

Ron Bonica
Juniper Networks

Email: rbonica@juniper.net

Ina Minei
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
U.S.A.

Email: inaminei@google.com

Michael Conn
Verizon

Email: michael.e.conn@verizon.com

Dante Pacella
Verizon

Email: dante.j.pacella@verizon.com

Luis Tomotaki
Verizon

Email: luis.tomotaki@verizon.com

Mark Wygant
Verizon

Email: mark.wygant@verizon.com

MPLS
Internet-Draft
Intended status: Informational
Expires: May 5, 2016

S. Bryant
Cisco Systems
S. Bryant
Independent
C. Pignataro
Cisco Systems
M. Chen
Z. Li
Huawei
G. Mirsky
Ericsson

November 2, 2015

MPLS Flow Identification Considerations
draft-bryant-mpls-flow-ident-03

Abstract

This memo discusses the desired capabilities for MPLS flow identification. The key application that needs this is in-band performance monitoring of user data packets.

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 May 5, 2016.

Copyright Notice

Copyright (c) 2015 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

This memo discusses the desired capabilities for MPLS flow identification. The key application that needs this is in-band performance monitoring of user data packets.

There is a need to identify flows in MPLS networks for applications such as packet loss and packet delay measurement. A method of loss and delay measurement in MPLS networks was defined in [RFC6374]. When used to measure packet loss [RFC6374] depends on the use of the injected Operations, Administration, and Maintenance (OAM) packets to designate the beginning and the end of the packet group over which packet loss is being measured. Where the misordering of packets from one group relative to the following group, or misordering of one of the packets being counted relative to the [RFC6374] packet occurs, then an error will occur in the packet loss measurement. In addition, this packet performance measurement system needs to be extended to deal with different granularities of flow and to address a number of the multi-point cases in which two or more ingress Label

Switching Routers (LSRs) could send packets to one or more destinations.

Improvements in link and transmission technologies mean that it may be difficult to assess packet loss using active performance measurement methods with synthetic traffic, due to the very low loss rate in normal operation. That, together with more demanding service level requirements, mean that network operators need to be able to measure the loss of the actual user data traffic by using passive performance measurement methods. Any technique deployed needs to be transparent to the end user, and it needs to be assumed that they will not take any active part in the measurement process. Indeed it is important that any flow identification technique be invisible to them and that no remnant of the identification of measurement process leak into their network.

Additionally where there are multiple traffic sources, such as in multi-point to point and multi-point to multi-point network environments there needs to be a method whereby the sink can distinguish between packets from the various sources, that is to say, that a multi-point to multi-point measurement model needs to be developed.

2. 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].

3. Loss Measurement Considerations

Modern networks, if not oversubscribed, normally drop very few packets, thus packet loss measurement is highly sensitive to counter errors. Without some form of coloring or batch marking such as that proposed in [I-D.tempia-ippm-p3m] it may not be possible to achieve the required accuracy in the loss measurement of customer data traffic. Thus where accuracy better than the data link loss performance of a modern optical network is required, it may be economically advantageous, or even a technical requirement, to include some form of marking in the packets to assign each packet to a particular counter.

Where this level of accuracy is required and the traffic between a source-destination pair is subject to Equal-Cost Multipath (ECMP) a demarcation mechanism is needed to group the packets into batches. Once a batch is correlated at both ingress and egress, the packet accounting mechanism is then able to operate on the batch of packets which can be accounted for at both the packet ingress and the packet

egress. Errors in the accounting are particularly acute in Label Switched Paths (LSPs) subjected to ECMP because the network transit time will be different for the various ECMP paths since:

a. The packets may traverse different sets of LSRs.

b. The packets may depart from different interfaces on different line cards on LSRs

c. The packets may arrive at different interfaces on different line cards on LSRs.

A consideration in modifying the identity label (the MPLS label ordinarily used to identify the LSP, Virtual Private Network, Pseudowire etc) to indicate the batch is the impact that this has on the path chosen by the ECMP mechanism. When the member of the ECMP path set is chosen by deep packet inspection a change of batch represented by a change of identity label will have no impact on the ECMP path. Where the path member is chosen by reference to an entropy label [RFC6790] then changing the batch identifier will not result in a change to the chosen ECMP path. ECMP is so pervasive in multi-point to (multi-) point networks that some method of avoiding accounting errors introduced by ECMP needs to be supported.

4. Delay Measurement Considerations

Most of the existing delay measurement methods are active measurement that depend on the extra injected test packet to evaluate the delay of a path. With the active measurement method, the rate, numbers and interval between the injected packets may affect the accuracy of the results. Also, for injected test packets, these may not be co-routed with the data traffic due to ECMP. Thus there exists a requirement to measure the delay of the real traffic.

For combined loss-delay measurements, both the loss and the delay considerations apply.

5. Units of identification

The most basic unit of identification is the identity of the node that processed the packet on its entry to the MPLS network. However, the required unit of identification may vary depending on the use case for accounting, performance measurement or other types of packet observations. In particular note that there may be a need to impose identify at several different layers of the MPLS label stack.

This document considers following units of identifications:

- o Per source LSR - everything from one source is aggregated.

- o Per group of LSPs chosen by an ingress LSR - an ingress LSP aggregates group of LSPs (ex: all LSPs of a tunnel).

- o Per LSP - the basic form.

- o Per flow [RFC6790] within an LSP - fine graining method.

Note that a fine grained identity resolution is needed when there is a need to perform these operations on a flow not readily identified by some other element in the label stack. Such fine grained resolution may be possible by deep packet inspection, but this may not always be possible, or it may be desired to minimise processing costs by doing this only in entry to the network, and adding a suitable identifier to the packet for reference by other network elements. An example of such a fine grained case might be traffic from a specific application, or from a specific application from a specific source, particularly if matters related to service level agreement or application performance were being investigated.

We can thus characterize the identification requirement in the following broad terms:

- o There needs to be some way for an egress LSR to identify the ingress LSR with an appropriate degree of scope. This concept is discussed further in Section 7.

- o There needs to be a way to identify a specific LSP at the egress node. This allows for the case of instrumenting multiple LSPs operate between the same pair of nodes. In such cases the identity of the ingress LSR is insufficient.

- o In order to conserve resources such as labels, counters and/or compute cycles it may be desirable to identify an LSP group so that a operation can be performed on the group as an aggregate.

- o There needs to be a way to identify a flow within an LSP. This is necessary when investigating a specific flow that has been aggregated into an LSP.

The unit of identification and the method of determining which packets constitute a flow will be application or use-case specific and is out of scope of this memo.

6. Types of LSP

We need to consider a number of types of LSP. The two simplest types to monitor are point to point LSPs and point to multi-point LSPs. The ingress LSR for a point to point LSP, such as those created using the Resource Reservation Protocol - Traffic Engineering (RSVP-TE) [RFC5420] signalling protocol, or those that conform to the MPLS Transport Profile (MPLS-TP) [RFC5654] may be identified by inspection of the top label in the stack, since at any provider-edge (PE) or provider (P) router on the path this is unique to the ingress-egress pair at every hop at a given layer in the LSP hierarchy. Provided that penultimate hop popping is disabled, the identity of the ingress LSR of a point to point LSP is available at the egress LSR and thus determining the identity of the ingress LSR must be regarded as a solved problem. Note however that the identity of a flow cannot to be determined without further information being carried in the packet, or gleaned from some aspect of the packet payload.

In the case of a point to multi-point LSP, and in the absence of Penultimate Hop Popping (PHP) the identity of the ingress LSR may also be inferred from the top label. However, it may not possible to adequately identify the flow from the top label alone, and thus further information may need to be carried in the packet, or gleaned from some aspect of the packet payload. In designing any solution it is desirable that a common flow identity solution be used for both point to point and point to multi-point LSP types. Similarly it is desirable that a common method of LSP group identification be used. In the above cases, a context label [RFC5331] needs to be used to provide the required identity information. This is widely supported MPLS feature.

A more interesting case is the case of a multi-point to point LSP. In this case the same label is normally used by multiple ingress or upstream LSRs and hence source identification is not possible by inspection of the top label by the egress LSRs. It is therefore necessary for a packet to be able to explicitly convey any of the identity types described in Section 5.

Similarly, in the case of a multi-point to multi-point LSP the same label is normally used by multiple ingress or upstream LSRs and hence source identification is not possible by inspection of the top label by egress LSRs. The various types of identity described in Section 5 are again needed. Note however, that the scope of the identity may be constrained to be unique within the set of multi-point to multi-point LSPs terminating on any common node.

7. Network Scope

The scope of identification can be constrained to the set of flows that are uniquely identifiable at an ingress LSR, or some aggregation thereof. There is no question of an ingress LSR seeking assistance from outside the MPLS protocol domain.

In any solution that constrains itself to carrying the required identity in the MPLS label stack rather than in some different associated data structure, constraints on the label stack size imply that the scope of identity reside within that MPLS domain. For similar reasons the identity scope of a component of an LSP should be constrained to the scope of that LSP.

8. Backwards Compatibility

In any network it is unlikely that all LSRs will have the same capability to support the methods of identification discussed in this memo. It is therefore an important constraint on any flow identity solution that it is backwards compatible with deployed MPLS equipment to the extent that deploying the new feature will not disable anything that currently works on a legacy equipment.

This is particularly the case when the deployment is incremental or when the feature is not required for all LSRs or all LSPs. Thus in broad the flow identification design MUST support the co-existence of both LSRs that can, and cannot, identify the traffic components described in Section 5. In addition the identification of the traffic components described in Section 5 MUST be an optional feature that is disabled by default. As a design simplification, a solution MAY require that all egress LSRs of a point to multipoint or a multi-point to multipoint LSP support the identification type in use so that a single packet can be correctly processed by all egress devices. The corollary of this last point is that either all egress LSRs are enabled to support the required identity type, or none of them are.

9. Dataplane

There is a huge installed base of MPLS equipment, typically this type of equipment remains in service for an extended period of time, and in many cases hardware constraints mean that it is not possible to upgrade its dataplane functionality. Changes to the MPLS data plane are therefore expensive to implement, add complexity to the network, and may significantly impact the deployability of a solution that requires such changes. For these reasons, the MPLS designers have set a very high bar to changes to the MPLS data plane, and only a very small number have been adopted. Hence, it is important that the

method of identification must minimize changes to the MPLS data plane. Ideally method(s) of identification that require no changes to the MPLS data plane should be given preferential consideration. If a method of identification makes a change to the data plane is chosen it will need to have a significant advantage over any method that makes no change, and the advantage of the approach will need to be carefully evaluated and documented. If a change is necessary to the MPLS data plane proves necessary, it should be (a) be as small a change as possible and (b) be a general purpose method so as to maximise its use for future applications. It is imperative that, as far as can be foreseen, any necessary change made to the MPLS data plane does not impose any foreseeable future limitation on the MPLS data plane.

Stack size is an issue with many MPLS implementations both as a result of hardware limitations, and due to the impact on networks and applications where a large number of small payloads need to be transported In particular one MPLS payload may be carried inside another. For example one LSP may be carried over another LSP, or a PW or similar multiplexing construct may be carried over an LSP and identification may be required at both layers. Of particular concern is the implementation of low cost edge LSRs that for cost reasons have a significant limit on the number of Label Stack Elements (LSEs) that they can impose or dispose. Therefore, any method of identity MUST NOT consume an excessive number of unique labels, and MUST NOT result in an excessive increase in the size of the label stack.

The MPLS data plane design provides two types of special purpose labels: the original 16 reserved labels and the much larger set of special purpose labels defined in [RFC7274]. The original reserved labels need one LSE, and the newer [RFC7274] special purpose labels need two LSEs. Given the tiny number of original reserved labels, it is core to the MPLS design philosophy that this scarce resource is only used when it is absolutely necessary. Using a single LSE reserved or special purpose label to encode flow identity thus requires two stack entries, one for the reserved label and one for the flow identity. The larger set of [RFC7274] labels requires two labels stack entries for the special purpose label itself and hence a total of three label stack entries to encode the flow identity.

The use of special purpose labels (SPL) [RFC7274] as part of a method to encode the identity information therefore has a number of undesirable implications for the data plane and hence whilst a solution may use SPL(s), methods that do not require SPLs need to be carefully considered.

10. Control Plane

Any flow identity design should both seek to minimise the complexity of the control plane and should minimise the amount of label co-ordination needed amongst LSRs.

11. Privacy Considerations

The inclusion of originating and/or flow information in a packet provides more identity information and hence potentially degrades the privacy of the communication. Recent IETF concerns on pervasive monitoring [RFC7258] would lead it to prefer a solution that does not degrade the privacy of user traffic below that of an MPLS network not implementing the flow identification feature. The minimizing the scope of the identity indication can be useful in minimizing the observability of the flow characteristics.

12. Security Considerations

Any solution to the flow identification needs must not degrade the security of the MPLS network below that of an equivalent network not deploying the specified identity solution. Propagation of identification information outside the MPLS network imposing it must be disabled by default. Any solution should provide for the restriction of the identity information to those components of the network that need to know it. It is thus desirable to limit the knowledge of the identify of an endpoint to only those LSRs that need to participate in traffic flow.

13. IANA Considerations

This memo has no IANA considerations.

14. Acknowledgements

The authors thank Nobo Akiya (nobo@cisco.com), Nagendra Kumar Nainar (naikumar@cisco.com) and George Swallow (swallow@cisco.com) for their comments.

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, <http://www.rfc-editor.org/info/rfc2119>.

15.2. Informative References

[I-D.tempia-ippm-p3m]
Capello, A., Cociglio, M., Fioccola, G., Castaldelli, L., and A. Bonda, "A packet based method for passive performance monitoring", draft-tempia-ippm-p3m-02 (work in progress), October 2015.

[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, <http://www.rfc-editor.org/info/rfc5331>.

[RFC5420] Farrel, A., Ed., Papadimitriou, D., Vasseur, JP., and A. Ayyangarps, "Encoding of Attributes for MPLS LSP Establishment Using Resource Reservation Protocol Traffic Engineering (RSVP-TE)", RFC 5420, DOI 10.17487/RFC5420, February 2009, <http://www.rfc-editor.org/info/rfc5420>.

[RFC5654] Niven-Jenkins, B., Ed., Brungard, D., Ed., Betts, M., Ed., Sprecher, N., and S. Ueno, "Requirements of an MPLS Transport Profile", RFC 5654, DOI 10.17487/RFC5654, September 2009, <http://www.rfc-editor.org/info/rfc5654>.

[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>.

[RFC7274] Kompella, K., Andersson, L., and A. Farrel, "Allocating and Retiring Special-Purpose MPLS Labels", RFC 7274, DOI 10.17487/RFC7274, June 2014, <http://www.rfc-editor.org/info/rfc7274>.

Authors' Addresses

Stewart Bryant
Cisco Systems

Email: stbryant@cisco.com

Stewart Bryant
Independent

Email: stewart.bryant@gmail.com

Carlos Pignataro
Cisco Systems

Email: cpignata@cisco.com

Mach Chen
Huawei

Email: mach.chen@huawei.com

Zhenbin Li
Huawei

Email: lizhenbin@huawei.com

Gregory Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 10, 2021

S. Bryant
Futurewei Technologies Inc.
G. Swallow
Southend Technical Center
S. Sivabalan
Ciena Corporation

December 07, 2020

A Simple Control Protocol for MPLS SFLs
draft-bryant-mpls-sfl-control-09

Abstract

In draft-ietf-mpls-sfl-framework the concept of MPLS synonymous flow labels (SFL) was introduced. This document describes a simple control protocol that runs over an associated control header to request, withdraw, and extend the lifetime of such labels. It is not the only control protocol that moght be used to support SFL, but it has the benefit of being able to be used without modifying of the existing MPLS control prodocols. The existance of this design is not intended to restrict the ability to enhance an existing MPLS control protocol to add a similar capability.

A Querier MUST wait a configured time (suggested wait of 60 seconds) before re-attempting a Withdraw request. No more than three Withdraw requests SHOULD be made. These restricctions are to prevent overloading the control plane of the actioning 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 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 June 10, 2021.

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

In [I-D.ietf-mpls-sfl-framework] the concept of MPLS synonymous flow labels (SFL) was introduced. This document describes a simple control protocol, for use in a well-managed MPLS network, that runs over an associated control header to request, withdraw, and extend the lifetime of such labels. It is not the only control protocol that moght be used to support SFL, but it has the benefit of being able to be used without modifying of the existing MPLS control

prodocols. The existance of this design is not intended to restrict the ability to enhance an existing MPLS control protocol to add a similar capability.

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. SFL Control

EDITOR'S note look at whether we continue to use RFC6374 terms query respond, or normal client server terms.

This section describes the process by which the [RFC6374] Querier requests SFLs, the process by which the [RFC6374] Responder sends them to the Querier, and the process for managing the SFL lifetime. SFL Control Messages are carried over the SFL Control ACH. The SFL ACH is carried over a Pseudowire(PW) in place of the PW Control Word (CW), over an MPLS LSP using the GAL, or over some other mutually agreed path. Similarly the response may be returned over a PW, over a bidirectional LSP or over some other mutually agreed path. See Section 4.

3.1. SFL Control Message

The format of an SFL Control message, which follows the Associated Channel Header (ACH), is as follows:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Flags |  Control Code |        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Session Identifier          | SFL Batch |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Lifetime (seconds)              |   Num SFL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 SFL 0                 |      LFlags           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 SFL n                 |      LFlags           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Forwarding Equivalence Class (FEC)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: SFL Control Message Format

Reserved fields MUST be set to 0 and ignored upon receipt. The possible values for the remaining fields are as follows:

Version Protocol version. Set to zero in this specification.

Flags Message control flags.

Control Code Code identifying the query or response type.

Message Length Total length of this message in bytes.

Session Identifier Set arbitrarily by the querier and used as a message handle.

SFL Batch (6 bits) Used where the SFLs for this Session Identifier are managed across multiple SFL Control Messages. A given set of SFLs MUST be retained in the same batch.

Lifetime The lifetime in seconds of the SFLs in this message. In a Query message it is the requested lifetime. In a Response message it is the lifetime that the SFLs have been allocated for by the Responder. The Querier MUST NOT use an SFL after expiry of its lifetime, a Responder MUST make the SFL available for at least its lifetime.

Num SFL The number of SFLs in this SFL Batch. This MUST be constant for the lifetime of the batch.

SFL n The n'th SFL carried in this TLV. This is an MPLS label which is a component of a label stack entry as defined in Section 2.1 of [RFC3032]. The position of a label within a batch is constant for the lifetime of the batch. Enumeration starts at zero.

LFlags The set of flags associated with the immediately preceding SFL. See below.

FEC The Forwarding Equivalence Class that the SFLs in this TLV correspond to. This is encoded as per Section 3.4.1 of [RFC5036].

Flags: The format of the Flags field is shown below.

```
+-+-+-+-+
|R|0|0|0|
+-+-+-+-+
```

SFL Control Message Flags.

The meanings of the flag bits are:

R: Query/Response indicator. Set to 0 for a Query and set to 1 for a Response.

0: Set to zero by the Sender and ignored by the Receiver.

Control Code: Set as follows according to whether the message is a Query or a Response as identified by the R flag.

For a Query:

0x0: SFL Request. This indicates that the responder is requested to allocate the set of SFLs marked with the R LFlag in this message.

0x1: SFL Refresh. This indicates that the responder is requested to refresh the set of SFLs marked with the V LFlag in this message.

0x2: SFL Withdraw. This indicates that the querier will no longer use the set of SFLs marked with the V Lflag and the responder may expire their lifetime.

For a Response:

Codes 0x0-0xF are reserved for non-error responses.

0x1: SFL Grant. This indicates that the responder allocated the set of SFLs marked with the A LFlag in this Message.

0x2: SFL Refresh-Ack. This indicates that the responder has refreshed the set of SFLs marked with the V LFlag in this message, and the lifetime is now as indicated by the lifetime field.

0x3: SFL Withdraw-Ack. This indicates that the responder has received the Withdraw message and will withdraw the SFLs

0x10: Unspecified Error. Indicates that the operation failed for an unspecified reason.

0x11: SFL-Unable. The Responder was unable to satisfy the SFL Request. The details of the failure can be determined by comparing the Request and Grant messages.

Editors Note - We need to revisit the RFC6374 errors and the protocol to see if we need some more error codes.

The LFlags field is defined as follows:

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

Figure 2: LFlags Bit Definition

Where:

0 (Valid (V)) The Label value of the corresponding SFL is valid. In an SFL Request setting the V Lflag indicates a request for the specified label value. Where an SFL has a valid flag clear in a request message this indicates that any SFL value is acceptable.

1 (Request (R)) Indicates to the Querier that this member of the SFL batch is requested. Where a value is specified in the request, but the Responder is unable honour that request, no SFL is allocated and the corresponding A flag MUST be cleared.

2 (Allocated (A) Indicates to the Querier that this SFL was allocated.

3 (Withdraw (W)) Indicates to the Responder that this SFL is to be withdrawn and to the Querier that the withdrawal has been carried out.

MBZ MUST be sent as zero and ignored on receive.

A flag value of one is true/set and a flag value of zero is false/clear. The use of these bits is described in more detail in the following sub-sections.

3.2. SFL Control Procedures

3.2.1. Request/Grant

To request a batch of SFLs the Querier constructs an SFL Control Request, encapsulates it in an SFL Control ACH and sends it to the Responder via an appropriate path. The Querier sets the Control Message Flag to Query and the Control Code to Request. The Querier chooses a session identifier as a handle for this transaction and as a way of binding this batch of SFLs to other operations that will use members of this SFL batch. Since members of the batch are treated as a group, the SFL Batch identifier is used to identify different SFL batches used in conjunction with the same session identifier.

The Querier sets the requested lifetime. This is the number of seconds from the time of the query to the time when the batch of SFLs will expire unless refreshed.

The Num SFL field is set to the SFL batch size.

Each SFL is set as follows: if a specific value is requested (for example for continuity across system restarts) this is written into the SFL n field and the V LFlag set. Otherwise, and including spare SFLs where an allocation is not requested, the label value is set to zero and the V LFlag is cleared. For each SFL entry where an allocation is requested the R LFlag is set. All other LFlags are cleared.

The Forwarding Equivalence Class (FEC) is set to the FEC for which the SFLs are requested.

The Message Length is determined and filled in.

The Responder proceeds as follows:

The Responder sets the control Message Flag to Response and initially sets the Control Code to Grant.

For each SFL with an R flag set, the Responder determines whether it can honour the request, if so sets the A Lflag, and if the SFL value in the query was zero it overwrites it with the allocated SFL label value. In all other cases it leaves the SFL value and LFlag unchanged.

The lifetime field is updated with the lifetime of the SFLs if this is different from the requested lifetime.

All other fields in the Query message are left unchanged and the message is sent back to the Querier using the signaled or previously agreed message path.

Where the offered lifetime is other than the requested lifetime the Querier may accept the proposed value, or withdraw the SFLs and attempt to negotiate a new set of SFLs with a different lifetime.

If the Responder is unable to allocate all of the requested SFLs it MUST respond with a response code of SFL-Unable. The Querier MUST determine whether the allocated SFLs were adequate for its purposes and MUST send a withdraw if there are not adequate. A Querier MUST NOT attempt to hoard labels in the hope that the residual labels needed may become available in the future.

A Querier MUST wait a configured time (suggested wait of 60 seconds) before re-attempting negotiation for a resource. Any failure to negotiate the required resources MUST be notified through the management interface of both Querier and Responder.

A Querier MUST NOT send an expired SFL to a Responder since to do so may invalidate another SFL operation.

3.2.2. Refresh

To request the lifetime refresh of a batch of SFLs the Querier constructs an SFL Refresh Request, encapsulates it in an SFL Control ACH and sends it to the Responder via an appropriate path. The Querier sets the Control Message Flag to Query and the Control Code to Refresh. The Querier uses the session identifier and the SFL Batch identifier that it used to request this SFL batch.

The Querier sets the requested lifetime. This is the number of seconds from the time of the query to the time when the batch of SFLs will expire unless refreshed.

The Querier sets the Num SFL field to the SFL batch size.

The Querier sets each SFL as follows: the allocated SFL label value is written into the SFL n field and the V LFlag set. All other LFlags are cleared.

The Forwarding Equivalence Class (FEC) is set to the FEC for which the SFLs are requested.

The Message Length is determined and filled in.

The Responder proceeds as follows:

The Responder sets the control Message Flag to Response and sets the Control Code to Refresh-Ack.

The Responder sets the lifetime to the lifetime of the SFL.

All other fields in the Query message are left unchanged and the message is sent back to the Querier using the signaled or previously agreed message path.

Where the offered lifetime is other than the requested lifetime the Querier may accept the proposed value, or withdraw the SFLs and attempt to negotiate a new set of SFLs with a different lifetime.

A Querier MUST wait a configured time (suggested wait of 60 seconds) before re-attempting negotiation for a resource. Any failure to negotiate the required resources MUST be notified through the management interface of both Querier and Responder.

3.2.3. Withdraw

To request the withdrawal of some or all of a batch of SFLs the Querier constructs an SFL Withdraw Request, encapsulates it in an SFL Control ACH and sends it to the Responder via an appropriate path. The Querier sets the Control Message Flag to Query and the Control Code to Withdraw. It uses the session identifier and the SFL Batch identifier that it used to request this SFL batch.

The Querier sets the requested lifetime to zero.

The Querier sets the Num SFL field to the SFL batch size.

Each SFL being withdrawn is set as follows: the allocated SFL label value is written into the SFL n field and the V and W LFlags set. All other LFlags are cleared.

The Forwarding Equivalence Class (FEC) is set to the FEC for which the SFLs are requested.

The Message Length field is determined and filled in.

The Responder proceeds as follows:

The Responder sets the control Message Flag to Response and sets the Control Code to Withdraw-Ack.

All other fields in the Query message are left unchanged and the message is sent back to the Querier using the signaled or previously agreed message path.

A Querier MUST wait a configured time (suggested wait of 60 seconds) before re-attempting a Withdraw request. No more than three Withdraw requests SHOULD be made. These restricctions are to prevent overloading the control plane of the actioning router.

3.2.4. Timer Accuracy

The lifetime of SFLs is expected to be sufficiently long that there are no significant constraints on timer accuracy. A node should be conservative in its assumptions concerning the lifetime of an SFL. A Querier MUST stop using a SFL significantly before the expiry of its lifetime and a Responder must maintain an SFL in active operation

significantly beyond nominal expiry. A margin of the order of minutes is RECOMMENDED.

4. Return Path

Where the LSP (or other MPLS construct) is multi-point to point, or multi-point to multi- point the RFC6374 Address TLV MUST be included in Query packet, even if the response is requested in-band, since this is needed to provide the necessary return address for this request.

EDITIORS NOTE - Look at this text and see if we need to make changes regarding operation over IP.

5. Privacy Considerations

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.

6. Security Considerations

It is assumed that this protocol is run in a well-managed MPLS network with strict access controls preventing unwanted parties from generating MPLS packets. The control protocol described in this memo thus introduced no additional MPLS security vulnerabilities.

7. IANA Considerations

7.1. Allocation of MPLS Generalized Associated Channel (G-ACh) Type

As per the IANA considerations in [RFC5586], IANA is requested to allocate the following Channel Type in the "MPLS Generalized Associated Channel (G-ACh) Types" registry:

Value	Description	TLV Follows	Reference
0x0XXX	SFL Control	No	This

A value of 0x5A is suggested.

7.2. Creation of SFL Simple Control Code Registry

IANA is requested to created a new "SFL Simple Control Code" registry within the Generic Associated Channel (G-ACh) Parameters namespace. This registry is divided into two separate parts, one for Query Codes and the other for Response Codes, with formats and initial allocations as follows:

Query Codes

Code	Description	Reference
0x0	SFL Request	This
0x1	SFL Refresh	This
0x2	SFL Withdraw	This

Response Codes

Code	Description	Reference
0x0	Reserved	This
0x1	SFL Grant	This
0x2	SFL Refresh-Ack	This
0x3	SFL Withdraw-Ack	This
0x10	Unspecified Error	This
0x11	SFL-Unable	

IANA should indicate that the values 0x0 - 0xF in the Response Code section are reserved for non-error response codes.

The range of the Code field is 0 - 255.

The allocation policy for this registry is IETF Review.

8. Acknowledgments

The authors thank Haomian Zheng for his review comments.

RFC Editor please remove this note which is used to force the following references to appear [RFC3032] [RFC5036]

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>.

[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>.

[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>.

[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>.

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <https://www.rfc-editor.org/info/rfc6374>.

[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>.

9.2. Informative References

[I-D.ietf-mpls-sfl-framework]
Bryant, S., Chen, M., Swallow, G., Sivabalan, S., and G. Mirsky, "Synonymous Flow Label Framework", draft-ietf-mpls-sfl-framework-11 (work in progress), October 2020.

Authors' Addresses

Stewart Bryant
Futurewei Technologies Inc.

Email: sb@stewartbryant.com

George Swallow
Southend Technical Center

Email: swallow.ietf@gmail.com

Siva Sivabalan
Ciena Corporation

Email: ssivabal@ciena.com

Network Working Group
Internet Draft
Intended status: Standards Track

Chandra Ramachandran (Ed)
Yakov Rekhter
Juniper Networks
Ina Minei
Google, Inc
Ebben Aries
Facebook
Dante Pacella
Verizon

Expires: September 06, 2015 March 06, 2015

Refresh Interval Independent FRR Facility Protection
draft-chandra-mpls-enhanced-frr-bypass-01

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on September 06, 2015.

Copyright Notice

Copyright (c) 2015 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.

Abstract

This document defines RSVP-TE extensions to facilitate refresh-interval independent FRR facility protection.

Conventions used in this document

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].

Table of Contents

1. Introduction

The facility backup protection mechanism is one of two methods discussed in [RFC4090] for enabling the fast reroute of traffic onto backup LSP tunnels in 10s of milliseconds, in the event of a failure. This document discusses a few shortcomings with some of the refresh-interval reliant procedures proposed for this method in [RFC4090]. These shortcomings come to the fore under scaled conditions and get highlighted even further when large RSVP-TE refresh intervals are used. The RSVP-TE extensions defined in this document will enhance the facility backup protection mechanism by making the corresponding procedures refresh-interval independent.

2. Motivation

Standard RSVP [RFC2205] maintains state via the generation of RSVP Path/Resv refresh messages. Refresh messages are used to both synchronize state between RSVP neighbors and to recover from lost RSVP messages. The use of Refresh messages to cover many possible failures has resulted in a number of operational problems. One problem relates to RSVP control plane scaling due to periodic refreshes of Path and Resv messages, another relates to the reliability and latency of RSVP signaling. An additional problem is the time to clean up the stale state after a tear message is lost. For more on these problems see Section 1 of [RFC2961]. All these problems adversely affect RSVP control plane scalability. RSVP-TE inherited all these problems from standard RSVP.

Procedures specified in [RFC2961] address the above mentioned problems by eliminating dependency on refreshes for state synchronization and for recovering from lost RSVP messages, and by eliminating dependency on refresh timeout for stale state cleanup. Implementing these procedures allows to improve RSVP-TE control plane scalability.

However, the procedures specified in [RFC2961] do not fully address stale state cleanup for facility backup protection [RFC4090], as facility backup protection still depends on refresh timeouts for stale state cleanup. Thus [RFC2961] is insufficient to address the problem of stale state cleanup when facility backup protection is used.

The procedures specified in this document, in combination with [RFC2961], eliminate facility backup protection dependency on refresh timeouts for stale state cleanup. These procedures, in combination with [RFC2961], fully address the above mentioned problem of RSVP-TE stale state cleanup, including the cleanup for facility backup protection.

The procedures specified in this document assume reliable delivery of RSVP messages, as specified in [RFC2961]. Therefore this document makes support for [RFC2961] a pre-requisite.

3. Problem Description

```
                    [E]
                   /   \
                  /     \
                 /       \
                /         \
               /           \
              /             \
             /               \
            /                 \
          [A]-----[B]-----[C]-----[D]
                   \             /
                    \           /
                     \         /
                      \       /
                       \     /
                        \   /
                         [F]
```

Figure 1: Example Topology

In the topology illustrated in Figure 1, consider a large number of LSPs from A to D transiting B and C. Assume that refresh interval has been configured to be large of the order of minutes and refresh reduction extensions are enabled on all routers.

Also assume that node protection has been configured for the LSPs and the LSPs are protected by each router in the following way

- A has made node protection available using bypass LSP A -> E -> C; A is the Point of Local Repair (PLR) and C is Node Protecting Merge Point (NP-MP)

- B has made node protection available using bypass LSP B -> F -> D; B is the PLR and D is the NP-MP

- C has made link protection available using bypass LSP C -> B -> F -> D; C is the PLR and D is the Link Protecting Merge Point (LP-MP)

In the above condition, assume that B-C link fails. The following is the sequence of events that is expected to occur for all protected LSPs under normal conditions.

1. B performs local repair and re-directs LSP traffic over the bypass LSP B -> F -> D.
2. B also creates backup state for the LSP and triggers sending of backup LSP state to D over the bypass LSP B -> F -> D.
3. D receives backup LSP states and merges the backups with the protected LSPs.
4. As the link on C over which the LSP states are refreshed has failed, C will no longer receive state refreshes. Consequently the protected LSP states on C will time out and C will send tear down message for all LSPs.

While the above sequence of events has been described in [RFC4090], there are a few problems for which no mechanism has been specified explicitly.

- If the protected LSP on C times out before D receives signaling for the backup LSP, then D would receive PathTear from C prior to receiving signaling for the backup LSP, thus resulting in deleting the LSP state. This would be possible at scale even with default refresh time.

- If upon the link failure C is to keep state until its timeout, then with long refresh interval this may result in a large amount of stale state on C. Alternatively, if upon the link failure C is to delete the state and send PathTear to D, this would result in deleting the state on D, thus deleting the LSP. D needs a reliable mechanism to determine whether it is MP or not to overcome this problem.

- If head-end A attempts to tear down LSP after step 1 but before step 2 of the above sequence, then B may receive the tear down message before step 2 and delete the LSP state from its state database. If B deletes its state without informing D, with long refresh interval this could cause (large) buildup of stale state on D.

- If B fails to perform local repair in step 1, then B will delete the LSP state from its state database without informing D. As B deletes its state without informing D, with long refresh interval this could cause (large) buildup of stale state on D.

The purpose of this document is to provide solutions to the above problems which will then make it practical to scale up to a large number of protected LSPs in the network.

4. Solution Aspects

The solution consists of five parts.

- Enhance the facility protection method defined in [RFC4090] by introducing an MP determination mechanism that enables PLR to signal availability of link or node protection to the MP. See section 4.1 for more details.

- Handle upstream link or node failures by cleaning up LSP states if the node has not found itself as MP through the MP determination mechanism. See section 4.2 for more details.

  The combination of "path state" maintained as Path State Block (PSB) and "reservation state" maintained as Reservation State Block (RSB) forms an individual LSP state on an RSVP-TE speaker.

- Introduce extensions to enable a router to send tear down message to downstream router that enables the receiving router to conditionally delete its local state. See section 4.3 for more details.

- Enhance facility protection by allowing a PLR to directly send tear down message to MP without requiring the PLR to either have a working bypass LSP or have already signaled backup LSP state. See section 4.4 for more details.

- Introduce extensions to enable the above procedures to be backward compatible with routers along the LSP path running implementation that do not support these procedures. See section 4.5 for more details.

4.1. Signaling Protection availability in Path RRO Flags

This section specifies a mechanism to allow the PLR to inform the MP if local protection is available. This mechanism relies on a combination of rules around the propagation of RRO flags carried in PATH messages (Section 4.1.2) and a targeted Node-ID Hello session (Section 4.1.3).

4.1.1. PLR Behavior

As per the procedures specified in RFC 4090, when a protected LSP comes up and if the "local protection desired" flag is set in the SESSION_ATTRIBUTE object, each node along the LSP path attempts to make local protection available for the LSP.

- If the "node protection desired" flag is set, then the node tries to become a PLR by attempting to create a NP-bypass LSP to the NNhop node avoiding the Nhop node on protected LSP path. In case node protection could not be made available after some time out, the node attempts to create a LP-bypass LSP to Nhop node avoiding only the link that protected LSP takes to reach Nhop

- If the "node protection desired" flag is not set, then the PLR attempts to create a LP-bypass LSP to Nhop node avoiding the link that the protected LSP takes to reach Nhop

With regard to the PLR procedures described above and that are specified in RFC 4090, this document specifies the following recommendations involving addresses selection, and additional PLR procedures involving RRO flags carried in PATH message as well as the initiation of Node-ID based Hello sessions.

- While selecting the destination address of the bypass LSP, the PLR SHOULD attempt to select the router ID of the NNhop or Nhop node. If the PLR and the MP are in same area, then the PLR may utilize the TED to determine the router ID from the interface

address in RRO (if NodeID is not included in RRO). If the PLR and the MP are in different IGP areas, then the PLR SHOULD use the NodeID address of NNhop MP if included in the RRO of RESV. If the NP-MP in a different area has not included NodeID in RRO, then the PLR SHOULD use NP-MP’s interface address present in the RRO. The PLR SHOULD use its router ID as the source address of the bypass LSP. The PLR SHOULD also include its router ID as the NodeID in PATH RRO unless configured explicitly not to include NodeID.

In parallel to the attempt made to create NP-bypass or LP-bypass, the PLR SHOULD initiate a Node-ID based Hello session to the NNhop or Nhop node respectively to establish the RSVP-TE signaling adjacency. This Hello session is used to track the state of the adjacency, including detection of adjacency failure.

- If the NP-bypass LSP comes up, then the PLR SHOULD set the "local protection available" and "NP available" RRO flags and triggers PATH to be sent.

- If the LP-bypass LSP comes up, then the PLR SHOULD set the "local protection available" RRO flag and triggers PATH to be sent.

- After signaling protection availability, if the PLR finds that the protection becomes unavailable then it SHOULD attempt to make protection available. The PLR SHOULD wait for a time out before resetting RRO flags relating to protection availability and triggering PATH downstream. On the other hand, the PLR need not wait for a time out to set RRO flags relating to protection availability and immediately trigger PATH downstream.

4.1.2. Remote Signaling Adjacency

A NodeID based RSVP-TE Hello session is one in which NodeID is used in source and destination address fields in RSVP Hello. [RFC4558] formalizes NodeID based Hello messages between two routers. This document extends NodeID based RSVP Hello session to track the state of RSVP-TE neighbor that is not directly connected by at least one interface. In order to apply NodeID based RSVP-TE Hello session between any two routers that are not immediate neighbors, the router that supports the extensions defined in the document SHOULD set TTL to 255 in the NodeID based Hello messages exchanged between PLR and MP.

In the rest of the document the term "signaling adjacency", or "remote signaling adjacency" refers specifically to the RSVP-TE signaling adjacency.

4.1.3. PATH RRO flags Propagation

As each node along the LSP path can make protection available, propagating PATH immediately due to change in RRO flags on any upstream node would increase control plane message load. So whenever a node receives PATH, it SHOULD check if the only change is in RRO flags. If the change is only in PATH RRO flags, then the node SHOULD decide whether to propagate the PATH based on the following rule.

- If "NP desired" flag is set and "NP available" flag has changed in Phop’s RRO flags, then PATH is triggered.

- In all other cases the change is not propagated.

4.1.4. MP Behavior

When the NNhop or Nhop node receives the triggered PATH with RRO flag(s) set, the node SHOULD check the presence of remote signaling adjacency with PLR (this check is needed to detect network being partitioned). If the flags are set and the RSVP-TE signaling adjacency is present, the node concludes that protection has been made available at the PLR. If the PLR has included NodeID in PATH RRO, then that NodeID is the remote neighbor address. Otherwise, the PLR’s interface address in RRO will be the remote neighbor address. If the "NP available" flag is set by PPhop node, then it is NP-MP. Otherwise, it concludes it is LP-MP.

4.1.5. "Remote" state on MP

Once a router concludes it is MP, it SHOULD create a remote path state for the LSP. The "remote" state is identical to the protected LSP path state except for the difference in HOP object. The HOP object corresponding to the "remote" path state contains the address of remote node signaling adjacency with PLR.

The MP SHOULD consider the "remote" path state automatically deleted if:

- NP-MP later receives a PATH with "NP available" flag reset in PLR’s RRO flags, or

- LP-MP later receives PATH with "local protection available" flag reset in PLR’s RRO flags, or

- Node signaling adjacency with PLR goes down, or

- MP receives backup LSP signaling from PLR or

- MP receives PathTear, or

- MP deletes the LSP state on local policy or exception event

Unlike the normal path state that is either locally generated on Ingress or created from PATH message from Phop node, the "remote" path state is not signaled explicitly form PLR. The purpose of "remote" path state is to enable the PLR to explicitly tear down path and reservation states corresponding to the LSP by sending tear message for the "remote" path state. Such message tearing down "remote" path state is called "Remote PathTear.

The scenarios in which "Remote" PathTear is applied are described in Section 4.4 - Remote State Teardown.

4.2. Impact of Failures on LSP State

This section describes the procedures for routers on the LSP path for different kinds of failures. The procedures described on detecting RSVP control plane adjacency failures do not impact the RSVP-TE graceful restart mechanisms ([RFC3473], [RFC5063]). If the router executing these procedures act as helper for neighboring router, then the control plane adjacency will be declared as having failed after taking into account the grace period extended for neighbor by the helper.

It should be noted that even though this section and the subsequent sections of the document mention "link failure" and "node failure" separately involving upstream or downstream of a protected LSP, a router implementing the procedures specified in the document need not have a mechanism to distinguish between these two types of failures. Optionally, a router MAY run Node-ID based RSVP-TE signaling adjacency with immediate neighbors to distinguish between these two types of failures.

4.2.1. Non-MP Behavior on Phop Link/Node Failure

When a router detects Phop link or Phop node failure and the router is not an MP for the LSP, then it SHOULD send Conditional PathTear (refer to Section "Conditional PathTear" below) and delete PSB and RSB states corresponding to the LSP.

4.2.2. LP-MP Behavior on Phop Link Failure

When the Phop link for an LSP fails on a router that is LP-MP for the LSP, the LP-MP SHOULD retain PSB and RSB states corresponding to the LSP till the occurrence of any of the following events.

- Node-ID signaling adjacency with Phop PLR goes down, or

- MP receives normal or "Remote" PathTear for PSB, or

- MP receives ResvTear RSB.

4.2.3. LP-MP Behavior on Phop Node Failure

When a router that is LP-MP for an LSP detects Phop node failure from Node-ID signaling adjacency state, the LP-MP SHOULD send normal PathTear and delete PSB and RSB states corresponding to the LSP.

4.2.4. NP-MP Behavior on Phop Link/Node Failure

When a router that is NP-MP for an LSP detects Phop link failure, or Phop node failure from Node-ID signaling adjacency, the router SHOULD retain PSB and RSB states corresponding to the LSP till the occurrence of any of the following events.

- Remote Node-ID signaling adjacency with PPhop PLR goes down, or

- MP receives normal or "Remote" PathTear for PSB, or

- MP receives ResvTear for RSB.

4.2.5. NP-MP Behavior on PLR Link Failure

If the PLR link that is not attached to NP-MP fails and if NP-MP receives Conditional PathTear from the Phop node, then the MP SHOULD retain PSB and RSB states corresponding to the LSP till the occurrence of any of the following events.

- Remote Node-ID signaling adjacency with PPhop PLR goes down, or

- MP receives normal or "Remote" PathTear for PSB, or

- MP receives ResvTear for RSB.

Receiving Conditional PathTear from the Phop node will not impact the "remote" state from the PLR. Note that Phop node would send Conditional PathTear if it was not an MP.

In the example topology in Figure 1, assume C & D are NP-MP for PLRs A & B respectively. Now when A-B link fails, as B is not MP and its Phop link signaling adjacency has failed, B will delete LSP state (this behavior is required for unprotected LSPs - Section 4.2.1). In the data plane, that would require B to delete the label forwarding entry corresponding to the LSP. So if B’s downstream nodes C and D continue to retain state, it would not be correct for D to continue to assume itself as NP-MP for PLR B.

The mechanism that enables D to stop considering itself as NP-MP and delete "remote" path state is given below.

1. When C receives Conditional PathTear from B, it decides to retain LSP state as it is NP-MP of PLR A. C also SHOULD check whether Phop B had previously signaled availability of node protection. As B had previously signaled NP availability in its PATH RRO flags, C SHOULD reset "local protection available" and "NP available" on RRO flags corresponding to B and trigger PATH to D.
2. When D receives triggered PATH, it realizes that it is no longer NP-MP and so deletes the "remote" path state. D does not propagate PATH further down because the only change is in PATH RRO flags of B.

4.2.6. Phop Link Failure on a Node that is LP-MP and NP-MP

A router may be both LP-MP as well as NP-MP at the same time for Phop and PPhop nodes respectively of an LSP. If Phop link fails on such node, the node SHOULD retain PSB and RSB states corresponding to the LSP till the occurrence of any of the following events.

- Both Node-ID signaling adjacencies with Phop and PPhop nodes go down, or

- MP receives normal or "Remote" PathTear for PSB, or

- MP receives ResvTear for RSB.

4.2.7. Phop Node Failure on Node that is LP-MP and NP-MP

If a router that is both LP-MP and NP-MP detects Phop node failure, then the node SHOULD retain PSB and RSB states corresponding to the LSP till the occurrence of any of the following events.

- Remote Node-ID signaling adjacency with PPhop PLR goes down, or

- MP receives normal or "Remote" PathTear for PSB, or

- MP receives ResvTear for RSB.

4.3. Conditional Path Tear

In the example provided in the Section 4.2.5 "NP-MP Behavior on PLR link failure", B deletes PSB and RSB states corresponding to the LSP once B detects its link to Phop went down as B is not MP. If B were to send PathTear normally, then C would delete LSP state immediately. In order to avoid this, there should be some mechanism by which B can indicate to C that B does not require the receiving node to unconditionally delete the LSP state immediately. For this, B SHOULD add a new optional object called CONDITIONS object in PathTear. The new optional object is defined in Section 4.3.3. If node C also understands the new object, then C SHOULD delete LSP state only if it is not an NP-MP - in other words C SHOULD delete LSP state if there is no "remote" PLR state on C.

4.3.1. Sending Conditional Path Tear

A router that is not an MP for an LSP SHOULD delete PSB and RSB states corresponding to the LSP if Phop link or Phop Node-ID signaling adjacency goes down (Section 4.2.1). The router SHOULD send Conditional PathTear if the following are also true.

- Ingress has requested node protection for the LSP, and

- PathTear is not received from upstream node

4.3.2. Processing Conditional Path Tear

When a router that is not an NP-MP receives Conditional PathTear, the node SHOULD delete PSB and RSB states corresponding to the LSP, and process Conditional PathTear by considering it as normal PathTear. Specifically, the node SHOULD NOT propagate Conditional PathTear downstream but remove the optional object and send normal PathTear downstream.

When a node that is an NP-MP receives Conditional PathTear, it SHOULD NOT delete LSP state. The node SHOULD check whether the Phop node previously set "NP available" flag in PATH RRO flags. If the flag had been set previously by Phop, then the node SHOULD clear "local protection available" and "NP available" flags in Phop’s RRO flags and trigger PATH downstream.

If Conditional PathTear is received from a neighbor that has not advertised support (refer to Section 4.5) for the new procedures defined in this document, then the node SHOULD consider the message as normal PathTear. The node SHOULD propagate normal PathTear downstream and delete LSP state.

4.3.3. CONDITIONS object

As any implementation that does not support Conditional PathTear SHOULD ignore the new object but process the message as normal PathTear without generating any error, the Class-Num of the new object SHOULD be 10bbbbbb where ’b’ represents a bit (from Section 3.10 of [RFC2205]).

The new object is called as "CONDITIONS" object that will specify the conditions under which default processing rules of the RSVP-TE message SHOULD be invoked.

The object has the following format:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length               |  Class        |     C-type    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Reserved                            |M|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length

This contains the size of the object in bytes and should be set to eight.

Class

TBD

C-type

1

M bit

This bit indicates that the message SHOULD be processed based on the condition whether the receiving node is Merge Point or not.

4.4. Remote State Teardown

If the Ingress wants to tear down the LSP because of a management event while the LSP is being locally repaired at a transit PLR, it would not be desirable to wait till backup LSP signaling to perform state cleanup. To enable LSP state cleanup when the LSP is being locally repaired, the PLR SHOULD send "remote" PathTear message instructing the MP to delete PSB and RSB states corresponding to the LSP.

Consider node C in example topology (Figure 1) has gone down and B locally repairs the LSP.

1. Ingress A receives a management event to tear down the LSP.
2. A sends normal PathTear to B.
3. To enable LSP state cleanup, B SHOULD send "remote" PathTear with destination IP address set to that of D used in Node-ID signaling adjacency with D, and HOP object containing local address used in Node-ID signaling adjacency.
4. B then deletes PSB and RSB states corresponding to the LSP.
5. On D there would be a remote signaling adjacency with B and so D SHOULD accept the remote PathTear and delete PSB and RSB states corresponding to the LSP.

4.4.1. PLR Behavior on Local Repair Failure

If local repair fails on the PLR after a failure, then this should be considered as a case for cleaning up LSP state from PLR to the Egress. PLR would achieve this using "remote" PathTear to clean up state from MP. If MP has retained state, then it would propagate PathTear downstream thereby achieving state cleanup. Note that in the case of link protection, the PathTear would be directed to LP-MP node IP address rather than the Nhop interface address.

4.4.2. PLR Behavior on Resv RRO Change

When a router that has already made NP available detects a change in the RRO carried in RESV message, and if the RRO change indicates

that the router’s former NP-MP is no longer present in the LSP path, then the router SHOULD send "Remote" PathTear directly to its former NP-MP.

In the example topology in Figure 1, assume A has made node protection available and C has concluded it is NP-MP. When the B-C link fails then implementing the procedure specified in Section 4.2.4 of this document, C will retain state till: remote NodeID control plane adjacency with A goes down, or PathTear or ResvTear is received for PSB or RSB respectively. If B also has made node protection available, B will eventually complete backup LSP signaling with its NP-MP D and trigger RESV to A with RRO changed. The new RRO of the LSP carried in RESV will not contain C. When A processes the RESV with a new RRO not containing C - its former NP-MP, A SHOULD send "Remote" PathTear to C. When C receives a "Remote" PathTear for its PSB state, C will send normal PathTear downstream to D and delete both PSB and RSB states corresponding to the LSP. As D has already received backup LSP signaling from B, D will retain control plane and forwarding states corresponding to the LSP.

4.4.3. LSP Preemption during Local Repair

If an LSP is preempted when there is no failure along the path of the LSP, the node on which preemption occurs would send PathErr and ResvTear upstream and only delete the forwarding state and RSB state corresponding to the LSP. But if the LSP is being locally repaired upstream of the node on which the LSP is preempted, then the node SHOULD delete both PSB and RSB states corresponding to the LSP and send normal PathTear downstream.

4.4.3.1. Preemption on LP-MP after Phop Link failure

If an LSP is preempted on LP-MP after its Phop or incoming link has already failed but the backup LSP has not been signaled yet, then the node SHOULD send normal PathTear and delete both PSB and RSB states corresponding to the LSP. As the LP-MP has retained LSP state because the PLR would signal the LSP through backup LSP signaling, preemption would bring down the LSP and the node would not be LP-MP any more requiring the node to clean up LSP state.

4.4.3.2. Preemption on NP-MP after Phop Link failure

If an LSP is preempted on NP-MP after its Phop link has already failed but the backup LSP has not been signaled yet, then the node SHOULD send normal PathTear and delete PSB and RSB states corresponding to the LSP. As the NP-MP has retained LSP state

because the PLR would signal the LSP through backup LSP signaling, preemption would bring down the LSP and the node would not be NP-MP any more requiring the node to clean up LSP state.

Consider B-C link goes down on the same example topology (Figure 1). As C is NP-MP for PLR A, C will retain LSP state.

1. The LSP is preempted on C.
2. C will delete RSB state corresponding to the LSP. But C cannot send PathErr or ResvTear to PLR A because backup LSP has not been signaled yet.
3. As the only reason for C having retained state after Phop node failure was that it was NP-MP, C SHOULD send normal PathTear to D and delete PSB state also. D would also delete PSB and RSB states on receiving PathTear from C.
4. B starts backup LSP signaling to D. But as D does not have the LSP state, it will reject backup LSP PATH and send PathErr to B.
5. B will delete its reservation and send ResvTear to A.

4.5. Backward Compatibility Procedures

The "Enhanced FRR facility protection" referred below in this section refers to the set of changes that have been proposed in previous sections. Any implementation that does not support them has been termed as "existing implementation". Of the proposed extensions, signaling protection using RRO flags is expected to be backward compatible and can work safely irrespective of whether the refresh time is small or arbitrarily long. This is because the existing implementations would not send error or tear down message in response to the flags in PATH RRO but would simply ignore and propagate them. On the other hand, changes proposed relating to LSP state cleanup namely Conditional and remote PathTear require support from other nodes along the LSP path. So procedures that fall under LSP state cleanup category SHOULD be turned on only if all nodes involved in the node protection FRR i.e. PLR, MP and intermediate node in the case of NP, support the extensions. Note that for LSPs requesting only link protection, the PLR and the LP-MP should support the extensions.

4.5.1. Detecting Support for Enhanced FRR Facility Protection

An implementation supporting the FRR facility protection extensions specified in previous sections SHOULD set a new flag "Enhanced facility protection" in CAPABILITY object in Hello messages.

- As nodes supporting the extensions SHOULD initiate Node Hellos with adjacent nodes, a node on the path of protected LSP can determine whether its Phop or Nhop neighbor supports FRR enhancements from the Hello messages sent by the neighbor.

- If a node attempts to make node protection available, then the PLR SHOULD initiate remote Node-ID signaling adjacency with NNhop. If the NNhop (a) does not reply to remote node Hello message or (b) does not set "Enhanced facility protection" flag in CAPABILITY object in the reply, then the PLR can conclude that NNhop does not support FRR extensions.

- If node protection is requested for an LSP and if (a) PPhop node has not set "local protection available" and "NP available" flags in its RRO flags or (b) PPhop node has not initiated remote node Hello messages, then the node SHOULD conclude that PLR does not support FRR extensions. The details are described in the "Procedures for backward compatibility" section below.

The new flag that will be introduced to CAPABILITY object is specified below.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             | Class-Num(134)|  C-Type  (1)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Reserved                      |E|T|R|S|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

E bit

Indicates that the sender supports Enhanced FRR facility protection

Any node that sets the new E-bit is set in its CAPABILITY object MUST also set Refresh-Reduction-Capable bit in common header of all RSVP-TE messages.

4.5.2. Procedures for backward compatibility

The procedures defined hereafter are performed on a subset of LSPs that traverse a node, rather than on all LSPs that traverse a node. This behavior is required to support backward compatibility for a subset of LSPs traversing nodes running existing implementations.

4.5.2.1. Lack of support on Downstream Node

- If the Nhop does not support enhanced facility protection FRR, then the node SHOULD reduce the "refresh period" in TIME_VALUES object carried in PATH to default small refresh default value.

- If node protection is requested and the NNhop node does not support the enhancements, then the node SHOULD reduce the "refresh period" in TIME_VALUES object carried in PATH to a small refresh default value.

If the node reduces the refresh time from the above procedures, it SHOULD also not send remote PathTear or Conditional PathTear messages.

Consider the example topology in Figure 1. If C does not support scalability improvements, then:

- A and B SHOULD reduce the refresh time to default value of 30 seconds and trigger PATH

- If B is not an MP and if Phop link of B fails, B cannot send Conditional PathTear to C but SHOULD time out PSB state from A normally. This would be accomplished if A would also reduce the refresh time to default value. So if C does not support enhanced facility protection, then Phop B and PPhop A SHOULD reduce refresh time to a small default value.

4.5.2.2. Lack of support on Upstream Node

- If Phop node does not support enhanced facility protection, then the node SHOULD reduce the "refresh period" in TIME_VALUES object carried in RESV to default small refresh time value.

- If node protection is requested and the Phop node does not support the enhancements, then the node SHOULD reduce the "refresh period" in TIME_VALUES object carried in PATH to default value.

- If node protection is requested and PPhop node does not support the enhancements, then the node SHOULD reduce the "refresh period" in TIME_VALUES object carried in RESV to default value.

- If the node reduces the refresh time from the above procedures, it SHOULD also not execute MP determination procedures.

4.5.2.3. Incremental Deployment

The backward compatibility procedures described in the previous sub-sections imply that a router supporting the FRR extensions specified in this document can apply the procedures specified in the document either in the downstream or upstream direction of an LSP, depending on the capability of the routers downstream or upstream in the LSP path.

- FRR extensions and procedures are enabled for downstream Path, PathTear and ResvErr messages corresponding to an LSP if link protection is requested for the LSP and the Nhop node supports the extensions

- FRR extensions and procedures are enabled for downstream Path, PathTear and ResvErr messages corresponding to an LSP if node protection is requested for the LSP and both Nhop & NNhop nodes support the extensions

- FRR extensions and procedures are enabled for upstream PathErr, Resv and ResvTear messages corresponding to an LSP if link protection is requested for the LSP and the Phop node supports the extensions

- FRR extensions and procedures are enabled for upstream PathErr, Resv and ResvTear messages corresponding to an LSP if node protection is requested for the LSP and both Phop and PPhop nodes support the extensions

For example, if implementation supporting the FRR extensions specified in this document is deployed on all routers in particular region of the network and if all the LSPs in the network request node protection, then the FRR extensions will only be applied for the LSP segments that traverse the particular region. This will aid incremental deployment of these extensions and also allow reaping the benefits of the extensions in portions of the network where it is supported.

5. Security Considerations

This document extends the applicability of Node-ID based Hello session between immediate neighbors. The Node-ID based Hello session between PLR and NP-MP may require the two routers to exchange Hello messages with non-immediate neighbor. So, the implementations SHOULD provide the option to configure Node-ID neighbor specific or global authentication key to authentication messages received from Node-ID neighbors. The network administrator MAY utilize this option to enable RSVP-TE routers to authenticate Node-ID Hello messages received with TTL greater than 1.

6. IANA Considerations

6.1. New Object - CONDITIONS

[RFC2205] defines the Class-Number name space for RSVP objects. The name space is managed by IANA.

IANA registry: RSVP Parameters
Subsection: Class Names, Class Numbers, and Class Types

A new RSVP object using a Class-Number of form 10bbbbbb called the "CONDITIONS" object is defined in Section 4.3 of this document. The Class-Number is TBD.

6.2. New CAPABILITY Object value

[RFC5063] defines the name space for RSVP Capability Object Values. The name space is managed by IANA.

IANA registry: RSVP PARAMETERS
Subsection: Capability Object Values

A new Capability flag called "Enhanced FRR facility protection" is defined in Section 4.5 of this document. The bit number for this flag is TBD.

7. Normative References

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

[RFC4090] Pan, P., "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, May 2005.

[RFC2961] Berger, L., "RSVP Refresh Overhead Reduction Extensions", RFC 2961, April 2001.

[RFC3209] Awduche, D., "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC2205] Braden, R., "Resource Reservation Protocol (RSVP)", RFC 2205, September 1997.

[RFC4558] Ali, Z., "Node-ID Based Resource Reservation (RSVP) Hello: A Clarification Statement", RFC 4558, June 2006.

8. Acknowledgments

Thanks to Raveendra Torvi and Yimin Shen for their comments and inputs.

9. Authors' Addresses

Chandra Ramachandran
Juniper Networks
Email: csekar@juniper.net

Yakov Rekhter
Juniper Networks
Email: yakov@juniper.net

Ina Minei
Google, Inc
inaminei@google.com

Ebben Aries
Facebook
Email: exa@fb.com

Dante Pacella
Verizon
Email: dante.j.pacella@verizon.com

Markus Jork
Juniper Networks
Email: mjork@juniper.net

Harish Sitaraman

Juniper Networks
Email: hsitaraman@juniper.net

Vishnu Pavan Beeram
Juniper Networks
Email: vbeeram@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 12, 2016

W. Cheng
L. Wang
H. Li
China Mobile
H. Helvoort
Hai Gaoming BV
K. Liu
J. Dong
J. He
Huawei Technologies
F. Li
China Academy of Telecommunication Research, MIIT., China
J. Yang
ZTE Corporation P.R.China
J. Wang
Fiberhome Telecommunication Technologies Co., LTD.
August 11, 2015

MPLS-TP Shared-Ring protection (MSRP) mechanism for ring topology draft-cheng-mpls-tp-shared-ring-protection-06

Abstract

This document describes requirements, architecture and solutions for MPLS-TP Shared Ring Protection (MSRP) in the ring topology for point-to-point (P2P) services. The mechanism of MSRP is illustrated and how it satisfies the requirements for optimized ring protection in RFC 5654 is analyzed. This document also defines the Ring Protection Switch (RPS) Protocol which is used to coordinate the protection behavior of the nodes on MPLS ring.

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 February 12, 2016.

Copyright Notice

Copyright (c) 2015 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

As described in 2.5.6.1 of [RFC5654], Ring Protection of MPLS-TP requirements , several service providers have expressed much interest in operating MPLS-TP in ring topologies and require a high-level survivability function in these topologies. In operational transport network deployment, MPLS-TP networks are often constructed with ring topologies. It calls for an efficient and optimized ring protection mechanism to achieve simple operation and fast, sub 50 ms, recovery performance.

The requirements for MPLS-TP [RFC5654] state that recovery mechanisms which are optimized for ring topologies could be further developed if it can provide the following features:

a. Minimize the number of OAM entities for protection

b. Minimize the number of elements of recovery

c. Minimize the required label number

d. Minimize the amount of control and management-plane transactions during maintenance operation

e. Minimize the impact on information exchange during protection if a control plane is supported

This document specifies MPLS-TP Shared-Ring Protection mechanisms that can meet all those requirements on ring protection listed in [RFC5654].

The basic concepts and architecture of Shared-Ring protection mechanism are specified in this document. This document focuses on the solutions for point-to-point transport paths. While the basic concepts may also apply to point-to-multipoint transport paths, the solution for point-to-multipoint transport paths is under study and will be presented in a separate document.

2. Requirements for MPLS-TP Ring Protection

The requirements for MPLS-TP ring protection are specified in [RFC5654]. This document elaborates on the requirements in detail.

2.1. Recovery of Multiple Failures

MPLS-TP is expected to be used in carrier grade metro networks and backbone transport networks to provide mobile backhaul, business services etc., in which the network survivability is very important. According to R106 B in [RFC5654], MPLS-TP recovery mechanisms in a ring SHOULD protect against multiple failures. The following text provides some more detailed illustration about "multiple failures". In metro and backbone networks, a single risk factor often affects multiple links or nodes. Some examples of risk factors are given as follows:

- multiple links use fibers in one cable or pipeline

- Several nodes share one power supply system

- Weather sensitive micro-wave system

Once one of the above risk factors happens, multiple links or nodes failures may occur simultaneously and those failed links or nodes may be located on a single ring as well as on interconnected rings. Ring protection against multiple failures should cover both multiple failures on a single ring and multiple failures on interconnected rings, as long as the connectivity between the ingress and egress node of the ring still exists.

2.2. Smooth Upgrade from Linear Protection to Ring Protection

It is beneficial for service providers to upgrade the protection scheme from linear protection to ring protection in their MPLS-TP network without service interruption. In-service insertion and removal of a node on the ring should also be supported. Therefore,

the MPLS-TP ring protection mechanism is supposed to be developed and optimized for compliance with this smooth upgrading principle.

2.3. Configuration Complexity

Ring protection can reduce the dependency of configuration on the quantity of services, thus will simplify the network protection configuration and operation effort. This is because the ring protection makes use of the characteristics of ring topology and mechanisms on the section layer. While in the application scenarios of deploying linear protection in ring topology MPLS-TP network, the configuration of protection has a close relationship with the quantities of services carried. Especially in some large metro networks with more than ten thousands of services in the access nodes, the LSP linear protection capabilities of the metro core nodes needs to be large enough to meet the network planning requirements, which also leads to the complexity of network protection configurations and operations.

3. Terminology and Notation

The following syntax will be used to describe the contents of the label stack:

1. The label stack will be enclosed in square brackets ("[]").

2. Each level in the stack will be separated by the '|' character. It should be noted that the label stack may contain additional layers. However, we only present the layers that are related to the protection mechanism.

3. If the Label is assigned by Node X, the Node Name is enclosed in bracket ("()")

4. Shared Ring Protection Architecture

4.1. Ring Tunnel

This document introduces a new logical layer of the ring for shared ring protection in MPLS-TP networks. As shown in Figure 1, the new logical layer consists of ring tunnels which provides a server layer for the LSPs traverse the ring. Once a ring tunnel is established, the configuration, management and protection of the ring are all performed at the ring tunnel level. One port can carry multiple ring tunnels, while one ring tunnel can carry multiple LSPs.

```
                                                  +-------------
                                     +-------------|
                        +-------------|             |
                =====PW1======|             |             |
                              |             |   Ring      |  Physical
                =====PW2======|    LSP      |  Tunnel     |  Port
                              |             |             |
                =====PW3======|             |             |
                        +-------------|             |
                                     +-------------|
                                                  +-------------
```

Figure 1. The logical layers of the ring

The label stack used in MPLS-TP Shared Ring Protection mechanism is shown as below:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Ring tunnel Label           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                LSP Label              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PW Label              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Payload               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2. Label stack used in MPLS-TP Shared Ring Protection

4.1.1. Establishment of Ring Tunnel

The Ring tunnels are established based on the egress node. The egress node is the node where traffic leaves the ring. LSPs which have the same egress node on the ring share the same ring tunnels. In other words, all the LSPs that traverse the ring and exit from the same node share the same working ring tunnel and protection ring tunnel. For each egress node, four ring tunnels are established:

- o one clockwise working ring tunnel, which is protected by the anticlockwise protection ring tunnel

- o one anticlockwise protection ring tunnel

- o one anticlockwise working ring tunnel, which is protected by the clockwise protection ring tunnel

- o one clockwise protection ring tunnel

The structure of the protection tunnels are determined by the selected protection mechanism. This will be detailed in subsequent sections.

As shown in Figure 3, LSP1, LSP2 and LSP3 enter the ring from Node E, Node A and Node B, respectively, and all leave the ring at Node D. To protect these LSPs that traverse the ring, a clockwise working ring tunnel (RcW_D) via E->F->A->B->C->D, and its anticlockwise protection ring tunnel (RaP_D) via D->C->B->A->F->E->D are established, Also, an anti-clockwise working ring tunnel (RaW_D) via C->B->A->F->E->D, and its clockwise protection ring tunnel (RcP_D) via D->E->F->A->B->C->D are established. For simplicity Figure 3 only shows RcW_D and RaP_D. A similar provisioning should be applied for any other node on the ring. In summary, for each node in Figure 3 when acting as egress node, the ring tunnels are created as follows:

- To Node A: RcW_A, RaW_A, RcP_A, RaP_A

- To Node B: RcW_B, RaW_B, RcP_B, RaP_B

- To Node C: RcW_C, RaW_C, RcP_C, RaP_C

- To Node D: RcW_D, RaW_D, RcP_D, RaP_D

- To Node E: RcW_E, RaW_E, RcP_E, RaP_E

- To Node F: RcW_F, RaW_F, RcP_F, RaP_F

```
                      +---+#############+---+
                      | F |-------------| A | +-- LSP2
                      +---+*************+---+
                      #/*                  *\#
                     #/*                    *\#
                    #/*                      *\#
                  +---+                    +---+
          LSP1-+ | E |                    | B |+-- LSP3
                  +---+                    +---+
                    #\                      */#
                     #\                    */#
                      #\                  */#
                      +---+*************+---+
           LSP1   +--| D |-------------| C |
           LSP2      +---+#############+---+
           LSP3

                     ---- physical links
                     **** RcW_D
                     #### RaP_D
```

Figure 3. Ring tunnels in MSRP

Through these working and protection ring tunnels, LSPs which enter the ring from any node can reach any egress nodes on the ring, and are protected from failures on the ring.

4.1.2. Label Assignment and Distribution

The ring tunnel labels are downstream-assigned labels as defined in [RFC3031]. The ring tunnel labels can be either configured statically, provisioned by a controller, or distributed dynamically via a control protocol.

4.1.3. Forwarding Operation

When an MPLS-TP transport path, such as an LSP, enters the ring, the ingress node on the ring pushes the working ring tunnel label according to the egress node and sends the traffic to the next hop. The transit nodes on the working ring tunnel swap the ring tunnel labels and forward the packets to the next hop. When the packet arrives at the egress node, the egress node pops the ring tunnel label and forwards the packets based on the inner LSP label and PW label. Figure 4 shows the label operation in the MPLS-TP shared ring protection mechanism. Assume that LSP1 enters the ring at Node A and exits from Node D, and the following label operations are executed.

1. Ingress node: Packets of LSP1 arrive at Node A with a label stack [LSP1] and is supposed to be forwarded in the clockwise direction of the ring. The clockwise working ring tunnel label RcW_D will be pushed at Node A, the label stack for the forwarded packet at Node A is changed to [RcW_D(B)|LSP1].

2. Transit nodes: In this case, Node B and Node C forward the packets by swapping the working ring tunnel labels. For example, the label [RcW_D(B)|LSP1] is swapped to [RcW_D(C)|LSP1] at Node B.

3. Egress node: When the packet arrives at Node D (i.e. the egress node) with label stack [RcW_D(D)|LSP1], Node D pops RcW_D(D), and subsequently deals with the inner labels of LSP1.

```
                +---+#####[RaP_D(F)]######+---+
                | F |---------------------| A | +-- LSP1
                +---+*****[RcW_D(A)]******+---+
                 #/*                        *\#
          [RaP_D(E)]#/*[RcW_D(F)]      [RcW_D(B)]*\#[RaP_D(A)]
                  #/*                          *\#
                +---+                        +---+
                | E |                        | B |
                +---+                        +---+
                  #\                          */#
          [RaP_D(D)]#\                [RxW_D(C)]*/#[RaP_D(B)]
                     #\                      */#
                     +---+*****[RcW_D(D)]****+---+
           LSP1  +-- | D |-------------------| C |
                     +---+#####[RaP_D(C)]####+---+

           -----physical links      ****** RcW_D    ###### RaP_D
```

Figure 4. Label operation of MSRP

4.2. Failure Detection

The MPLS-TP section layer OAM is used to monitor the connectivity between each two adjacent nodes on the ring using the mechanisms defined in [RFC6371]. Protection switching is triggered by the failure detected on the ring by the OAM mechanisms.

Two end ports of a link form a Maintenance Entity Group (MEG), and an MEG end point (MEP) function is installed in each ring port. CC OAM packets are periodically exchanged between each pair of MEPs to monitor the link health. Three consecutive CC packets losses will be interpreted as a link failure.

A node failure is regarded as the failure of two links attached to that node. The two nodes adjacent to the failed node detect the failure in the links that are connected to the failed node.

4.3. Ring Protection

This section specifies the ring protection mechanisms in detail. In general, the description uses the clockwise working ring tunnel and the corresponding anti-clockwise protection ring tunnel as example, but the mechanism is applicable in the same way to the anti-clockwise working and clockwise protection ring tunnels.

Taking the topology in Figure 4 as example, the LSP1 enters the ring at Node A and leaves the ring at Node D. In normal state, LSP1 is carried by clockwise working ring tunnel (RcW_D) through the path A->B->C->D, the label operation is:

[LSP1](original data traffic carried by LSP1) ->
[RCW_D(B)|LSP1](NodeA) -> [RCW_D(C)|LSP1](NodeB) -> [RCW_D(D)|
LSP1](NodeC) -> [LSP1](data traffic carried by LSP1). Then at node D the packet will be forwarded based on label stack of LSP1.

The following sections describes the protection mechanisms used in ring topology.

4.3.1. Wrapping

With the wrapping mechanism, the protection ring tunnel is a closed ring identified by the egress node. As shown in Figure 4, the RaP_D is the anticlockwise protection ring tunnel for the clockwise working ring tunnel RcW_D. As specified in the following sections, the closed ring protection tunnel can protect both the link failure and the node failure.

4.3.1.1. Wrapping for Link Failure

When a link failure between Node B and Node C occurs, if it is a bi-directional failure, both Node B and Node C can detect the failure via OAM mechanism; if it is a uni-directional failure, one of the two nodes would detect the failure and it would inform the other node via the Ring Protection Switch Protocol (RPS) which is specified in section 5. Then Node B switches the clockwise working ring tunnel (RcW_D) to the anticlockwise protection ring tunnel (RaP_D) and Node C switches anticlockwise protection ring tunnel(RaP_D) to the clockwise working ring tunnel(RcW_D). The data traffic which enters the ring at Node A and leaves the ring at Node D follows the path A->B->A->F->E->D->C->D. The label operation is:

[LSP1](Original data traffic) -> [RcW_D(B)|LSP1](Node A) -> [RaP_D(A)|LSP1](Node B) -> [RaP_D(F)|LSP1](Node A) -> [RaP_D(E)|LSP1] (Node F) -> [RaP_D(D)|LSP1] (Node E) -> [RaP_D(C)|LSP1] (Node D) -> [RcW_D(D)|LSP1](Node C) -> [LSP1](data traffic leaves the ring).

```
                      +---+#####[RaP_D(F)]######+---+
                      | F |---------------------| A | +-- LSP1
                      +---+*****[RcW_D(A)]******+---+
                      #/*                        *\#
            [RaP_D(E)]#/*[RcW_D(F)]      [RcW_D(B)]*\#RaP_D(A)
                     #/*                          *\#
                    +---+                        +---+
                    | E |                        | B |
                    +---+                        +---+
                     #\                           *x#
            [RaP_D(D)]#\                [RcW_D(C)]*x#RaP_D(B)
                      #\                         *x#
                      +---+*****[RcW_D(D)]****+---+
             LSP1  +-- | D |-------------------| C |
                      +---+#####[RaP_D(C)]####+---+

               -----physical links    xxxx Failure Link
               ****** RcW_D           ###### RaP_D
```

Figure 5.Wrapping for link failure

4.3.1.2. Wrapping for Node Failure

As shown in Figure 6, when Node B fails, Node A detects the failure between A and B and switches the clockwise work ring tunnel (RcW_D) to the anticlockwise protection ring tunnel (RaP_D), Node C detects the failure between C and B and switches the anticlockwise protection ring tunnel (RaP_D) to the clockwise working ring tunnel (RcW_D). The data traffic which enters the ring at Node A and exits at Node D follows the path A->F->E->D->C->D. The label operation is:

[LSP1](original data traffic carried by LSP1) -> [RaP_D(F)|LSP1](NodeA) -> [RaP_D(E)|LSP1](NodeF) -> [RaP_D(D)|LSP1](NodeE) -> [RaP_D(C)|LSP1] (NodeD) -> [RcW_D(D)|LSP1] (NodeC) -> [LSP1](data traffic carried by LSP1).

In one special case where node D fails, all the ring tunnels with node D as egress will become unusable. However, before the failure location is propagated to all the ring nodes, the wrapping protection mechanism may cause temporary traffic loop: node C detects the failure and switches the traffic from the clockwise work ring tunnel (RcW_D) to the anticlockwise protection ring tunnel (RaP_D), node E also detects the failure and would switch the traffic from

anticlockwise protection ring tunnel (RaP_D) back to the clockwise work ring tunnel (RcW_D). A possible mechanism to mitigate the temporary loop problem is: the TTL of the ring tunnel label is set to 2*N by the ingress ring node of the traffic, where N is the number of nodes on the ring.

```
                        +---+#####[RaP_D(F)]######+---+
                        | F |---------------------| A | +-- LSP1
                        +---+*****[RcW_D(A)]******+---+
                        #/*                        *\#
             [RaP_D(E)]#/*[RcW_D(F)]      [RcW_D(B)]*\#RaP_D(A)
                      #/*                            *\#
                     +---+                          xxxxx
                     | E |                          x B x
                     +---+                          xxxxx
                       #\                            */#
             [RaP_D(D)]#\                [RcW_D(C)]*/#RaP_D(B)
                        #\                        */#
                        +---+*****[RcW_D(D)]****+---+
              LSP1  +-- | D |-------------------| C |
                        +---+#####[RaP_D(C)]####+---+

                -----physical links     xxxxxx  Failure Node
                *****RcW_D              ######  RaP_D
```

Figure 6. Wrapping for node failure

4.3.2. Short Wrapping

With the traditional wrapping protection scheme, Protection switching is executed at both nodes detecting the failure, consequently the traffic will be wrapped twice. This mechanism will cause additional latency and bandwidth consumption when traffic is switched to the protection path.

With short wrapping protection, data traffic switching is executed only at the upstream node detecting the failure, and data traffic leaves the ring in the protection ring tunnel at the egress node. This scheme can reduce the additional latency and bandwidth consumption when traffic is switched to the protection path.

In the traditional wrapping solution, the protection ring tunnel is a closed ring in normal state, while in the short wrapping solution, the protection ring tunnel is ended at the egress node, which is similar to the working ring tunnel. Short wrapping is easy to implement in shared ring protection because both the working and protection ring tunnels are terminated on the egress nodes. Figure 7

shows the clockwise working ring tunnel and the anticlockwise protection ring tunnel with node D as the egress node.

4.3.2.1. Short Wrapping for Link Failure

As shown in Figure 7, in normal state, LSP1 is carried by the clockwise working ring tunnel (RcW_D) through the path A->B->C->D. When a link failure between Node B and Node C occurs, Node B switches The working ring tunnel RcW_D to the protection ring tunnel RaP_D in the opposite direction. The difference occurs in the protection ring tunnel at egress node. In short wrapping protection, Rap_D ends in Node D and then traffic will be forwarded based on the LSP labels. Thus with short wrapping mechanism, LSP1 will follow the path A->B->A->F->E->D when link failure between Node B and Node C happens. For node failure, the protection with short wrapping is similar to the mechanism with link failure.

```
                        +---+#####[RaP_D(F)]######+---+
                        | F |---------------------| A | +-- LSP1
                        +---+*****[RcW_D(A)]******+---+
                        #/*                        *\#
              [RaP_D(E)]#/*[RcW_D(F)]      [RcW_D(B)]*\#RaP_D(A)
                       #/*                           *\#
                      +---+                         +---+
                      | E |                         | B |
                      +---+                         +---+
                       #\                           *x#
              [RaP_D(D)]#\                [RcW_D(C)]*x#RaP_D(B)
                        #\                         *x#
                        +---+*****[RcW_D(D)]****+---+
               LSP1  +-- | D |-------------------| C |
                        +---+                   +---+

                  ----- physical links    xxxxx Failure Link
                  ****** RcW_D           ###### RaP_D
```

Figure 7. Short wrapping for link failure

4.3.2.2. Short Wrapping for Node Failure

For the failure scenarios which happen on a non-egress node, short wrapping protection switching is similar to the link failure as described in the previous section. This section specifies the scenario of egress node failure.

As shown in Figure 8, LSP1 enters the ring on node A, and leaves the ring on node D. in normal state, LSP1 is carried by the clockwise working ring tunnel (RcW_D) through the path A->B->C->D. When node D

fails, traffic of LSP1 cannot be protected by any ring tunnels which use node D as the egress node. However, before the failure location is propagated to all the ring nodes, node C switches all the traffic on the working ring tunnel RcW_D to the protection ring tunnel RaP_D in the opposite direction. When the traffic arrives at node E which also detects the failure of node D, the protection ring tunnel RaP_D cannot be used to forward traffic to node D. Since with short wrapping mechanism, protection switching can only be performed once from the working ring tunnel to the protection ring tunnel, thus node E MUST NOT switch the traffic which is already carried on the protection ring tunnel back to the working ring tunnel in the opposite direction. Instead, node E will discard the traffic received on RaP_D locally. This can avoid the temporary traffic loop when the faiulre happens on the egress node of the ring tunnel. This also illustrates one of the benefits of having separate working and protection ring tunnels in each ring direction.

```
                      +---+#####[RaP_D(F)]######+---+
                      | F |---------------------| A | +-- LSP1
                      +---+*****[RcW_D(A)]******+---+
                      #/*                        *\#
[RaP_D(E)]#/*[RcW_D(F)]      [RcW_D(B)]*\#RaP_D(A)
         #/*                                  *\#
       +---+                          +---+
       | E |                          | B |
       +---+                          +---+
         #\                            */#
[RaP_D(D)]#\                [RcW_D(C)]*/#RaP_D(B)
           #\                        */#
           xxxxx*****[RcW_D(D)]****+---+
 LSP1  +-- x D x-------------------| C |
           xxxxx                   +---+

   -----physical links     xxxxxx  Failure Node
   *****RcW_D              ######  RaP_D
```

Figure 8. Short Wrapping for egress node failure

4.3.3. Steering

In ring topology, each working ring tunnel is associated with a protection ring tunnel in the opposite direction, and every node can obtain the ring topology either by configuration or via some topology discovery mechanism. The ring topology and the connectivity (Intact or Severed) between the adjacent ring nodes form the ring map. Every ring node maintains its ring map. When a failure occurs in the ring, the nodes that detect the failure via OAM mechanism will transmit the failure information in the opposite direction of the failure hop by

hop along the ring. When a node receives the message that identifies a failure, it can quickly determine the location of the fault by using the topology information that is maintained by the node and upate the ring map accordingly, then it can determine whether the LSPs entering the ring locally need to switchover or not. For LSPs that needs to switchover, it will switch the LSPs from the working ring tunnels to its corresponding protection ring tunnels.

```
  Ring map of F                                          +--LSPl
 +-+-+-+-+-+-+-+     +---+ ###[RaP_D(F)]### +---/  +-+-+-+-+-+-+-+
 |F|A|B|C|D|E|F|     | F | ---------------- | A |  |A|B|C|D|E|F|A|
 +-+-+-+-+-+-+-+     +---+ ***[RcW_D(A)]*** +---+  +-+-+-+-+-+-+-+
  |I|I|I|S|I|I|                                       |I|I|S|I|I|I|
  +-+-+-+-+-+-+      #/*                      *\#    +-+-+-+-+-+-+
        [RaP_D(E)]  #/*           [RcW_D(B)]  *\# [RaP_D(A)]
                   #/* [RcW_D(F)]              *\#
+-+-+-+-+-+-+-+   #/*                            *\#
|E|F|A|B|C|D|E| +---+                             +---+ +-- LSP2
+-+-+-+-+-+-+-+ | E |                             | B |  +-+-+-+-+-+-+-+
 |I|I|I|I|S|I|  +---+                             +---+  |B|C|D|E|F|A|B|
 +-+-+-+-+-+-+     #\*                              */#   +-+-+-+-+-+-+-+
                    #\* [RcW_D(E)]    [RcW_D(C)] */#     |I|S|I|I|I|I|
        [RaP_D(D)]   #\*                         */#      +-+-+-+-+-+-+
                      #\*                      */# [RaP_D(B)]
+-+-+-+-+-+-+-+       +---+     [RcW_D(D)]    +---+    +-+-+-+-+-+-+-+
|D|E|F|A|B|C|D|  +--  | D | xxxxxxxxxxxxxxxxx | C |    |C|D|E|F|A|B|C|
+-+-+-+-+-+-+-+ LSP1  +---+     [RaP_D(C)]    +---+    +-+-+-+-+-+-+-+
 |I|I|I|I|I|S|  LSP2                                     |S|I|I|I|I|I|
 +-+-+-+-+-+-+                                           +-+-+-+-+-+-+

       ----- physical links  ***** RcW_D  ##### RaP_D
          I: Intact   S: Severed
```

Figure 9. Steering operation and protection switching

As shown in Figure 9, LSP1 enters the ring from Node A while LSP2 enters the ring from Node B, and both of them have the same destination node D.

In the normal state, LSP1 is carried by the clockwise working ring tunnel (RcW_D) through the path A->B->C->D, the label operation is: [LSP1] -> [RcW_D(B)|LSP1](NodeA) -> [RcW_D(C)| LSP1](NodeB) -> [RcW_D(D)|LSP1](NodeC) -> [LSP1] (data traffic carried by LSP1) .

LSP2 is carried by the clockwise working ring tunnel (RcW_D) throught the path B->C->D, the label operation is: [LSP2] -> [RcW_D(C)|LSP2](NodeB) -> [RcW_D(D)|LSP2](NodeC) -> [LSP2] (data traffic carried by LSP2) .

If the link between nodes C and D fails, according to the fault detection and distribution mechanisms, Node D will find out that there is a failure in the link between C and D, and it will update the link state of its ring topology, changing the link between C and D from normal to fault. In the direction that opposite to the failure position, Node D will send the state report message to Node E, informing Node E of the fault between C and D, and E will update the link state of its ring topology accordingly, changing the link between C and D from normal to fault. In this way, the state report message is sent hop by hop in the clockwise direction. Similar to Node D, Node C will send the failure information in the anti-clockwise direction.

When Node A receives the failure report message and updates the link state of its ring topology, it is aware that there is a fault on the clockwise working ring tunnel to node D (RcW_D), and LSP1 enters the ring locally and is carried by this ring tunnel, thus Node A will decide to switch the LSP1 onto the anticlockwise protection ring tunnel to node D (RaP_D). After the switchover, LSP1 will follow the path A->F->E->D, the label operation is: [LSP1] -> [RaP_D(F)|LSP1](NodeA) -> [RaP_D(E)|LSP1](NodeF) -> [RaP_D(D)|LSP1](NodeE) -> [LSP1] (data traffic carried by LSP1).

The same also apply to the operation of LSP2. When Node B updates the link state of its ring topology, and finds out that the working ring tunnel RcW_D has failed, it will switch the LSP2 to the anticlockwise protection tunnel RaP_D. After the switchover, LSP2 goes through the path B->A->F->E->D, and the label operation is: [LSP2] -> [RaP_D(A)|LSP2](NodeB) -> [RaP_D(F)|LSP2](NodeA) -> [RaP_D(E)|LSP2](NodeF) -> [RaP_D(D)|LSP2](NodeE) -> [LSP2](data traffic carried by LSP2).

Then assume the link between nodes A and B breaks down, as shown in Figure 10. Similar to the above failure case, Node B will detect a fault in the link between A and B, and it will update the link state of its ring topology, changing the link state between A and B from normal to fault. The state report message is sent hop by hop in the clockwise direction, notifying every node that there is a fault between node A and B, and every node updates the link state of its ring topology. As a result, Node A will detect a fault in the working ring tunnel to node D, and switch LSP1 to the protection ring tunnel, while Node B determine that the working ring tunnel for LSP2 still works fine, and will not perform the switchover.

```
                                                      /-- LSPl
 +-+-+-+-+-+-+-+      +---+ ###[RaP_D(F)]####  +---/  +-+-+-+-+-+-+-+
 |F|A|B|C|D|E|F|      | F | -----------------  | A |  |A|B|C|D|E|F|A|
 +-+-+-+-+-+-+-+      +---+ ***[RcW_D(A)]****  +---+  +-+-+-+-+-+-+-+
  |I|S|I|I|I|I|       #/*                       x      |S|I|I|I|I|I|
  +-+-+-+-+-+-+      #/*                         x     +-+-+-+-+-+-+
        [RaP_D(E)]  #/*[RcW_D(F)]       [RcW_D(B)]x [RaP_D(A)]
                   #/*                            x    +-- LSP2
 +-+-+-+-+-+-+-+  +---+                             +---++-+-+-+-+-+-+-+
 |E|F|A|B|C|D|E|  | E |                             | B ||B|C|D|E|F|A|B|
 +-+-+-+-+-+-+-+  +---+                             +---++-+-+-+-+-+-+-+
  |I|I|S|I|I|I|     #\*                            */#    |I|I|I|I|I|S|
  +-+-+-+-+-+-+      #\*[RcW_D(E)]    [RcW_D(C)]  */#     +-+-+-+-+-+-+
          [RaP_D(D)]  #\*                         */# [RaP_D(B)]
 +-+-+-+-+-+-+-+       #\*                       */#     +-+-+-+-+-+-+-+
 |D|E|F|A|B|C|D|       +---+ ***[RcW_D(D)]*** +---+     |C|D|E|F|A|B|C|
 +-+-+-+-+-+-+-+  +--  | D | ---------------- | C |     +-+-+-+-+-+-+-+
  |I|I|I|S|I|I|   LSP1 +---+ ###[RaP_D(C)]### +---+      |I|I|I|I|S|I|
  +-+-+-+-+-+-+   LSP2                                    +-+-+-+-+-+-+

           ----- physical links  ***** RcW_D  ##### RaP_D
```

Figure 10. Steering operation and protection switching (2)

4.4. Interconnected Ring Protection

4.4.1. Interconnected Ring Topology

Interconnected ring topology is often used in MPLS-TP networks. This document will discuss two typical interconnected ring topologies:

1. Single-node interconnected rings

   In single-node interconnected rings, the connection between the two rings is through a single node. Because the interconnection node is in fact a single point of failure, this topology should be avoided in real transport networks. Figure 10 shows the topology of single-node interconnected rings. Node C is the interconnection node between Ring1 and Ring2.

2. Dual-node interconnected rings

   In dual-node interconnected rings, the connection between the two rings is through two nodes. The two interconnection nodes belong to both interconnected rings. This topology can recover from one interconnection node failure.

Figure 11 shows the topology of single-node interconnected rings. Node C is the interconnection node between Ring1 and Ring2.

```
+---+      +---+                        +---+      +---+
| A |------| B |-----              -----| G |------| H |
+---+      +---+      \           /     +---+      +---+
  |                    \         /                    |
  |                     \ +---+ /                     |
  |        Ring1          | C |         Ring2         |
  |                     / +---+ \                     |
  |                    /         \                    |
+---+      +---+      /           \     +---+      +---+
| F |------| E |-----              -----| J |------| I |
+---+      +---+                        +---+      +---+
```

Figure 11. Single-node interconnected rings

Figure 12 shows the topology of dual-node interconnected rings. Nodes C and Node D are the interconnection nodes between Ring1 and Ring2.

```
+---+      +---+      +---+      +---+      +---+
| A |------| B |------| C |------| G |------| H |
+---+      +---+      +---+      +---+      +---+
  |                    | |                    |
  |                    | |                    |
  |        Ring1       | |       Ring2        |
  |                    | |                    |
  |                    | |                    |
+---+      +---+      +---+      +---+      +---+
| F |------| E |------| D |------| J |------| I |
+---+      +---+      +---+      +---+      +---+
```

Figure 12. Dual-node interconnected rings

4.4.2. Interconnected Ring Protection Mechanisms

Interconnected rings can be treated as two independent rings. Ring protection switching (RPS) protocol operates on each ring independently. Failure in one ring only triggers protection switching on the ring itself and does not affect the other ring. This way, protection switching on each ring is the same as the mechanisms described in section 4.3.

The service LSPs that traverse the interconnected rings via the interconnection nodes MUST use different ring tunnels in different rings, and the service LSPs traversing the interconnected rings are stitched by the interconnection node. On the interconnection node,

the ring tunnel label used in the source ring will be popped, the service LSP label will be swapped, and the ring tunnel label of the destination ring will be pushed.

In the dual-node interconnected ring scenario, the two interconnection nodes can be managed as a virtual interconnection node group. Each ring should assign working and protection ring tunnels for the virtual interconnection node group. Both the interconnection nodes in the virtual interconnection node group can terminate the working ring tunnel of each ring. The protection ring tunnel is used to protect the working ring tunnel of each ring and can be terminated by any node in the virtual interconnection node group.

On the nodes in the virtual interconnection node group of the dual-node interconnected ring, the same label is allocated for each service LSP. This way any interconnection node in the virtual node group can stitch the service LSPs between the source ring tunnel and the destination ring tunnel.

When the service traffic passes through the interconnection node, the direction of the working ring tunnels in each ring for this service traffic should be the same. For example, if the working ring tunnel follows the clockwise direction in Ring1, the working ring tunnel for the same service traffic in Ring2 SHOULD also follow the clockwise direction when the service leaves Ring1 and enters Ring2.

4.4.3. Ring Tunnels in Interconnected Rings

The same ring tunnels as described in section 4.1 are used in each ring of the interconnected rings. Note that ring tunnels to the virtual interconnection node group will be established by each ring of the interconnected rings, i.e.:

- o one clockwise working ring tunnel to the virtual interconnection node group
- o one anticlockwise protection ring tunnel to the virtual interconnection node group
- o one anticlockwise working ring tunnel to the virtual interconnection node group
- o one clockwise protection ring tunnel to the virtual interconnection node group

These ring tunnels will terminated at all nodes in the virtual interconnection node group.

For example, all the ring tunnels on Ring1 of Figure 12 are established as follows:

- To Node A: R1cW_A, R1aW_A, R1cP_A, R1aP_A
- To Node B: R1cW_B, R1aW_B, R1cP_B, R1aP_B
- To Node C: R1cW_C, R1aW_C, R1cP_C, R1aP_C
- To Node D: R1cW_D, R1aW_D, R1cP_D, R1aP_D
- To Node E: R1cW_E, R1aW_E, R1cP_E, R1aP_E
- To Node F: R1cW_F, R1aW_F, R1cP_F, R1aP_F
- To the virtual interconnection node group (including Node F and Node A): R1cW_F&A, R1aW_F&A, R1cP_F&A, R1aP_F&A

All the ring tunnels established in Ring2 in Figure 13 are provisioned as follows:

- To Node A: R2cW_A, R2aW_A, R2cP_A, R2aP_A
- To Node F: R2cW_F, R2aW_F, R2cP_F, R2aP_F
- To Node G: R2cW_G, R2aW_G, R2cP_G, R2aP_G
- To Node H: R2cW_H, R2aW_H, R2cP_H, R2aP_H
- To Node I: R2cW_I, R2aW_I, R2cP_I, R2aP_I
- To Node J: R2cW_J, R2aW_J, R2cP_J, R2aP_J
- To the virtual interconnection node group(including Node F and Node A): R2cW_FandA, R2aW_FandA, R2cP_FandA, R2aP_FandA

```
                          +---+cccccccccccc +---+
                          | H |-------------| I |--->LSP1
                          +---+             +---+
                          c/a                   a\
                         c/a                     a\
                        c/a                       a\
                      +---+                     +---+
                      | G |        Ring2        | J |
                      +---+                     +---+
                        c\a                      a/c
                         c\a                    a/c
                          c\a  aaaaaaaaaaaaa   a/c
                          +---+cccccccccccccc+---+
                          | F |-------------| A |
                          +---+cccccccccccccc+---+
                          c/aaaaaaaaaaaaaaaaaaa a\
                         c/                      a\
                        c/                        a\
                      +---+                     +---+
                      | E |        Ring1        | B |
                      +---+                     +---+
                        c\a                      a/c
                         c\a                    a/c
                          c\a                  a/c
                          +---+aaaaaaaaaaaa +---+
                LSP1--->| D |-------------| C |
                          +---+cccccccccccccc+---+

                           cccccccccccc  R1cW_F&A
                           aaaaaaaaaaaa  R1aP_F&A
                           cccccccccccc  R2cW_I
                           aaaaaaaaaaaa  R2aP_I
```

Figure 13. Ring tunnels for the interconnected rings

4.4.4. Interconnected Ring Switching Procedure

As shown in Figure 13, for the service traffic LSP1 which enters Ring1 at Node D and leaves Ring1 at Node F and continues to enter Ring2 at Node F and leaves Ring2 at Node I, the protection scheme is described as below.

In normal state, LSP1 follows R1cW_F&A in Ring1 and R2CW_I in Ring2. The label used for the working ring tunnel R1cW_F&A in Ring1 is popped and the label used for the working ring tunnel R2cW_I will be pushed based the inner label lookup at the interconnection node F. The working path that the service traffic LSP1 follows is: LSP1->R1cW_F&A (D->E->F)->R2cW_I(F->G->H->I)->LSP1.

In case of link failure, for example, when a failure occurs on the link between Node F and Node E, Nodes F and E will detect the failure and execute protection switching as described in 4.3.1.1. The path that the service traffic LSP1 follows after switching change to LSP1->R1cW_F&A(D->E)->R1aP_F&A(E->D->C->B->A->F)->R1cW_F(F) ->R2cW_I(F->G->H->I)->LSP1.

In case of a non interconnection node failure, for example, when the failure occurs at Node E in Ring1, Nodes F and D will detect the failure and execute protection switching as described in 4.3.1.2. The path that the service traffic LSP1 follows after switching becomes: LSP1->R1cW_F&A(D)->R1aP_F&A(D->C->B->A->F)-> R1cW_F(F)->R2cW_I(F->G->H->I).

In case of an interconnection node failure, for example, when the failure occurs at the interconnection Node F. Nodes E and A in Ring1 will detect the failure, and execute protection switching as described in 4.3.1.2. Nodes G and A in Ring2 will also detects the failure, and execute protection switching. The path that the service traffic LSP1 follows after switching is: LSP1->R1cW_F&A(D->E)->R1aP_F&A(E->D->C->B->A)->R1cW_A(A) ->R2aP_I(A->J->I)->LSP1.

4.4.5. Interconnected Ring Detection Mechanism

As show in Figure 14, the service traffic LSP1 traverses A->B->C in Ring1 and C->G->H->I in Ring2. Node C and Node D are the interconnection nodes. When both the link between Node C and Node G and the link between Node C and Node D fail, the ring tunnel from Node C to Node I in Ring2 becomes unreachable. However, Node D is still available, and LSP1 can still reach Node I.

```
      +---+ *********+---+**********+---+          +---+**********+---+
LSP1->| A |----------| B |----------| C |XXXXXXXXXX| G |----------| H |
      +---+##########+---+##########+---+          +---+##########+---+
        |#                            X                            #|*
        |#                            X                            #|*
        |#           Ring1            X           Ring2            #|*
        |#                            X                            #|*
        |#                            X                            #|*
      +---+##########+---+##########+---+######### +---+##########+---+
      | F |----------| E |----------| D |----------| J |----------| I | ->LSP1
      +---+          +---+          +---+          +---+          +---+

                         ***********  R1cW_C&D
                         ###########  R1aP_C&D
                         ***********  R2cW_I
                         ###########  R2aP_I
```

Figure 14. Interconnected ring

In order to achieve this, the interconnection nodes need to know the ring topology of each ring so that they can judge whether a node is reachable. This judgment is based on the knowledge of each ring topology and the fault location as described in section 3.4. The ring topology can be obtained from the NMS or topology discovery mechanisms. The fault location can be obtained by transmitting the fault information around the ring. The nodes that detect the failure will transmit the fault information in the opposite direction node by node in the ring. When the interconnection node receives the message that informs the failure, it will quickly calculate the location of the fault by the topology information that is maintained by itself and determines whether the LSPs entering the ring at itself can reach the destination. If the destination node is reachable, the LSP will leave the source ring and enter the destination ring. If the destination node is not reachable, the LSP will switch to the anticlockwise protection ring tunnel.

In Figure 14, Node C determines that the ring tunnel to Node I is unreachable, the service traffic LSP1 for which the destination node on the ring tunnel is Node I should switch to the protection LSP (R1aP_C&D) and consequently the service traffic LSP1 traverses the interconnected rings at Node D. Node D will remove the ring tunnel label of Ring1 and add the ring tunnel label of Ring2.

5. Ring Protection Coordination Protocol

5.1. RPS Protocol

The MSRP protection operation MUST be controlled with the help of the Ring Protection Switch Protocol (RPS). The RPS processes in each of the individual ring nodes that form the ring SHOULD communicate using the G-ACh channel.

The RPS protocol MUST carry the ring status information and RPS requests, i.e., automatically initiated and externally initiated, between the ring nodes.

Each node on the ring MUST be uniquely identified by assigning it a node ID. The node ID MUST be unique on each ring. The maximum number of nodes on the ring supported by the RPS protocol is 127. The node ID SHOULD be independent of the order in which the nodes appear on the ring. The node ID is used to identity the source and destination nodes of each RPS request.

Every node obtains the ring topology either by configuration or via some topology discovery mechanism. The ring map consists of the ring topology information, and connectivity status (Intact or Severed) between the adjacent ring nodes, which is determined via the OAM message exchange between the adjacent nodes. The ring map is used by every ring node to determine the switchover behavoir of the ring tunnels.

When no protection switching is active on the ring, each node MUST dispatch periodically RPS requests to the two adjacent nodes, indicating No Request (NR). When a node determines that a protection switching is required, it MUST send the appropriate RPS request in both directions.

```
                +---+ A->B(NR)    +---+ B->C(NR)    +---+ C->D(NR)
        -------| A |-------------| B |-------------| C |-------
      (NR)F<-A +---+    (NR)A<-B +---+    (NR)B<-C +---+
```

Figure 15. RPS communication between the ring nodes in case of no failures in the ring

A destination node is a node that is adjacent to a node that identified a failed span. When a node that is not the destination node receives an RPS request and it has no higher priority local request, it MUST transfer in the same direction the RPS request as received. In this way, the switching nodes can maintain direct RPS protocol communication in the ring.

```
                  +---+ C->B(SF)    +---+ B->C(SF)    +---+ C->B(SF)
          -------| A |-------------| B |----- X -----| C |-------
       (SF)C<-B +---+    (SF)C<-B +---+    (SF)B<-C +---+
```

Figure 16. RPS communication between the ring nodes in case of failure between nodes B and C

Note that in the case of a bidirectional failure such as a cable cut, the two adjacent nodes detect the failure and send each other an RPS request in opposite directions.

- o In rings utilizing the wrapping protection. When the destination node receives the RPS request it MUST perform the switch from/to the working ring tunnels to/from the protection ring tunnels if it has no higher priority active RPS request.

- o In rings utilizing the steering protection. When a ring switch is required, any node MUST perform the switches if its added/dropped traffic is affected by the failure. Determination of the affected traffic SHOULD be performed by examining the RPS requests (indicating the nodes adjacent to the failure or failures) and the stored ring maps (indicating the relative position of the failure and the added traffic destined towards that failure).

When the failure has cleared and the Wait-to-Restore (WTR) timer has expired, the nodes sourcing RPS requests MUST drop their respective switches (tail end) and MUST source an RPS request carrying the NR code. The node receiving from both directions such RPS request (head end) MUST drop its protection switches.

A protection switch MUST be initiated by one of the criteria specified in Section 3.2. A failure of the RPS protocol or controller MUST NOT trigger a protection switch.

Ring switches MUST be preempted by higher priority RPS requests. For example, consider a protection switch that is active due to a manual switch request on the given span, and another protection switch is required due to a failure on another span. Then an RPS request MUST be generated, the former protection switch MUST be dropped, and the latter protection switch established.

MSRP mechanism SHOULD support multiple protection switches in the ring, resulting the ring being segmented into two or more separate segments. This may happen when several RPS requests of the same priority exist in the ring due to multiple failures or external switch commands.

Proper operation of the MSRP mechanism relies on all nodes having knowledge of the state of the ring (nodes and spans) so that nodes do not preempt existing RPS request unless they have a higher-priority RPS request. In order to accommodate ring state knowledge, during a protection switch the RPS requests MUST be sent in both directions.

5.1.1. Transmission and Acceptance of RPS Requests

A new RPS request MUST be transmitted immediately when a change in the transmitted status occurs.

The first three RPS protocol messages carrying new RPS request SHOULD be transmitted as fast as possible. For fast protection switching within 50 ms, the interval of the first three RPS protocol messages SHOULD be 3.3 ms. The successive RPS requests SHOULD be transmitted with the interval of 5 seconds.

5.1.2. RPS PDU Format

Figure 17 depicts the format of an RPS packet that is sent on the G-ACh. The Channel Type field is set to indicate that the message is an RPS message. The ACH MUST NOT include the ACH TLV Header [RFC5586] meaning that no ACH TLVs can be included in the message.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 1|0 0 0 0|0 0 0 0 0 0 0 0|    RPS Channel Type (TBD)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Dest Node ID  | Src Node ID   |   Request     |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17. G-ACh RPS Packet Format

The following fields MUST be provided:

- o Destination Node ID: The destination node ID MUST always be set to value of the node ID of the adjacent node. The Node ID MUST be unique on each ring. Valid destination node ID values are 1-127.

- o Source node ID: The source node ID MUST always be set to the ID value of the node generating the RPS request. The Node ID MUST be unique on each ring. Valid source node ID values are 1-127.

- o RPS request code: A code consisting of eight bits as specified below:

Bits (MSB - LSB)	Condition, State or external Request	Priority
0 0 0 0 1 1 1 1	Lockout of Protection (LP)	highest
0 0 0 0 1 1 0 1	Forced Switch (FS)	
0 0 0 0 1 0 1 1	Signal Fail (SF)	
0 0 0 0 0 1 1 0	Manual Switch (MS)	
0 0 0 0 0 1 0 1	Wait-To-Restore (WTR)	
0 0 0 0 0 0 1 1	Exercise (EXER)	
0 0 0 0 0 0 0 1	Reverse Request (RR)	
0 0 0 0 0 0 0 0	No Request (NR)	lowest

5.1.3. Ring Node RPS States

Idle state: A node is in the idle state when it has no RPS request and is sourcing and receiving NR code to/from both directions.

Switching state: A node not in the idle or pass-through states is in the switching state.

Pass-through state: A node is in the pass-through state when its highest priority RPS request is a request not destined to it or sourced by it. The pass-through is bidirectional.

5.1.3.1. Idle State

A node in the idle state MUST source the NR request in both directions.

A node in the idle state MUST terminate RPS requests flow in both directions.

A node in the idle state MUST block the traffic flow on protection LSPs/tunnels in both directions.

5.1.3.2. Switching State

A node in the switching state MUST source RPS request to adjacent node with its highest RPS request code in both directions when it detects a failure or receives an external command.

A node in the switching state MUST terminate RPS requests flow in both directions.

As soon as it receives an RPS request from the short path, the node to which it is addressed MUST acknowledge the RPS request by replying with the RR code on the short path, and with the received RPS request code on the long path. Here the short path refers to the shorter span on the ring between the source and destination node of the RPS request, and the long path refers to the longer span on the ring between the source and destination node of the RPS request.

This rule refers to the unidirectional failure detection: the RR SHOULD be issued only when the node does not detect the failure condition (i.e., the node is a head end), that is, it is not applicable when a bidirectional failure is detected, because, in this case, both nodes adjacent to the failure will send an RPS request for the failure on both paths (short and long).

The following switches MUST be allowed to coexist:

- o LP and LP
- o FS and FS
- o SF and SF
- o FS and SF

When multiple MS RPS requests over different spans exist at the same time, no switch SHOULD be executed and existing switches MUST be dropped. The nodes MUST signal, anyway, the MS RPS request code.

Multiple EXER requests MUST be allowed to coexist in the ring.

A node in a ring switching state that receives the external command LP for the affected span MUST drop its switch and MUST signal NR for the locked span if there is no other RPS request on another span. Node still SHOULD signal relevant RPS request for another span.

5.1.3.3. Pass-through State

When a node is in a pass-through state, it MUST transfer the received RPS Request in the same direction.

When a node is in a pass-through state, it MUST enable the traffic flow on protection ring tunnels in both directions.

5.1.4. RPS State Transitions

All state transitions are triggered by an incoming RPS request change, a WTR expiration, an externally initiated command, or locally detected MPLS-TP section failure conditions.

RPS requests due to a locally detected failure, an externally initiated command, or received RPS request shall pre-empt existing RPS requests in the prioritized order given in Section 3.1.2, unless the requests are allowed to coexist.

5.1.4.1. Transitions Between Idle and Pass-through States

The transition from the idle state to pass-through state MUST be triggered by a valid RPS request change, in any direction, from the NR code to any other code, as long as the new request is not destined to the node itself. Both directions move then into a pass-through state, so that, traffic entering the node through the protection Ring tunnels are transferred transparently through the node.

A node MUST revert from pass-through state to the idle state when it detects NR codes incoming from both directions. Both directions revert simultaneously from the pass-through state to the idle state.

5.1.4.2. Transitions Between Idle and Switching States

Transition of a node from the idle state to the switching state MUST be triggered by one of the following conditions:

- o A valid RPS request change from the NR code to any code received on either the long or the short path and destined to this node

- o An externally initiated command for this node

- o The detection of an MPLS-TP section layer failure at this node

Actions taken at a node in the idle state upon transition to switching state are:

- o For all protection switch requests, except EXER and LP, the node MUST execute the switch

- o For EXER, and LP, the node MUST signal appropriate request but not execute the switch

A node MUST revert from the switching state to the idle state when it detects NR codes received from both directions.

- o At the tail end: When a WTR time expires or an externally initiated command is cleared at a node, the node MUST drop its switch, transit to the Idle State and signal the NR code in both directions.

- o At the head end: Upon reception of the NR code, from both directions, the head-end node MUST drop its switch, transition to Idle State and signal the NR code in both directions.

5.1.4.3. Transitions Between Switching States

When a node that is currently executing any protection switch receives a higher priority RPS request (due to a locally detected failure, an externally initiated command, or a ring protection switch request destined to it) for the same span, it MUST update the priority of the switch it is executing to the priority of the received RPS request.

When a failure condition clears at a node, the node MUST enter WTR condition and remain in it for the appropriate time-out interval, unless:

- o A different RPS request with a higher priority than WTR is received

- o Another failure is detected

- o An externally initiated command becomes active

The node MUST send out a WTR code on both the long and short paths.

When a node that is executing a switch in response to incoming SF RPS request (not due to a locally detected failure) receives a WTR code (unidirectional failure case), it MUST send out RR code on the short path and the WTR on the long path.

5.1.4.4. Transitions Between Switching and Pass-through States

When a node that is currently executing a switch receives an RPS request for a non-adjacent span of higher priority than the switch it is executing, it MUST drop its switch immediately and enter the pass-through state.

The transition of a node from pass-through to switching state MUST be triggered by:

- o An equal priority, a higher priority, or an allowed coexisting externally initiated command

- o The detection of an equal priority, a higher priority, or an allowed coexisting automatic initiated command

- o The receipt of an equal, a higher priority, or an allowed coexisting RPS request destined to this node

5.2. RPS State Machine

5.2.1. Initial States

	State	Signaled RPS
A	Idle Working: no switch Protection: no switch	NR
B	Pass-trough Working: no switch Protection: pass through	N/A
C	Switching - LP Working: no switch Protection: no switch	LP
D	Idle - LW Working: no switch Protection: no switch	NR
E	Switching - FS Working: switched Protection: switched	FS
F	Switching - SF Working: switched Protection: switched	SF
G	Switching - MS Working: switched Protection: switched	MS
H	Switching - WTR Working: switched Protection: switched	WTR
I	Switching - EXER Working: no switch Protection: no switch	EXER

5.2.2. State transitions When Local Request is Applied

In the state description below 'O' means that new local request will be rejected because of exiting request.

Initial state	New request	New state

```
   A (Idle)             LP                C (Switching - LP)
                        LW                D (Idle - LW)
                        FS                E (Switching - FS)
                        SF                F (Switching - SF)
                        Recover from SF   N/A
                        MS                G (Switching - MS)
                        Clear             N/A
                        WTR expires       N/A
                        EXER              I (Switching - EXER)
   ======================================================================
   Initial state        New request       New state
   -------------        -----------       ---------
   B (Pass-trough)      LP                C (Switching - LP)
                        LW                B (Pass-trough)
                        FS                O - if current state is due to
                                              LP sent by another node
                                          E (Switching - FS) - otherwise
                        SF                O - if current state is due to
                                              LP sent by another node
                                          F (Switching - SF) - otherwise
                        Recover from SF   N/A
                        MS                O - if current state is due to
                                              LP, SF or FS sent by
                                              another node
                                          G (Switching - MS) - otherwise
                        Clear             N/A
                        WTR expires       N/A
                        EXER              O
   ======================================================================
   Initial state        New request       New state
   -------------        -----------       ---------
   C (Switching - LP)   LP                N/A
                        LW                O
                        FS                O
                        SF                O
                        Recover from SF   N/A
                        MS                O
                        Clear             A (Idle) - if there is no
                                             failure in the ring
                                          F (Switching - SF) - if there
                                             is a failure at this node
                                          B (Pass-trough) - if there is
                                             a failure at another node
                        WTR expires       N/A
                        EXER              O
   ======================================================================
   Initial state        New request       New state
   -------------        -----------       ---------
```

D (Idle - LW)	LP	C (Switching - LP)
	LW	N/A - if on the same span D (Idle - LW) - if on another span
	FS	O - if on the same span E (Switching - FS) - if on another span
	SF	O - if on the addressed span F (Switching - SF) - if on another span
	Recover from SF	N/A
	MS	O - if on the same span G (Switching - MS) - if on another span
	Clear	A (Idle) - if there is no failure on addressed span F (Switching - SF) - if there is a failure on this span
	WTR expires	N/A
	EXER	O

Initial state	New request	New state
E (Switching - FS)	LP	C (Switching - LP)
	LW	O - if on another span D (Idle - LW) - if on the same span
	FS	N/A - if on the same span E (Switching - FS) - if on another span
	SF	O - if on the addressed span E (Switching - FS) - if on another span
	Recover from SF	N/A
	MS	O
	Clear	A (Idle) - if there is no failure in the ring F (Switching - SF) - if there is a failure at this node B (Pass-trough) - if there is a failure at another node
	WTR expires	N/A
	EXER	O

Initial state	New request	New state
F (Switching - SF)	LP	C (Switching - LP)
	LW	O - if on another span

```
                                        D (Idle - LW) - if on the same
                                           span
                        FS              E (Switching - FS)
                        SF              N/A - if on the same span
                                        F (Switching - SF) - if on
                                           another span
                        Recover from SF H (Switching - WTR)
                        MS              O
                        Clear           N/A
                        WTR expires     N/A
                        EXER            O
   ======================================================================
   Initial state        New request     New state
   -------------        -----------     ---------
   G (Switching - MS)   LP              C (Switching - LP)
                        LW              O - if on another span
                                        D (Idle - LW) - if on the same
                                           span
                        FS              E (Switching - FS)
                        SF              F (Switching - SF)
                        Recover from SF N/A
                        MS              N/A - if on the same span
                                        G (Switching - MS) - if on
                                           another span release the
                                           switches but signal MS
                        Clear           A
                        WTR expires     N/A
                        EXER            O
   ======================================================================
   Initial state        New request     New state
   -------------        -----------     ---------
   H (Switching - WTR)  LP              C (Switching - LP)
                        LW              D (Idle - W)
                        FS              E (Switching - FS)
                        SF              F (Switching - SF)
                        Recover from SF N/A
                        MS              G (Switching - MS)
                        Clear           A
                        WTR expires     A
                        EXER            O
   ======================================================================
   Initial state        New request     New state
   -------------        -----------     ---------
   I (Switching - EXER) LP              C (Switching - LP)
                        LW              D (idle - W)
                        FS              E (Switching - FS)
                        SF              F (Switching - SF)
                        Recover from SF N/A
```

	MS	G (Switching - MS)
	Clear	A
	WTR expires	N/A
	EXER	N/A - if on the same span I (Switching - EXER)

5.2.3. State Transitions When Remote Request is Applied

The priority of a remote request does not depend on the side from which the request is received.

Initial state	New request	New state
A (Idle)	LP	C (Switching - LP)
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	MS	G (Switching - MS)
	WTR	N/A
	EXER	I (Switching - EXER)
	RR	N/A
	NR	A (Idle)

Initial state	New request	New state
B (Pass-trough)	LP	C (Switching - LP)
	FS	N/A - cannot happen when there is LP request in the ring E (Switching - FS) - otherwise
	SF	N/A - cannot happen when there is LP request in the ring F (Switching - SF) - otherwise
	MS	N/A - cannot happen when there is LP, FS or SF request in the ring G (Switching - MS) - otherwise
	WTR	N/A - cannot happen when there is LP, FS, SF or MS request in the ring
	EXER	N/A - cannot happen when there is LP, FS, SF, MS or WTR request in the ring I (Switching - EXER) - otherwise
	RR	N/A
	NR	A (Idle) - if received from

both sides

Initial state	New request	New state
C (Switching - LP)	LP	C (Switching - LP)
	FS	N/A - cannot happen when there is LP request in the ring
	SF	N/A - cannot happen when there is LP request in the ring
	MS	N/A - cannot happen when there is LP request in the ring
	WTR	N/A
	EXER	N/A - cannot happen when there is LP request in the ring
	RR	C (Switching - LP)
	NR	N/A

Initial state	New request	New state
D (Idle - LW)	LP	C (Switching - LP)
	FS	E (Switching - FS)
	SF	F (Switching - SF)
	MS	G (Switching - MS)
	WTR	N/A
	EXER	I (Switching - EXER)
	RR	N/A
	NR	D (Idle - LW)

Initial state	New request	New state
E (Switching - FS)	LP	C (Switching - LP)
	FS	E (Switching - FS)
	SF	E (Switching - FS)
	MS	N/A - cannot happen when there is FS request in the ring
	WTR	N/A
	EXER	N/A - cannot happen when there is FS request in the ring
	RR	E (Switching - FS)
	NR	N/A

Initial state	New request	New state
F (Switching - SF)	LP	C (Switching - LP)
	FS	F (Switching - SF)
	SF	F (Switching - SF)
	MS	N/A - cannot happen when there is SF request in the ring

```
                        WTR                N/A
                        EXER               N/A - cannot happen when there
                                                is SF request in the ring
                        RR                 F (Switching - SF)
                        NR                 N/A
   ======================================================================
   Initial state        New request        New state
   -------------        -----------        ---------
   G (Switching - MS)   LP                 C (Switching - LP)
                        FS                 E (Switching - FS)
                        SF                 F (Switching - SF)
                        MS                 G (Switching - MS) - release
                                              the switches but signal MS
                        WTR                N/A
                        EXER               N/A - cannot happen when there
                                                is MS request in the ring
                        RR                 G (Switching - MS)
                        NR                 N/A
   ======================================================================
   Initial state        New request        New state
   -------------        -----------        ---------
   H (Switching - WTR)  LP                 C (Switching - LP)
                        FS                 E (Switching - FS)
                        SF                 F (Switching - SF)
                        MS                 G (Switching - MS)
                        WTR                H (Switching - WTR)
                        EXER               N/A - cannot happen when there
                                                is WTR request in the ring
                        RR                 H (Switching - WTR)
                        NR                 N/A
   ======================================================================
   Initial state        New request        New state
   -------------        -----------        ---------
   I (Switching - EXER) LP                 C (Switching - LP)
                        FS                 E (Switching - FS)
                        SF                 F (Switching - SF)
                        MS                 G (Switching - MS)
                        WTR                N/A
                        EXER               I (Switching - EXER)
                        RR                 I (Switching - EXER)
                        NR                 N/A
   ======================================================================
```

5.2.4. State Transitions When Request Addresses to Another Node is Received

The priority of a remote request does not depend on the side from which the request is received.

Initial state	New request	New state
A (Idle)	LP	B (Pass-trough)
	FS	B (Pass-trough)
	SF	B (Pass-trough)
	MS	B (Pass-trough)
	WTR	B (Pass-trough)
	EXER	B (Pass-trough)
	RR	N/A
	NR	N/A

Initial state	New request	New state
B (Pass-trough)	LP	B (Pass-trough)
	FS	N/A - cannot happen when there is LP request in the ring B (Pass-trough) - otherwise
	SF	N/A - cannot happen when there is LP request in the ring B (Pass-trough) - otherwise
	MS	N/A - cannot happen when there is LP, FS or SF request in the ring B (Pass-trough) - otherwise
	WTR	N/A - cannot happen when there is LP, FS, SF or MS request in the ring B (Pass-trough) - otherwise
	EXER	N/A - cannot happen when there is LP, FS, SF, MS or WTR request in the ring B (Pass-trough) - otherwise
	RR	N/A
	NR	B (Pass-trough)

Initial state	New request	New state
C (Switching - LP)	LP	C (Switching - LP)
	FS	N/A - cannot happen when there is LP request in the ring
	SF	N/A - cannot happen when there

```
                                                 is LP request in the ring
                        MS                 N/A - cannot happen when there
                                                 is LP request in the ring
                        WTR                N/A - cannot happen when there
                                                 is LP in the ring
                        EXER               N/A - cannot happen when there
                                                 is LP request in the ring
                        RR                 N/A
                        NR                 N/A
   ======================================================================
   Initial state        New request        New state
   -------------        -----------        ---------
   D (Idle - LW)        LP                 B (Pass-trough)
                        FS                 B (Pass-trough)
                        SF                 B (Pass-trough)
                        MS                 B (Pass-trough)
                        WTR                B (Pass-trough)
                        EXER               B (Pass-trough)
                        RR                 N/A
                        NR                 N/A
   ======================================================================
   Initial state        New request        New state
   -------------        -----------        ---------
   E (Switching - FS)   LP                 B (Pass-trough)
                        FS                 E (Switching - FS)
                        SF                 E (Switching - FS)
                        MS                 N/A - cannot happen when there
                                                 is FS request in the ring
                        WTR                N/A - cannot happen when there
                                                 is FS request in the ring
                        EXER               N/A - cannot happen when there
                                                 is FS request in the ring
                        RR                 N/A
                        NR                 N/A
   ======================================================================
   Initial state        New request        New state
   -------------        -----------        ---------
   F (Switching - SF)   LP                 B (Pass-trough)
                        FS                 F (Switching - SF)
                        SF                 F (Switching - SF)
                        MS                 N/A - cannot happen when there
                                                 is SF request in the ring
                        WTR                N/A - cannot happen when there
                                                 is SF request in the ring
                        EXER               N/A - cannot happen when there
                                                 is SF request in the ring
                        RR                 N/A
                        NR                 N/A
```

Initial state	New request	New state
G (Switching - MS)	LP	B (Pass-trough)
	FS	B (Pass-trough)
	SF	B (Pass-trough)
	MS	G (Switching - MS) - release the switches but signal MS
	WTR	N/A - cannot happen when there is MS request in the ring
	EXER	N/A - cannot happen when there is MS request in the ring
	RR	N/A
	NR	N/A

Initial state	New request	New state
H (Switching - WTR)	LP	B (Pass-trough)
	FS	B (Pass-trough)
	SF	B (Pass-trough)
	MS	B (Pass-trough)
	WTR	N/A
	EXER	N/A - cannot happen when there is WTR request in the ring
	RR	N/A
	NR	N/A

Initial state	New request	New state
I (Switching - EXER)	LP	B (Pass-trough)
	FS	B (Pass-trough)
	SF	B (Pass-trough)
	MS	B (Pass-trough)
	WTR	N/A
	EXER	I (Switching - EXER)
	RR	N/A
	NR	N/A

5.3. RPS and PSC Comparison on Ring Topology

This section provides comparison between RPS and PSC [RFC6378] [RFC6974] on ring topologies. This can be helpful to explain the reason of defining a new protocol for ring protection switching.

The PSC protocol [RFC6378] is designed for point-to-point LSPs, on which the protection switching can only be performed on one or both of the end points of the LSP. While RPS is designed for ring

tunnels, which consist of multiple ring nodes, and the failure could happen on any segment of the ring, thus RPS SHOULD be capable of identifying and handling the different failures on the ring, and coordinating the protection switching behavior of all the nodes on the ring. As specified in section 5, this is achieved with the introduction of the "Pass-Through" state for the ring nodes, and the location of the protection request is identified via the Node IDs in the RPS Request message.

Taking a ring topology with N nodes as example:

With the mechanism specified in RFC6974, on every ring-node, a linear protection configuration has to be provisioned with every other node in the ring, i.e. with (N-1) other nodes. This means that on every ring node there will be (N-1) instances of the PSC protocol. And in order to detect faults and to transport the PSC message, each instance shall have a MEP on the working path and a MEP on the protection path respectively. This means that every node on the ring needs to be configured with (N-1) * 2 MEPs.

With the mechanism defined in this document, on every ring node there will only be a single instance of the RPS protocol. In order to detect faults and to transport the RPS message, each node only needs to have a MEP on the section to its adjacent nodes respectively. In this way, every ring-node only needs to be configured with 2 MEPs.

As shown in the above example, RPS is designed for ring topologies and can achieve ring protection efficiently with minimum protection instances and OAM entities, which meets the requirements on topology specific recovery mechanisms as specified in [RFC5654].

6. IANA Considerations

IANA is requested to administer the assignment of new values defined in this document and summarized in this section.

6.1. G-ACh Channel Type

The Channel Types for the Generic Associated Channel (GACH) are allocated from the IANA PW Associated Channel Type registry defined in [RFC4446] and updated by [RFC5586].

IANA is requested to allocate a new GACH Channel Type as follows:

Value	Description	Reference
TBD	Ring Protection Switching Protocol (RPS)	this document

6.2. RSP Request Codes

IANA is requested to create a new sub-registry under the "Multiprotocol Label Switching (MPLS) Operations, Administration, and Management (OAM) Parameters" registry called the "MPLS RPS Request Code Registry". All code points within this registry shall be allocated according to the "Standards Action" procedure as specified in [RFC5226].

The RPS Request Field is 8 bits, the allocated values are as follows:

Value	Description	Reference
0	No Request (NR)	this document
1	Reverse Request (RR)	this document
2	not assigned	
3	Exercise (EXER)	this document
4	not assigned	
5	Wait-To-Restore (WTR)	this document
6	Manual Switch (MS)	this document
7-10	not assigned	
11	Signal Fail (SF)	this document
12	not assigned	
13	Forced Switch (FS)	this document
14	not assigned	
15	Lockout of Protection (LP)	this document
16-255	not assigned	

7. Security Considerations

The RPS protocol defined in this document is carried in the G-ACh [RFC5586], which is a generalization of the Associated Channel defined in [RFC4385]. The security considerations specified in these documents apply to the proposed RPS mechanism.

8. Contributing Authors

Wen Ye, Minxue Wang, Sheng Liu (China Mobile)

Guanghui Sun (Huawei)

9. Acknowledgements

The authors would like to thank Gregory Mirsky, Yimin Shen, Eric Osborne and Spencer Jackson for their valuable comments and suggestions.

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, <http://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, <http://www.rfc-editor.org/info/rfc3031>.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <http://www.rfc-editor.org/info/rfc4385>.

[RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, DOI 10.17487/RFC4446, April 2006, <http://www.rfc-editor.org/info/rfc4446>.

[RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <http://www.rfc-editor.org/info/rfc5586>.

[RFC5654] Niven-Jenkins, B., Ed., Brungard, D., Ed., Betts, M., Ed., Sprecher, N., and S. Ueno, "Requirements of an MPLS Transport Profile", RFC 5654, DOI 10.17487/RFC5654, September 2009, <http://www.rfc-editor.org/info/rfc5654>.

[RFC6371] Busi, I., Ed. and D. Allan, Ed., "Operations, Administration, and Maintenance Framework for MPLS-Based Transport Networks", RFC 6371, DOI 10.17487/RFC6371, September 2011, <http://www.rfc-editor.org/info/rfc6371>.

10.2. Informative References

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6378] Weingarten, Y., Ed., Bryant, S., Osborne, E., Sprecher, N., and A. Fulignoli, Ed., "MPLS Transport Profile (MPLS-TP) Linear Protection", RFC 6378, DOI 10.17487/RFC6378, October 2011, <http://www.rfc-editor.org/info/rfc6378>.

[RFC6974] Weingarten, Y., Bryant, S., Ceccarelli, D., Caviglia, D., Fondelli, F., Corsi, M., Wu, B., and X. Dai, "Applicability of MPLS Transport Profile for Ring Topologies", RFC 6974, DOI 10.17487/RFC6974, July 2013, <http://www.rfc-editor.org/info/rfc6974>.

Authors' Addresses

Weiqiang Cheng
China Mobile

Email: chengweiqiang@chinamobile.com

Lei Wang
China Mobile

Email: wangleiyj@chinamobile.com

Han Li
China Mobile

Email: lihan@chinamobile.com

Huub van Helvoort
Hai Gaoming BV

Email: huubatwork@gmail.com

Kai Liu
Huawei Technologies

Email: alex.liukai@huawei.com

Jie Dong
Huawei Technologies

Email: jie.dong@huawei.com

Jia He
Huawei Technologies

Email: hejia@huawei.com

Fang Li
China Academy of Telecommunication Research, MIIT., China

Email: lifang@ritt.cn

Jian Yang
ZTE Corporation P.R.China

Email: yang.jian90@zte.com.cn

Junfang Wang
Fiberhome Telecommunication Technologies Co., LTD.

Email: wjf@fiberhome.com.cn

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: September 10, 2015

M. Dai, Ed.
Y. Rekhter
E. Aries
Facebook
Muhammad Nauman Chaudhry
Verizon Communications

March 9, 2015

MPLS RSVP-TE MBB Label Reuse
draft-dai-mpls-rsvp-te-mbb-label-reuse-00

Abstract

The concept of "make-before-break (MBB)" while rerouting MPLS RSVP-TE tunnels is discussed in [RFC3209]. In the procedure that is outlined, the behavior of downstream label assignment for the new LSP (new tunnel instance) is not well defined. As a general practice, a different label is assigned by each downstream router and advertised to the upstream router in the RESV message for the new LSP; this results in a separate end-to-end data-plane path for the new LSP (with the exception of PHP LSPs or UHP LSP with explicit label on the last hop). This practice allows independent end to end LSP path data-plane verification for each tunnel instance. The consequence of this practice is that the label entry gets added/deleted in the LFIB at every non-ingress router along the LSP path during MBB. Also, the ingress router would need to update all the applications using this LSP when switching to the new tunnel instance, as the new tunnel instance uses different outgoing label. This in turn may also cause other elements of the network which are dependent on the LSP to do the update.

Such network churn can be avoided/minimized if the same label can be re-used (kept intact) wherever it is affecting neither the routing functionalities nor the data path verification of each instance. In addition, whenever label is reused, the setup time for the new tunnel instance would be faster because there is no need for the transit routers along the path of the new LSP to wait for the new LFIB entry to be added. This document proposes a set of procedures to facilitate label reuse when there is a total or partial path overlap between the two tunnel instances during MBB.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Copyright (c) 2015 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

MPLS RSVP-TE make-before-break (MBB) procedure is defined in [RFC3209]. The behavior of downstream label assignment for the new LSP (new tunnel instance) is not well-defined in this procedure. In most MBB implementations, a different label is assigned by each downstream router and advertised to upstream router in the RESV message for the new Label Switched Path (LSP). This means a separate end-to-end data-plane path for the new tunnel instance (with the exception of PHP LSPs or UHP LSPs with explicit NULL label at the last hop). Although this allows for independent end-to-end path verification for each tunnel instance, it requires an LFIB entry add/delete at every non-ingress router along the path of the LSP during MBB even if the paths for the new tunnel instance and the old tunnel instance might be partially or totally overlapping. Label reuse under partial or total overlap condition reduces unnecessary LFIB update, reduces the possibility of errors and improves network convergence latency. In cases where there is a total overlap of paths between the two tunnel instances and the label is reused at each hop along the overlapping path, the necessity of data plane verification for the new tunnel instance is no longer needed.

1.1. Common LSP MBB triggers

The MBB procedure can be triggered because of a change to any property of the RSVP-TE tunnel. The most common case is a change to the bandwidth requirement, especially with the widely implemented auto-bandwidth feature, which dynamically adjusts the LSP bandwidth based on traffic-monitoring feedback. With CSPF commonly used to compute path to meet the new bandwidth requirements, it is possible that the existing path is still one of the best paths which can satisfy the new requirements. This provides the opportunity to reuse labels to maximize the benefits described. If given the choice and the goal of selecting the best path is not the highest priority, CSPF can also prefer the existing path to other possible paths to take full advantage of the label reuse as long as the requirements are still met by the existing path.

2. Recommended conditions for label reuse

The notion of "Label reuse" can be applied for both point-to-point (P2P) LSP and point-to-multipoint (P2MP) LSP, but due to the complexity of P2MP and many possible variations of the solutions, this document will only focus on the recommendations for P2P LSPs.

Labels can be reused when the primary paths of the two tunnel instances have complete overlap starting from a certain point in the paths and going all the way to the egress router of the LSP. The best case scenario is complete overlap of the two paths end to end; in which case there is no need for any label changes and LFIB updates, both in the transit as well as in the ingress routers. In this scenario there is also no need to perform data plane verification for the new tunnel instance. For the case where the two paths overlaps only from a certain transit router (rather than from the ingress), label reuse starts at that router and continues all the way to the egress router. In this case the existing data plane verification method can still be used to verify new tunnel instance as before. Data traversing on either instance will take a different label path from the ingress to this transit router and from then on the traffic will merge into the shared label switched path towards the egress router.

The conditions under which label reuse can be applied are as following:

- o Egress router of LSP: Reuse-label functionality can always be applied.

- o Transit routers of the LSP: For any given transit router of P2P LSP, label can be reused if the following conditions are met:

  (a) Downstream label received is the same

  (b) NHOP is the same

- o Ingress router of the LSP: When the same conditions as listed under transit router are met, instead of no label change, there is no need for ingress route update for LSP to applications depending on it.

The label reuse procedure starts from the egress of the LSP as RESV traverses upstream towards the ingress of the LSP; it terminates at the first transit router where paths of the two tunnel instances diverge towards the ingress of the LSP or at the transit router which doesn’t support label reuse.

3. Control of label-reuse behavior

3.1. Enable/Disable label-reuse capability

This document recommends enabling "label-reuse" capability by default. Allow it to be disabled if needed by changing configuration.

3.2. Prefer overlapping path to facilitate label-reuse

In order to take full advantage of the label-reuse capability, path computation for the new tunnel instance may seek to maximize path overlap. This can be achieved through two approaches.

- o The first approach is to select from the best paths available the path which has the most path overlap with the existing path starting from the egress router.

- o The second approach is to prefer the existing path if it still satisfies the new requirement, even though it might not be the best path.

The choice between the approaches is a matter of local computation policy and can be different for different types of MBB trigger.

4. IANA Considerations

This document makes no request for IANA action.

5. Security Considerations

This document does not introduce new security issues.

6. Acknowledgements

The authors wish to thank Vishnu Pavan Beeram for his input, feedback, and helpful suggestions.

7. Normative References

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

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

Authors' Addresses

Minjie Dai (editor)
Juniper Networks
Email: mdai@juniper.net

Yakov Richter
Juniper Networks
Email: yakov@juniper.net

Ebben Aries
Facebook
1 Hacker Way
Menlo Park, CA 94025
US
Email: exa@fb.com Muhammad Nauman Chaudhry Verizon Communications Email
: muhammd.n.chaudhry@verizon.com

Raveendra Torvi
Juniper Networks
Email: rtorvi@juniper.net

Markus Work
Juniper Networks
Email: mjork@juniper.net

Yimin Shen
Juniper Networks
Email: yshen@juniper.net

Natrajan Venkatraman
Juniper Networks
Email: natv@juniper.net

Harish Sitaraman
Juniper Networks
Email: hsitaraman@juniper.net

INTERNET-DRAFT
Intended Status: Informational
Expires: July 26, 2016

Luyuan Fang
Deepak Bansal
Microsoft
Fabio Chiussi

Chandra Ramachandran
Juniper Networks
Ebben Aries
Facebook
Shahram Davari
Broadcom
Barak Gafni
Mellanox
Daniel Voyer
Bell Canada
Nabil Bitar
Verizon

January 23, 2016

MPLS-Based Hierarchical SDN for Hyper-Scale DC/Cloud
draft-fang-mpls-hsdn-for-hsdc-05

Abstract

This document describes Hierarchical SDN (HSDN), an architectural solution to scale the Data Center (DC) and Data Center Interconnect (DCI) networks to support tens of millions of physical underlay endpoints, while efficiently handling both Equal Cost Multi Path (ECMP) load-balanced traffic and any-to-any end-to-end Traffic Engineered (TE) traffic. HSDN achieves massive scale using surprisingly small forwarding tables in the network nodes. HSDN introduces a new paradigm for both forwarding and control planes, in that all paths in the network are pre-established in the forwarding tables and the labels can identify entire paths rather than simply destinations. The HSDN forwarding architecture is based on four main concepts: 1. Dividing the DC and DCI in a hierarchically-partitioned structure; 2. Assigning groups of Underlay Border Nodes in charge of forwarding within each partition; 3. Constructing HSDN MPLS label stacks to identify endpoints and paths according to the HSDN structure; and 4. Forwarding using the HSDN MPLS labels.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering

Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright Notice

Copyright (c) 2016 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

With the growth in the demand for cloud services, the end-to-end cloud network, which includes Data Center (DC) and Data Center Interconnect (DCI) networks, has to scale to support millions to tens of millions of underlay network endpoints. These endpoints can be bare-metal servers, virtualized servers, or physical and virtualized network functions and appliances.

The scalability challenge is twofold: 1. Being able to scale using low-cost network nodes while achieving high resource utilization in the network; and 2. Being able to scale at low operational and computational complexity while supporting both Equal-Cost Multi-Path (ECMP) load-balanced traffic and any-to-any Traffic Engineering (TE) traffic.

Being able to scale at low cost requires to avoid the potential explosion of the routing tables in the network nodes as the number of underlay network endpoints increases. Current commodity switches have relatively small routing and forwarding tables. For example, the typical Forwarding Information Base (FIBs) and Label Forwarding Information Base (LFIBs) tables in current low-cost network nodes contain 16K or 32K entries. These small sizes are clearly insufficient to support entries for all the endpoints in the hyper-scale cloud. Address aggregation is used to ameliorate the problem, but the scalability challenges remain, since the dynamic and elastic environment in the DC/cloud often brings the need to handle finely granular prefixes in the network in order to support Virtual Machine (VM) and Virtualized Network Function (VNF) mobility.

Other factors contribute to the FIB/LFIB explosion. For example, in a typical DC using a fat Clos topology, even the support of ECMP load balancing may become an issue if the individual outgoing paths belonging to an ECMP group carry different outgoing labels, since a single destination may contribute multiple entries in the tables.

Another key scalability issue to resolve is the complexity of certain desired functions that should be supported in the network, the most prominent one being TE. Currently, any-to-any server-to-server TE in the DC/DCI is simply unfeasible, as path computation and bandwidth allocation at scale, an NP-complete problem, becomes rapidly unmanageable. Furthermore, the forwarding state needed in the network nodes for TE tunnels contributes in a major way to the explosion of the LFIBs, since each TE tunnel corresponds to an entry in the tables.

Other major scalability issues are related to the efficient creation, management, and use of tunnels, for example the configuration of

protection paths for fast restoration.

Many additional scalability issues in terms of operational and computational complexity need to be resolved in order to scale the control plane and the network state. In particular, the controller-centric approach of Software Defined Networks (SDNs), which is increasingly accepted as "the way to build the next generation clouds," still needs to be demonstrated to be scalable to the levels required in the hyper-scale DC and cloud.

Finally, the underlay network architecture should offer certain capabilities to facilitate the support of the demands of the overlay network.

In this document, we present Hierarchical SDN (HSDN), a set of solutions for all these scalability challenges in the underlay network, both in the forwarding and in the control plane.

Although HSDN can be used with any forwarding technology, including IPv4 and IPv6, it has been designed to leverage Multi Protocol Label Switching (MPLS)-based forwarding [RFC3031], using label stacks [RFC3032] constructed according to the HSDN structure. This document therefore describes MPLS-based HSDN. Here, we describe end-to-end (host-to-host) MPLS-based HSDN, where the entire HSDN label stacks from source to destination are imposed at the server’s Network Interface Cards (NICs), and thus all the IP lookups are confined to the network edges. However, MPLS-based HSDN does not need to be end-to-end, since label imposition could happen instead at the network nodes (e.g., at the Top-of-Rack (ToR) switches), or intermediate lookups in the network could be introduced, or even a combination of MPLS and IP forwarding could be deployed as part of the HSDN network.

The HSDN underlay network is suited to support any Layer 2 or Layer 3 virtualized overlay network technology. In this document, we assume a MPLS-based overlay technology using a Virtual Network (VN) Label, which is encapsulated in the HSDN label stack. However the description can be easily generalized to any overlay technology, such as BGP/MPLS IP VPNs [RFC4364], EVPN [RFC7432], VXLAN [RFC7348], NVGRE [RFC7637], Geneve [I-D.draft-gross-geneve], and other technologies.

HSDN achieves massive scale using surprisingly small LFIBs in the network nodes, while supporting both ECMP load-balanced traffic and any-to-any end-to-end TE traffic [HSDNSOSR15]. HSDN also brings important simplifications in the control plane and in the architecture of the SDN controller.

The HSDN architecture and operation is characterized by two fundamental properties. First, all paths in the network are pre-

established in the forwarding tables. Second, the HSDN labels can identify entire paths or groups of paths rather than simply destinations.

These two properties radically simplify establishing and handling tunnels. In addition to optimally handling both ECMP and Non-Equal Cost Multi Path load balancing, HSDN enables any-to-any, end-to-end, server-to-server TE at scale. With HSDN, the "cost" of establishing a tunnel is essentially eliminated, since the "tunnels" are pre-established in the network, and the TE task becomes one of path assignment and bandwidth allocation to the flows. As a larger portion of the traffic can be engineered effectively, the network can be run at a higher utilization using comparatively smaller buffers at the nodes.

The HSDN forwarding architecture in the underlay network is based on four main concepts: 1. Dividing the DC and DCI in a hierarchically-partitioned structure; 2. Assigning groups of Underlay Border Nodes in charge of forwarding within each partition; 3. Constructing HSDN MPLS label stacks to identify the end points according to the HSDN structure; and 4. Forwarding using the HSDN MPLS labels.

HSDN is designed to allow the physical decoupling of control and forwarding, and have the LFIBs configured by a controller according to a full SDN approach. The controller-centric approach is described in this document. In this context, "MPLS forwarding" in HSDN simply means using MPLS labels to forward the packets, since there is no need for label distribution protocols.

However, the HSDN control plane can also be built using a hybrid approach, in which a routing or label distribution protocol is used to distribute the labels, in conjunction with a SDN controller. This hybrid approach may be particularly useful during technology migration. The use of BGP Labeled Unicast (BGP-LU) for label distribution and LFIB configuration in a HSDN architecture is described in [I-D.fang-idr-bgplu-for-hsdn].

1.1. Terminology

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].

Term	Definition
BGP	Border Gateway Protocol
BGP-LU	Border Gateway Protocol Labeled Unicast
DC	Data Center

DCGW	DC Gateway (Border Leaf)
DCI	Data Center Interconnect
DID	Destination Identifier
ECMP	Equal Cost MultiPathing
FIB	Forwarding Information Base
HSDN	Hierarchical SDN
LDP	Label Distribution Protocol
LFIB	Label Forwarding Information Base
LN	Leaf Node
MPLS	Multi-Protocol Label Switching
NIC	Network Interface Card
PID	Path Identifier
SDN	Software Defined Network
SN	Spine Node
SVR	Server
UP	Underlay Partition
UPBG	Underlay Partition Border Group
UPBN	Underlay Partition Border Node
TE	Traffic Engineering
ToR	Top-of-Rack switch
TR	Top-of-Rack switch (used in figures)
VN	Virtual Network
VM	Virtual Machine
VNF	Virtualized Network Function
WAN	Wide Area Network

In this document, we also use the following terms.

- End device: A physical device attached to the DC/DCI network. Examples of end devices include bare metal servers, virtualized servers, network appliances, etc.

- Level: A layer in the hierarchy of underlay partitions in the HSDN architecture.

- Overlay Network (ON): A virtualized network that provides Layer 2 or Layer 3 virtual network services to multiple tenants. It is implemented over the underlay network.

- Path Label (PL): A label used for MPLS-based HSDN forwarding in the underlay network.

- Row: A row of racks where end devices reside in a DC.

- Tier: One of the layers of network nodes in a Clos-based topology.

- Underlay Network (UN): The physical network that provides the connectivity among physical end devices. It provides transport for

the overlay network traffic.

- o Underlay Partition (UP): A logical portion of the underlay network designed according to the HSDN architecture. Underlay partitions are arranged in a hierarchy consisting of multiple levels.

- o VN Label (VL): A label carrying overlay network traffic. It is encapsulated in the underlay network in a stack of path labels constructed according to the HSDN forwarding scheme.

1.2. DC and DCI Reference Model

Here we show the typical structure of the DC and DCI, which we use in the rest of this document to describe the HSDN architecture. We also introduce a few commonly used terms to assist in the explanation.

Figure 1 illustrates multiple DCs interconnected by the DCI/WAN.

```
                  +-------------+
                  |             |
                  |     DC      |
                  |             |
                  +-------------+
                            \-----.
                            (       ')
                        .--(.       '.---.
                       (     '      '     )
                      (       DCI/WAN      )
                       (.                 .)
                        (     (        .)  \
                         /'--' '-''---'  +-------------+
             +-------------+             |             |
             |             |             |     DC      |
             |     DC      |             |             |
             |             |             +-------------+
             +-------------+
```

Figure 1. DCIWAN interconnecting multiple DCs.

Figure 2 below illustrates the typical structure of a Clos-based DC fabric.

```
                                  +----+   +----+
  DCGW                  +---------+ GW +   + GW +---------+
                        |         ++--++\ /++--++         |
                        | +--------|--|--\--\  |          |
                        | |        |  \-/-\-|--|--------+ |
                      +-+-++     +-+--+/   \+--+-+     ++-+-+
  Spine    +---------+ SN +     | SN |     | SN +     + SN +---------+
           |         +----+\    ++--++     ++--++\   /+----+         |
           | +---------------\---+  |       |  +---/---------------+ |
           | |        ...      \    |       |    /      ...        | |
          ++-+-+ +----+  +----+ ++--++     ++--++ +----+  +----+ +-+-++
  Leaf    | LN | | LN |  | LN | | LN |     ||LN | | LN |  | LN | | LN |
          ++--++ ++--++  +---++ ++---+     +---++ ++--++  +---++ ++--++
           |   \ /   |    |   \ /   |       |   \ /   |    |   \ /   |
           |   / \   |    |   / \   |       |   / \   |    |   / \   |
           | / ... \ |    | / ... \ |       | / ... \ |    | / ... \ |
         ++++ +--+ ++++ ++++ +--+ ++++    ++++ +--+ ++++ ++++ +--+ ++++
  ToR    |TR| |TR| |TR| |TR| |TR| |TR|    |TR| |TR| |TR| |TR| |TR| |TR|
         +--+ +--+ +--+ +--+ +--+ +--+    +--+ +--+ +--+ +--+ +--+ +--+
          |    |    |    |    |    |       |    |    |    |    |    |
         +--+ +--+ +--+ +--+ +--+ +--+    +--+ +--+ +--+ +--+ +--+ +--+
  Server +--+ +--+ +--+ +--+ +--+ +--+    +--+ +--+ +--+ +--+ +--+ +--+
  rack   +--+ +--+ +--+ +--+ +--+ +--+    +--+ +--+ +--+ +--+ +--+ +--+
         +--+ +--+ +--+ +--+ +--+ +--+    +--+ +--+ +--+ +--+ +--+ +--+
```

Figure 2. Typical Clos-based DC fabric topology.

Note: Not all nodes and links are shown in Figure 2.

The DC fabric shown in Figure 2 uses what is known as a spine and leaf architecture with a multi-stage Clos-based topology interconnecting multiple tiers of network nodes. The DC Gateways (DCGWs) connect the DC to the DCI/WAN. The DCGW connect to the Spine Nodes (SNs), which in turn connect to the Leaf Nodes (LFs). The Leaf Nodes connect to the ToRs. Each ToR typically resides in a rack (hence the name) accommodating a number of servers connected to their respective ToR. The servers may be bare metal or virtualized.

Each tier of switches and the connectivity between switches is designed to offer a desired capacity and provide sufficient bandwidth to the servers and end devices.

Figure 2 is not meant to represent the precise topology of the DC. In fact, the precise topology and connectivity between the tiers of switches depends on the specific design of the DC. More or less tiers of switches (spines or leafs) or asymmetric topologies, not shown in the figure, may be used. A precise description of the possible topologies and related design criteria is out of the scope of this

document.

What is relevant for this document is the fact that a typical large-scale DC topology does not have all the tiers fully connected to the adjacent tier (i.e., not all network nodes in a tier are necessarily connected to all network nodes in the adjacent tiers). This is especially true for the tiers closer to the endpoints, and is due to the sheer number of connections and devices (in other words, in a large, fat Clos there are too many network nodes in some tiers for all network nodes to connect to one another), and to the physical constraints of the DC (i.e., the network nodes may be located physically apart in separate rooms or buildings, and full connectivity may become too costly).

In a typical DC, the racks of servers are physically organized in "clusters" of racks, and dedicated banks of leaf switches may serve the ToRs in each cluster. For example, the racks may be physically placed in rows of racks, and a cluster of racks may correspond to a portion of a row, an entire row, or multiple rows of racks. Indeed, the leaf nodes are sometimes called "middle (or end) of the row switches" because they are physically located in a rack in the middle (or end) of a row of racks of servers. In turn, leaf nodes may also be organized in "zones" (we use "clusters" and "zones" as generic terms, but other terms may be used in the industry to refer to similar concepts), and banks of spines may be assigned to serve each zone. For example, a zone may include all the banks of leaf nodes that are in a room or in a building in the DC.

The actual connectivity is typically organized following an aggregation/multiplexing connectivity architecture that consolidates traffic from the edges into the leafs and spines, while allowing for over-subscription in order to strike a reasonable trade-off between cost and available capacity. The connectivity between each tier may use some form of shuffle-exchange topology that attempts to "mix" the available paths while taking in account the physical constraints.

The key observation is that it is impractical, uneconomical, and ultimately unnecessary to use a fully connected Clos-based topology in a large scale DC. Because of the physical constraints, the topology of a large DC is not a flat, fully-connected Clos, but rather has a dertain hierarchy. The HSDN architecture recognizes this fact, and uses it to dramatically simplify forwarding and control planes using an approach that is also hierarchical.

2. Requirements

2.1. MPLS-Based HSDN Design Requirements

The following are the key design requirements for MPLS-based HSDN solutions.

1) MUST support millions to tens of millions of underlay network endpoints in the DC/DCI.

2) MUST use very small LFIB sizes (e.g., 16K or 32K LFIB entries) in all network nodes.

3) MUST support both ECMP load-balanced traffic and any-to-any, end-to-end, server-to-server TE traffic.

4) MUST support ECMP traffic load balancing using a single forwarding entry in the LFIBs per ECMP group.

5) MUST require IP lookup only at the network edges.

6) MUST support encapsulation of overlay network traffic, and support any network virtualization overlay technology.

7) MUST support control plane using both full SDN controller approach, and traditional distributed control plane approach using any label distribution protocols.

2.2. Hardware Requirements

The following are the hardware requirements to support HSDN.

1) The server NICs MUST be able to push a HSDN label stack consisting of as many path labels as levels in the HSDN hierarchical partition (e.g., 3 path labels).

2) The network nodes MUST support MPLS forwarding.

3) The network nodes MUST be able perform ECMP load balancing on packets carrying a label stack consisting of as many path labels as levels in the HSDN hierarchical partition, plus one or more VN label/header for the overlay network (e.g., 3 path labels + 1 VN label/header). For example, if the hash function used for ECMP forwarding is based on the IP 5-tuple, as is often the case, this requirement implies that the network nodes MUST be able to lookup the 5-tuple inside up to four labels.

3. HSDN Architecture - Forwarding Plane

As mentioned above, a primary design requirement for HSDN is to enable scalability of the forwarding plane to tens of millions of network endpoints using very small LFIB sizes in all network nodes in

the DC/DCI, while supporting both ECMP and any-to-any server-to-server TE traffic.

The driving principle of the HSDN forwarding plane is "divide and conquer" by partitioning the forwarding task into local and independent forwarding. When designed properly, such an approach enables extreme horizontal scaling of the DC/DCI.

HSDN is based on four concepts:

1) Dividing the underlay network in a hierarchy of partitions;
2) Assigning groups of Underlay Partition Border Nodes (UPBN) to each partition, in charge of forwarding within the corresponding partition;
3) Constructing HSDN label stacks for the endpoint Forward Equivalency Classes (FECs) in accordance with the underlay network partition hierarchy;
4) Configuring the LFIBs in all network nodes and forwarding using the label stacks.

As explained in Section 3.3.1, the HSDN label stacks can be used to identify entire paths to each endpoint, rather than simply the destination endpoint itself. As a matter of fact, the HSDN solution is meant to be configured with all possible paths in the network pre-established in the LFIBs in the network nodes. In this case, a FEC per path to each endpoint is defined. However, because of the way the HSDN architecture is designed, the required local number of entries in the LFIB of each network node remains surprisingly small.

In this section, we explain in detail each of these concepts. Scalability analysis for both ECMP load-balanced and TE traffic is presented in Section 4. In Section 5, we describe a possible label stack assignment scheme for HSDN.

3.1. Hierarchical Underlay Partitioning

HSDN is based on dividing the DC/DCI underlay network into logical partitions arranged in a multi-level hierarchy.

The HSDN hierarchical partitioning is illustrated in Figure 3.

```
            +-------------------------------------------------------------+
  UP        |                           UP0                               |
  Level 0   |                                                             |
            |+---------------------------------+   +------------------+|
            ||                                 |...|                  ||
            +|---------------------------------|---|------------------|+
  UP         |             UP1-0               |   |       UP1-N      |
  Level 1    |                                 |   |                  |
             |+-------------+   +-------------+|   |   +-------------+|
             ||             |...|             ||   |...|             ||
             +|-------------|---|-------------|+   +---|-------------|+
  UP          |   UP2-0-0   |   |   UP2-0-N   |        |  UP2-N-N    |
  Level 2     |             |   |             |        |             |
              +-----+ +-----+   +-----+ +-----+        +-----+ +-----+
              | SVR | | SVR |   | SVR | | SVR |        | SVR | | SVR |
              |-----|-|-----|   |-----|-|-----|        |-----|-|-----|
  Overlay     |VM|VM| |VM|VM|   |VM|VM| |     |        |VM|VM| |     |
  Level       |-----| |-----|   |-----| |     |        |-----| |     |
              |VM|VM| |VM|VM|   |VM|VM| |     |        |VM|VM| |     |
              +-----+ +-----+   +-----+ +-----+        +-----+ +-----+
```

Figure 3. HSDN underlay network hierarchical partitioning of DC/DCI.

The hierarchy consists of multiple levels of Underlay Partitions (UPs). For simplicity, we describe HSDN using three levels of partitioning, but more or less levels can be used, depending on the size and architecture of the overall network, using similar design principles (as shown in Section 4, three levels of partitions are sufficient to achieve scalability to tens of millions servers using very small LFIBs).

The levels of partitions are nested into a hierarchical structure. At each level, the combination of all partitions covers the entire DC/DCI topology. In general, within each level, the UPs do not overlap, although there may be design scenarios in which overlapping UPs within a level may be used. The top level (Level 0) consists of a single underlay partition UP0 (the HSDN concept can be extended to multi-partitioned Level 0).

We use the following naming convention for the UPs:

- Partitions at Level i are referred to as UPi (e.g., UP0 for Level 0, UP1 for Level 1, UP2 for Level2, and so on).

- Within each level, partitions are identified by a rightmost sequential number (starting from 1) referring to the corresponding level and a set of sequential number(s) for each partition in a

higher level that the specific partition is nested into.

For example, at Level 1, there are N partitions, referred to as UP1-1 to UP1-N.

Similarly, at Level 2, there are M partitions for each Level 1 partitions, for a total of NxM partitions. For example, the Level 2 partitions nested into Level 1 partition UP1-1 are UP2-1-1 to UP2-1-M, while the ones nested into UP1-N are UP2-N-1 to UP2-N-M.

- Note that for simplicity in illustrating the partitioning, we assume a symmetrical arrangement of the partitions, where the number of partitions nested into each partition at a higher level is the same (e.g., all UP1 partitions have M UP2 partitions). In practice, this is rarely the case, and the naming convention can be adapted accordingly for different numbers of partitions nesting into each higher level partition (e.g., partition UP1-1 has M1 UP2 partitions, partition UP1-2 has M2 UP2 partitions, and so on).

The following considerations complete the description of Figure 3.

o The servers (bare metal or virtualized) are attached to the bottom UP level (in our case, Level 2). A similar naming convention as the one used for the partitions may be used.

o In Figure 3, we also show an additional Overlay Level. This corresponds to the virtualized overlay network (if any) providing Virtual Networks (VN) connecting Virtual Machines (VMs) and other overlay network endpoints. Overlay network traffic is encapsulated by the HSDN underlay network. As mentioned in the Introduction, the HSDN underlay network is suited to support any Layer 2 or Layer 3 virtualized overlay network technology, such as BGP/MPLS IP VPNs [RFC4364], EVPN [RFC7432], VXLAN [RFC7348], NVGRE [RFC7637], Geneve [I-D.draft-gross-geneve], and other technologies. A full description of the encapsulation of these technologies into the HSDN underlay label stack is out of scope of this document and will be addressed in a separate document.

The UPs are designed to contain one or more tiers of switches in the DC topology or nodes in the DCI. The key design criteria in defining the partitions at each level is that they need to follow the "natural" connectivity implemented in the DC/DCI topology. An example is given in Section 3.2.3 to further clarify how the partitions are designed.

3.2. Underlay Partition Border Nodes

Once the HSDN hierarchical partitioning is defined, Underlay

Partition Border Nodes (UPBNs) are assigned to each UP. This is illustrated in Figure 4.

```
  ++++++++++++++++++++++++++++++++++++++++++++++++++
  |                     UP0              <--------|-----+
  |                                               |     |
  | ++++++++++++++++++++++++++++++++++++++++++++ |     |
  | |   +-----------------------------------+   | |     |
  | |   | +---------+  UPBG1-i  +---------+ |   | |     |
  | |   | | UPBN1-i |    ...    | UPBN1-i | |   | |     |
  | |   | +---------+           +---------+ |   | |     |
  | |   +-----------------------------------+   | |     |
  ++|++++++++++++++++++++++++++++++++++++++++++|++     |
    |                     UP1              <------|----+  |   +---------+
    | ++++++++++++++++++++++++++++++++++++++++ |    |  +---|   PL0   |
    | | +-----------------------------------+ | |    |      +---------+
    | | | +---------+ UPBG2-i-j +---------+ | | |    +------|   PL1   |
    | | | |UPBN2-i-j|    ...    |UPBN2-i-j| | | |           +---------+
    | | | +---------+           +---------+ | | |    +------|   PL2   |
    | | +-----------------------------------+ | |    |      +---------+
    ++|++++++++++++++++++++++++++++++++++++++++|++    |  +---|   VL    |
      |                     UP2              <----|------+  |   +---------+
      |          +---------------+            |         |
      |          |   SVR-i-j-k   |            |         |
      |          | +-----------+ |            |         |
      |          | |    NVE    |<-------------|---------+
      |          | +-----+-----+ |            |
      ++++++++++++| | VM  | VM  | |+++++++++++++
                 | +-----+-----+ |
                 | | VM  | VM  | |
                 +-+-----+-----+-+
```

Figure 4. UBPNs, UBPGs, and label stack assignment.

The UPBNs serve as the connecting nodes between adjacent partitions. As such, the UPBNs belong to two partitions in adjacent levels in the hierarchy and they constitute the entry points for traffic from the higher level partition destined to the corresponding lower level partition (and vice-versa, they are the exit points for traffic from a lower level partition to a higher level partition). As such, they constitute the forwarding end destinations within each partition.

In order to provide sufficient capacity and support traffic load balancing between the levels in the hierarchy, multiple UPBNs are assigned to each partition. The UPBNs for each partition are grouped into an Underlay Partition Border Group (UPBG). As shown in Section 5, using an appropriate Label Stack Assignment scheme all UPBNs in a

UPBG can be made identical for ECMP traffic forwarding (i.e., the ECMP entries in the LFIBs in all UPBNs in a UPBG are identical). Thus, for ECMP traffic load balancing, all UPBNs belong to the same FEC as far as the higher level partition is concerned. For TE traffic, a desired UPBN within a UPBG group may need to be specified, and thus the UPBNs in a UPBG are not forwarding-wise equivalent.

In practice, the UPs are designed by finding the most advantageous way to partition the DC Clos-based topology and the DCI topology. As mentioned above, the connectivity of any large-scale DC is not fully flat, but rather contains some sort of hierarchical organization. Recognizing the hierarchy of the physical connectivity is an important starting point in the design of the partitions.

Within the DC, the UPBNs in each level are subsets of the network nodes in one of the tiers that form the multi-stage Clos architecture.

In general, in addition to the UPBNs, the UPs may internally contain tiers of network nodes that are not UPBNs. A specific design example to further illustrate the HSDN partitioning is provided in Section 3.2.3.

As explained in more detail in Section 3.3, for forwarding purposes, by partitioning the DC/DCI in this manner and using HSDN forwarding, the UPBNs need to have entries in their LFIBs only to reach destinations in the two partitions to which they belong to (i.e., their own corresponding lower-level partition and the higher-level partition to which they nested to). The network nodes inside the UPs only need to have entries in their LFIB to reach the destinations in their partition.

Similarly, in order to establish all possible paths in the entire network, the UPBNs need to have entries in their LFIBs only for all possible paths to the destinations in the two partitions to which they belong to.

From these considerations, a first design heuristic for choosing the partitioning structure is to keep the number of partitions nested at each level into the higher level relatively small for all levels. For the lowest level, the number of endpoints (servers) in each partition should also be kept to manageable levels.

Clearly, the design tradeoff is between the size and the number of partitions at each level. Although finding the optimal design choice may require a little trial-and-error computation of different options,fortunately, for most practical deployments, it is relatively simple to find a good tradeoff that achieves the desired scalability

to millions or tens of millions of endpoints.

3.2.1. UPBN and UPBG Naming Convention

We use a similar naming convention for the UPBNs and UPBGs as the one used for the UPs:

- UPBNi is a UPBN between partitions at Level(i) and Level(i-1). Similarly for UPBG.

- Within each level, the UPBNs are identified by a set of sequential number(s) equal to the corresponding sequential number(s) of the corresponding partition within that level.

  For example, at Level 1, UPBN1-1 corresponds to partition UP1-1, and connects UP0 with UP1-1. UPBN1-N corresponds to partition UP1-N and connects UP0 with UP1-N, and so on. Similarly for UPBG.

  At Level 2, UPBN2-1-1 corresponds to partition UP2-1-1 and connects UP1-1 with UP2-1-1, and so on. Similarly for UPBG.

  Note that the UPBNs within an UPBGs can be further distinguished using an appropriate naming convention (for example, using an additional sequential number within the UPBG), which for simplicity is not shown here. This more granular naming convention is needed to configure the paths and the TE tunnels.

3.2.2. HSDN Label Stack

In MPLS-Based HSDN, an MPLS label stack is defined and used for forwarding. The key notion in HSDN is that the label stack is defined and the labels are assigned in accordance with the hierarchical partitioned structure defined above.

The label stack, shown in Figure 4 above, is constructed as follows.

- The label stack contains as many Path Labels (PLs) as levels in the partitioning hierarchy.

- Each PL in the label stack is associated to a corresponding level in the partition hierarchy and is used for forwarding at that level.

  In the scenario of Figure 4, PL0 is associated to Level 0 and is used to forward to destinations in UP0, PL1 is associated to Level 1 and is used to forward to destinations in any UP1 partitions, and PL2 is associated to Level 2 and is used to forward to destinations in any UP2 partitions.

- A VN Label (VL) is also shown in the label stack in Figure 4. This label is associated to the Overlay Level and is used to forward in the overlay network. The VL is simply encapsulated in the label stack and transported in the HSDN underlay network. As mentioned above, the HSDN underlay network is suited to support any Layer 2 or Layer 3 virtualized overlay network technology, and thus the VL may be a label, a tag, or some other identifier, depending on the overlay technology used. The details of the VL encapsulation and processing for different overlay technologies are out of scope of this document.

Each endpoint in the DC/DCI is identified by a corresponding label stack. For a given endpoint, the label stack is constructed in such a way that the PLO specifies the UP1 to which the endpoint is attached to, the PL1 specifies the UP2 to which the endpoint is attached to, and the PL2 specifies the FEC in the UP2 corresponding to the endpoint.

The labels in the HSDN label stack can identify entire paths, rather than simply the end destination within the corresponding partition. This can be used to bring dramatic simplifications in handling tunnels and TE traffic in particular, as further explained in Section 3.3.2.

As mentioned above, in this draft we describe end-to-end MPLS-based HSDN forwarding, where the entire HSDN label stacks from the sources to the destinations are inserted at the server’s NICs. In this scenario, the label stack imposed at a server points all the way to the end destination of the packet, which may be in a different DC. With any-to-any, end-to-end TE, the HSDN label stack identifies the entire path to the destination. For inter-DC traffic, there may be cases where the path through the remote DC would be preferably determined when the packet arrives at that DC, or when the packet leaves the source DC, so it may be desirable that part of the label stack be imposed inside the network rather than at the server. Nothing precludes this design choice, and a lookup may be added where desired in the HSDN network.

A scheme to assign the PL labels in the HSDN label stack is described in Section 5.

3.2.3. HSDN Design Example

We use an example to further explain the HSDN design criteria to define the hierarchically-partitioned structure of the DC/DCI. We use the same design example in the Scalability Analysis section (Section 4) to show the LFIB sizing with ECMP and TE traffic.

To summarize some of the design heuristics for the HSDN underlay partitions:

- The UPs should be designed to follow the "natural" connectivity topology in the DC/DCI.

- The number of partitions at each level nested into the higher level should be relatively small (since they are FEC entries in the LFIBs in the network nodes in the corresponding levels).

- The number of endpoints (servers) in each partition in the lowest level should be relatively small (since they are FEC entries in the LFIBs in the network nodes in the lowest level).

- The number of levels should be kept small (since it corresponds to the number of path labels in the stack).

- The number of tiers in each partition in each level should be kept small. This is due to the multiplicative fanout effect for TE traffic (explained in Section 4.2), which has a major impact on the LFIB size needed to support any-to-any server-to-server TE.

The HSDN forwarding plane design consists in finding the best tradeoff among these conflicting objectives. Although the optimal design choices ultimately depend on the specific deployment, fortunately, it is generally rather straightforward to identify design choices that can support scalability to millions or tens of millions of servers.

Here we describe a design example to illustrate that a three-level HSDN hierarchy is sufficient to scale the DC/DCI to tens of millions of servers.

With three levels, a possible design choice for the UP1s is to have each UP1 correspond to a DC. With this choice, the UP0 corresponds to the DCI and the UPBN1s are the DCGWs in each DC (the UPBG1s group the DCGWs in each DC).

Once the UP1s are chosen this way, a possible design choice for the UP2s is to have each UP2 correspond to a group of racks, where each group of racks may correspond to a portion of a row of racks, an entire row of racks, or multiple rows of racks. The specific best choice of how many racks should be in a group of racks corresponding to each UP2 ultimately depends on the specific connectivity in the DC and the number of servers per racks.

While precise numbers depend on the specific technologies used in each deployment, here and in the Scalability Analysis section

(Section 4) we want to give some ideas of the scaling capabilities of HSDN. For this purpose, we use some hypothetical yet reasonable numbers to characterize the partitioning design example.

Assume the following: a) 20 DCs connected via the DCI/WAN; b) 50 servers per rack; c) 20 racks per group of racks; d) 50 groups of racks per DC.

With these numbers, there are 500K servers per DC, for a total of 10M underlay network endpoints in the DC/DCI.

In the HSDN structure in this example, there are 20 UP1s, 500 UP2s per UP1, and 1000 servers per UP2.

3.3. MPLS-Based HSDN Forwarding

The hierarchically partitioned structure and the corresponding label stack are used in HSDN to scale the forwarding plane horizontally while using LFIBs of surprising small sizes in the network nodes.

As explained above, each label in the HSDN label stack is associated with one of the levels in the hierarchy and is used to forward to destinations in the underlay partitions at that level.

With HSDN, by superimposing a hierarchically-partitioned structure and using a label stack constructed according to such a structure, we are able to impose a forwarding scheme that is aggregated by construction. This translates in dramatic reductions in the size of the LFIBs in the network nodes, since each node only needs to know a limited portion of the forwarding space.

HSDN supports any label assignment scheme to generate the labels in the label stack. However, if a label assignment scheme that is consistent with the HSDN structure is used, additional simplifications of the LFIBs and the control plane can be achieved.

In Section 5 below, we present one example of such a scheme, where the labels in the label stack represent the "physical" location of the endpoint, expressed according to the HSDN structure. For TE traffic, the labels represent a specific path towards the desired destination through the HSDN structure.

In the Scalability Analysis section (Section 4) and in the Control Plane section (Section 6) we assume that such a Label Assignment scheme is used.

In the rest of this section, we describe the life of a packet in the HSDN DC/DCI. We use the specific design example described in Section

3.2.3 above to help in the explanation, but of course the forwarding would be similar for other design choices.

3.3.1 Non-TE Traffic

We first describe the behavior for ECMP load-balanced, non-TE traffic. In the HSDN DC/DCI, for a packet that needs to be forwarded to a specific endpoint in the underlay network, the outer label PL0 specifies which UP1 contains the endpoint. Let’s refer to this UP1 as UP1-a. For ECMP traffic, the PL0 binding is with a FEC corresponding to the UPBG1-a associated with UP1-a. Note that all the endpoints reachable via UP1-a are forwarded using the same FEC entry for Level 0 in the hierarchical partitioning.

Once the packet reaches one of the network nodes UPBN1-a in the UPBG1-a group (the upstream network nodes perform ECMP load balancing, thus the packet may enter UP1-a via any of the UPBN1-a nodes), the PL0 is popped and the PL1 is used for forwarding in the UP1-a (to be precise, because of penultimate hop popping, it is the network node immediately upstream of the chosen UPBN1-a that pops the label P0).

The PL1 is used within UP1-a to reach the UP2 which contains the endpoint. Let’s refer to this UP2 as UP2-a. In the UP2 network nodes the PL1 binding is with a FEC corresponding to the UPBG2-a associated with UP2-a. Similarly as above, note that all the endpoints reachable via UP2-a are forwarded using the same FEC entry for Level 1 in the hierarchical partitioning.

Once the packet reaches one of the network nodes UPBN2-a in the UPBG2-a group (once again, the upstream network nodes perform ECMP load balancing, so the packet may transit to any of the UPBN2-a nodes), the PL1 is popped and the PL2 is used for the rest of the forwarding (again, to be precise, the penultimate network node upstream of UPBN2-a is the one popping the PL1 label).

The PL2 is used within UP2-a to reach the desired endpoint. Note that the UPBN2 nodes and the network nodes in the UP2s have entries in their LFIBs only to reach endpoints within their UP2. They can reach endpoints in other UP2s by using a FEC entry corresponding to the UP2 containing the destination endpoint, identified by PL1.

The following two observations help in further clarifying the forwarding operation above.

- The PL0 is used for forwarding from the source to the UPBN1-a. For a packet originating from an endpoint attached to a certain UP2, say UP2-b, nested to a different UP1, say UP1-b, PL0 is used for

forwarding in all network nodes that the packet transits until it reaches the UPBN1-a. This includes network nodes in UP2-b and UP1-b (i.e., "on the way up" from UP2). It also includes one of the UPBN1-b nodes.

It is important to note, however, that the PL0 is not popped at the UPBN1-b, since it is used for forwarding to the destination UPBN1-a.

- It should be pointed out that an important requirement for HSDN is to achieve route optimization for ECMP traffic, meaning that the hierarchy should forward a packet from any source to any destination using the same number of hops and without introducing any additional latency compared to a flat architecture. For example, a packet originating from an endpoint in UP2,N,M, and destined to an endpoint in the same UP2,N,M should not be forwarded all the way to the highest level in the hierarchy and back, but should be forwarded to the desired endpoint by "turning it around" towards the destination at the first node in the UP2,N,M that contains an entry to that desired endpoint. Indeed, if the packet turns around at the proper node, it will go through the same number of hops as it would have gone through in a flat architecture. This should hold true even in the case where the UP1s and/or UP2s contain intermediate tiers of switches and the packet needs to be turned around in the intermediate nodes.

  Route optimization is easily achieved in HSDN by simply having the packets only carry the portion of the label stack that is needed to reach the destination using the appropriate turn around node. Continuing with our example, the packet above only needs PL2 to be optimally forwarded, since it should never "go out" of UP2,N,M. Thus, PL0 and PL1 should not be included in its label stack, to avoid an unnecessary round trip up and down the DC through all the levels in the hierarchy. Similarly, a packet originating and terminating in the same UP1, but in different UP2s, only needs PL1 and PL2 to be forwarded.

  In this case, a network node would have to process different labels for traffic going up and out the partition versus traffic staying in the partition ("going up" and "coming down" refer to the direction of traffic in Figure 3). Since the label spaces for the two path labels may overlap, ambiguity would result. Depending on the LFIB configuration, the two Most Significant Bits (MSBs) in each label in Figure 4 may be reserved for identifying the layer (i.e., whether the label is PL0, PL1, or PL2) and resolve ambiguity.

  A better solution to achieve the same without using precious bits

in the labels is to use a "turn around entry" in the LFIBs, which flags that the packet needs to turn around at that node and the relevant label is not the outer label (as it would be for traffic going up or coming down, for which the outer label just needs to pass through), but is the one underneath (thus, the outer label needs to be popped to expose the relevant label). In our example, the packet destined to an endpoint in the same UP2,N,M of the originating server may carry a PL1 corresponding to the "turn around" label value and a PL2 corresponding to the desired endpoint within UP2, and does not need a PL0.

In the case of ECMP load-balanced non-TE traffic, the labels in the HSDN label stack identify ECMP groups for each destination in the corresponding partition. In this way, at each node in the partition, the outgoing label is the same for all paths belonging to the same ECMP group. A label allocation scheme for this is described in Section 5.

3.3.2 TE Traffic

Handling TE traffic in the hyper-scale DC/DCI presents major scalability challenges, since each TE tunnel contributes one entry in the forwarding tables, and the TE path and bandwidth allocation computation is a NP-complete problem.

HSDN introduces radical simplifications in establishing and handling tunnels, and in supporting TE in particular.

In HSDN, all paths in the network can be pre-established in the LFIBs. Because of the way the HSDN architecture is constructed, the number of entries that have to be stored in the local LFIB in each network node remains surprisingly small.

In this case, the labels in the HSDN stack identify entire paths, or groups of paths, to each destination in each partition, rather than just the destination itself.

With HSDN, since the "cost" of establishing a tunnel is essentially eliminated (all "tunnels" are pre-established in the network), and the TE task becomes one of path assignment and bandwidth allocation to the flows. Furthermore, the hierarchical structure of HSDN makes it possible to devise algorithms and heuristics for path and bandwidth allocation computation that operate largely independently in each partition, and are therefore computationally feasible even at large scale. A description of such algorithms is out of scope of this document. As a larger portion of the traffic can be engineered effectively, the network can be run at a higher utilization using comparatively smaller buffers at the nodes.

Since all paths can be accommodated in the LFIBs, HSDN makes it possible to support "TE Max Case" with small LFIB sizes. In TE Max Case, all sources are connected to all destinations (e.g., server to server) with TE tunnels, the tunnels using all possible distinct paths in the network. TE Max Case gives therefore an upper bound to the number of TE tunnels (and consequently, LFIB entries) in the network.

The fact that the LFIBs remain relatively small even when all possible paths are configured is the consequence of two desirable properties of HSDN.

First, since in HSDN the individual UPs are designed in such a way to be relatively small, the number of paths in each partition can be kept to a manageable number.

Second, the hierarchical structure of HSDN makes it possible to use the partitioning astutely to break the "TE Fanout Multiplicative Effect," which defines the number of paths to a destination, and can easily contribute to the LFIB explosion as the number of hops and the fanout of each hop to each destination in the network increases. As explained in Section 4.2, with the hierarchical structure, the TE Fanout Multiplicative Effect is only multiplicative within each level in the hierarchy. Thus, by properly designing the partitioning, the multiplicative effect can be kept to a manageable level.

In the case of TE traffic, the processing of the different labels in the label stack is similar to what described above for ECMP load-balanced non-TE traffic. However, the labels are bound to FECs identifying a specific path within each UPs that is traversed.

4. Scalability Analysis

In this section, we compute the maximum size of the LFIBs for non-TE/ECMP traffic and any-to-any server-to-server TE traffic.

4.1. LFIB Sizing - ECMP

For ECMP traffic, at each level, all destinations belonging to the same partition at a lower level are forwarded using the same FEC entry in the LFIB, which identifies the destination UPBG for that level, or the destination endpoint at the lower level. Since the UPs are designed in such a way to keep the number of destinations small in all UPs, and the network nodes only need to know how to reach destinations in their own UP and in the adjacent UP at the higher level in the hierarchy, this translate to the fact that hyper scale of the DC/DCI can be achieved with very small LFIB sizes in all the individual network nodes.

A detailed explanation of how the LFIB size can be computed in all the nodes of an HSDN network is given in [HSDNSOSR15]. The worst case for the LFIB size occurs at one of the network nodes that serve as UPBNs for one of the levels of UPs in the hierarchy. The level where the LFIB size occurs depend on the specific choice of the partitioning design.

4.2. LFIB Sizing - TE

As noted above, TE traffic may add a considerable number of entries to LFIB, since it creates one new FEC per TE tunnel to each destination.

HSDN provides a solution to this problem, and in fact, HSDN can support any-to-any server-to-server "TE Max Case" with small LFIB sizes.

In a Clos Topology (the analysis can be extended to generic topologies), the number of paths in a UP with N destination can be easily computed. The number of paths (and the maximum number of LFIB entries) is equal to the products of the switch fanout in each tier traversed from the source to the destination in that UP. This is the "TE Fanout Multiplicative Effect" mentioned above, which is illustrated in Figure 5. Accordingly:

Total # LFIB Entries for TE Max Case = N * F1 * F2 * ... * F(M-1)

Where Fi is the fanout of a switch in each tier traversed to the destination, M is the number of tiers in the UP, and N is the number of destinations in the UP.

Once again, by properly designing the UPs, the TE Fanout Multiplicative Effect can be kept under control, since the path computation is local for each of the UPs. HSDN breaks the multiplicative effect, since the TE Fanout Multiplicative Effect is multiplicative only within each UP, rather than in the entire network, and the "multiplication" restarts at each level of the hierarchy. A detailed description of the LFIB computation in all network nodes to support TE Max Case is given in [HSDNSOSR15].

```
                                +-------+
           Source               |  Src  |
                                | Node  |
                                +-------+  F1=3
                               /    |    \
                              /     |     \
                        +-------+ +-------+ +-------+
           Tier 1       |       | |       | |       |
                        |  Node | |  Node | |  Node |
                        +-+-----+ +---+---+ +-----+-+  F2=3
                          |   \ \ /   |   \ / /   |
                          |    \/ \---|---/ \/    |
                          |   / \ /---|---\ / \   |
                          | /  / \    |    / \  \ |
                        +-+-----+ +---+---+ +-----+-+
           Tier 2       |       | |       | |       |
                        |  Node | |  Node | |  Node |
                        +-------+ +-------+ +-------+  F3=2
                          \      \ /     \ /     /
                           \     / \     / \    /
                            \   /    / \    \  /
                             +-------+ +-------+
                             |       | |       |
           Tier 3            | Node  | | Node  |
                             +-------+ +-------+  F4=1
                                 \        /
                                  \      /
                                +-------+
                                | Dest  |
           Destination          | Node  |
                                +-------+
```

Figure 5. Fan out multiplicative effect with TE.

5. HSDN Label Stack Assignment Scheme

HSDN can use any scheme to assign the labels in the label stack. However, if a label assignment scheme which assigns labels in a way consistent with the HSDN structure, important simplifications can be achieved in the control plane and in the LFIBs.

For non-TE FECs, the HSDN label assignment scheme assigns labels according to the "physical" location of the endpoint in the HSDN structure. Continuing our design example from above, for an endpoint X in UP2-a, PL0 would identify the DC in which the endpoint is located, PL1 would identify the group of racks in which the endpoint is located within the DC, and PL2 would identify the endpoint within

the group of servers within the DC.

For TE FECs, the HSDN label assignment scheme assigns labels to identify a specific path in each UP that is traversed. In our example, for a specific TE tunnel to endpoint X, PL0 would identify the specific path that should be followed in the DCI, PL1 would identify the path that should be followed within the DC to reach the group of racks, and PL2 would identify the path to reach the endpoint within the group of racks (if there are multiple paths).

In order to assign labels to both non-TE traffic and TE traffic, HSDN uses a label format in which the labels are divided into two logical sub-fields, one identifying the destination within the UP, called Destination Identifier (DID), and one identifying the path, called Path Identifier (PID). The Path Identifier is only relevant for TE traffic, and can be zero for non-TE traffic. The HSDN Label format is illustrated in Figure 6.

```
0                           d                       19
++++++++++++++++++++++++++++++++++++++++++++++++++++++
|  Destination Identifier  |     Path Identifier    |
++++++++++++++++++++++++++++++++++++++++++++++++++++++
|<------- (d) bits ------->|<---- (20-d) bits ----->|
LSB                                               MSB
```

Figure 6. HSDN Label format.

In this label assignment scheme, the path labels associated with a partition are globally unique within that partition, meaning that different partitions at the same level can use the same label space. For PL0, the path labels are globally unique within the entire network, since there is only one UP0. Neither of these is a scaling limitation, since all partitions are relatively small.

The bits in the DID for each level must be sufficient to represent the distinct destinations that need to be known in the UPs at that level, and the bits in the PID for each level must be sufficient to represent all the distinct paths that need to be defined in the UPs at that level (closed-form expressions for both these numbers are given in [HSDNSOSR15]). In practice, this is not a significant scalability constraint: with three MPLS labels in the stack, partitioning architectures and label formats according to this scheme can be found to scale to tens of millions of servers.

Depending on the LFIB configuration, the two MSBs may be reserved for

identifying the level (i.e., whether the label is PL0, PL1, or PL2) to resolve ambiguity (not shown in Figure 6). Note, however, that this is not strictly necessary and the same function of identifying the level can be achieved by simply allocating "turn around" entries in the nodes, as explained in Section 3.3.1, so an individual node always sees the same label in the stack.

By properly designing the UPs, this label assignment scheme can support the desired scalability and the support of end-to-end TE traffic.

Note that by using this type of label assignment scheme important benefits can be achieved, including:

- The LFIBs become rather "static," since the FECs are tied to "physical" locations and paths, which change infrequently. This simplifies the use of the SDN approach to configure the LFIBs via a controller.

- All paths in each ECMP group use the same outgoing labels. This guarantees that a single LFIB entry can be used for each ECMP group.

The label stack needs to be imposed at the entry points. For an endpoint, this implies that the server NIC must be able to push a three-label stack of path labels (in addition to possibly push one additional VL label for the overlay network).

6. HSDN Architecture - Control Plane

HSDN has been designed to support the controller-centric SDN approach in a scalable fashion. HSDN also supports the traditional distributed control plane approach.

HSDN introduces important simplifications in the control plane and in the network state as well.

6.1. The SDN Approach

In the controller-centric SDN approach, the SDN controller configures the LFIBs in all the network nodes. With HSDN, the hierarchical partitioned structure offers a natural framework for a distributed implementation of the SDN controller, since the control plane in each UP is largely independent from other UPs. The individual UP control planes operate in parallel, with loose synchronization among one another.

Therefore, the HSDN control plane is logically partitioned in a way

that is consistent with the forwarding plane partitioning. Each UP is assigned a corresponding UP controller, which configures the LFIBs in the network nodes in the corresponding UP. The individual UP controllers communicate with one another to exchange the labels and construct the label stacks. In HSDN, configuring the LFIBs in the network nodes is not a difficult task, since the labels are static and configuration updates are needed only when the physical topology changes or endpoints are added or permanently removed, and thus they are not too frequent.

Each UP controller at the lowest level of the hierarchy is also in charge of providing the label stacks to the server’s NICs in the corresponding partition. For this purpose, a number of label servers, which may also be arranged in a hierarchy, are used to provide the mappings between IP addresses and label stacks.

Redundancy is superimposed to the structure of UP controllers, with each UP controller shadowing UP controllers in other UPs.

The HSDN UP controllers may also be in charge of TE computation. HSDN TE path computation algorithms that perform for the most part partition-local computation (so the computation is also horizontally scalable) but still approach global optimality using inter-UP-controller synchronization at a different time scale, can be devised.

6.2. HSDN Distributed Control Plane

The HSDN control plane can also be built using a hybrid approach, in which a routing or label distribution protocol is used to distribute the labels, in conjunction with a controller. An example using BGP-LU [RFC3107] is presented in [I-D.fang-idr-bgplu-for-hsdn].

7. Security Considerations

When the SDN approach is used, the protocols used to configure the LFIBs in the network nodes MUST be mutually authenticated.

For general MPLS/GMPLS security considerations, refer to [RFC5920].

Given the potentially very large scale and the dynamic nature in the cloud/DC environment, the choice of key management mechanisms need to be further studied.

To be completed.

8. IANA Considerations

TBD.

9. Acknowledgments

We would like to acknowledge Yakov Rekhter for many discussions related to HSDN.

10. Contributors

Vijay Gill
Salesforce
Email: vgill@salesforce.com

Linda Dunbar
Huawei Technologies
5430 Legacy Drive, Suite #175
Plano, TX 75024
Email: linda.dunbar@huawei.com

Andrew Qu
MediaTek
2860 Junction Ave.
San Jose, CA 95134
Email: andrew.qu@mediatek.com

Jeff Tantsura
Ericsson
200 Holger Way
San Jose, CA 95134
Email: jeff.tantsura@ericsson.com

Wen Wang
Century Link
2355 Dulles Corner Blvd.
Herndon, VA 20171
Email: wen.wang@centurylink.com

Himanshu Shah
Ciena
3939 North 1st Street
San Jose, CA 95112
Email: hshah@ciena.com

Ramki Krishnan
Dell
Email: ramki_krishnan@dell.com

Iftekhar Hussain
Infinera Corporation
140 Caspian Ct,

Sunnyvale, CA 94089
Email: ihussain@infinera.com

11. References

11.1 Normative References

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

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

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

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, May 2001.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[RFC7432] Sajassi et al., "BGP MPLS Based Ethernet VPN", RFC 7432, February 2015.

11.2 Informative References

[RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, July 2010.

[RFC7348] M. Mahalingam et al., "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.

[RFC7637] P. Garg et al., "NVGRE: Network Virtualization using Generic Routing Encapsulation", RFC 7637, Sept. 2015.

[I-D.fang-idr-bgplu-for-hsdn] L. Fang et al., "BGP-LU for HSDN Label Distribution", draft-fang-idr-bgplu-for-hsdn-02 (work in progress), July 2015.

[I-D.draft-gross-geneve] J. Gross et al., "Geneve: Generic Network Virtualization Encapsulation", draft-gross-geneve-02 (work in progress), October 2014.

[HSDNSOSR15] L. Fang et al., "Hierarchical SDN for the Hyper-Scale,

Hyper-Elastic Data Center and Cloud", ACM SIGCOMM Symposium on SDN Research 2015, Santa Clara, CA, June 2015.

Authors' Addresses

Luyuan Fang
Microsoft
15590 NE 31st St.
Redmond, WA 98052
Email: lufang@microsoft.com

Deepak Bansal
Microsoft
15590 NE 31st St.
Redmond, WA 98052
Email: dbansal@microsoft.com

Fabio Chiussi
Seattle, WA 98116
Email: fabiochiussi@gmail.com

Chandra Ramachandran
Juniper Networks
Electra, Exora Business Park Marathahalli - Sarjapur Outer Ring Road
Bangalore, KA 560103, India
Email: csekar@juniper.net

Ebben Aries
Facebook
1601 Willow Road
Menlo Park, CA 94025
Email: exa@fb.com

Shahram Davari
Broadcom
3151 Zanker Road
San Jose, CA 95134
Email: davari@broadcom.com

Barak Gafni
Mellanox
6 Habarzel St.
Tel Aviv, Israel
Email: gbarak@mellanox.com

Daniel Voyer

Bell Canada
Email: daniel.voyer@bell.ca

Nabil Bitar
Verizon
40 Sylvan Road
Waltham, MA 02145
Email: nabil.bitar@verizon.com

Network Working Group
Internet-Draft
Updates: 4379 (if approved)
Intended status: Standards Track
Expires: April 7, 2016

J. Luo, Ed.
ZTE
L. Jin, Ed.
Individual
T. Nadeau, Ed.
Lucidvision
G. Swallow, Ed.
Cisco

October 5, 2015

Relayed Echo Reply mechanism for LSP Ping
draft-ietf-mpls-lsp-ping-relay-reply-11

Abstract

In some inter autonomous system (AS) and inter-area deployment scenarios for RFC 4379 "Label Switched Path (LSP) Ping and Traceroute", a replying Label Switching Router (LSR) may not have the available route to an initiator, and the Echo Reply message sent to the initiator would be discarded resulting in false negatives or complete failure of operation of LSP Ping and Traceroute. This document describes extensions to LSP Ping mechanism to enable the replying LSR to have the capability to relay the Echo Response by a set of routable intermediate nodes to the initiator. This document updates RFC 4379.

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 April 7, 2016.

Copyright Notice

Copyright (c) 2015 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

This document describes extensions to the Label Switched Path (LSP) Ping as specified in [RFC4379], by adding a relayed echo reply mechanism which could be used to report data plane failures for inter autonomous system (AS) and inter-area LSPs. Without these extensions, the ping functionality provided by [RFC4379] would fail in many deployed inter-AS scenarios, since the replying Label Switching Router (LSR) in one AS may not have an available route to the initiator in the other AS. The mechanism in this document defines a new message type referred as "Relayed Echo Reply message", and a new TLV referred as "Relay Node Address Stack TLV".

This document is also to update [RFC4379], include updating of Echo Request sending procedure in section 4.3 of [RFC4379], Echo Request receiving procedure in section 4.4 of [RFC4379], Echo Reply sending procedure in Section 4.5 of [RFC4379], Echo Reply receiving procedure in section 4.6 of [RFC4379].

1.1. Conventions Used in This Document

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].

2. Motivation

LSP Ping [RFC4379] defines a mechanism to detect data plane failures and localize faults. The mechanism specifies that the Echo Reply should be sent back to the initiator using an UDP packet with the IPv4/IPv6 destination address set to an address of the LSR that originated the Echo Request. This works in administrative domains where IP address reachability is allowed among LSRs, and every LSR is able to route back to the originating LSR. However, in practice, this is often not the case due to intra-provider routing policy, route hiding, and network address translation at autonomous system border routers (ASBR). In fact, it is almost always the case that in inter-AS scenarios the only node in one AS to which direct routing is allowed from the other AS is the ASBR, and routing information from within one AS is not distributed into another AS.

Figure 1 demonstrates a case where an LSP is set up between PE1 and PE2. If PE1’s IP address is not distributed to AS2, a traceroute from PE1 directed towards PE2 can result in a failure because an LSR in AS2 may not be able to send the Echo Reply message. E.g., P2 cannot forward packets back to PE1 given that it is an routable IP address in AS1 but not routable in AS2. In this case, PE1 would detect a path break, as the Echo Reply messages would not be

received. Then localization of the actual fault would not be possible.

Note that throughout the document, routable address means that it is possible to route an IP packet to this address using the normal information exchanged by the IGP and BGP (or EGP) operating in the AS.

```
+-------+   +-------+   +------+   +------+   +------+   +------+
|       |   |       |   |      |   |      |   |      |   |      |
|  PE1  +---+   P1  +---+ ASBR1+---+ ASBR2+---+  P2  +---+  PE2 |
|       |   |       |   |      |   |      |   |      |   |      |
+-------+   +-------+   +------+   +------+   +------+   +------+
<---------------AS1-------------><---------------AS2------------>
<----------------------------- LSP ------------------------------>
```

Figure 1: Simple Inter-AS LSP Configuration

A second example that illustrates how [RFC4379] would be insufficient would be the inter-area situation in a seamless MPLS architecture [I-D.ietf-mpls-seamless-mpls] as shown below in Figure 2. In this example LSRs in the core network would not have IP reachable route to any of the access nodes (AN). When tracing an LSP from one AN to the remote AN, the LSR1/LSR2 node cannot send the Echo Reply either, like the P2 node in the inter-AS scenario in Figure 1.

```
           +-------+   +-------+   +------+   +------+
           |       |   |       |   |      |   |      |
        +--+ AGN11 +---+ AGN21 +---+ ABR1 +---+ LSR1 +--> to AGN
       /   |       |  /|       |   |      |   |      |
+----+/    +-------+\/ +-------+   +------+  /+------+
| AN |              /\                     \/
+----+\    +-------+  \+-------+   +------+/\ +------+
       \   |       |   |       |   |      |  \|      |
        +--+ AGN12 +---+ AGN22 +---+ ABR2 +---+ LSR2 +--> to AGN
           |       |   |       |   |      |   |      |
           +-------+   +-------+   +------+   +------+
static route     ISIS L1 LDP             ISIS L2 LDP
<-Access-><--Aggregation Domain--><---------Core--------->
```

Figure 2: Seamless MPLS Architecture

This document describes extensions to the LSP Ping mechanism to facilitate a response from the replying LSR, by defining a mechanism that uses a relay node (e.g, ASBR) to relay the message back to the initiator. Every designated or learned relay node must be reachable to the next relay node or to the initiator. Using a recursive approach, relay node could relay the message to the next relay node until the initiator is reached.

The LSP Ping relay mechanism in this document is defined for unicast case. How to apply the LSP Ping relay mechanism in multicast case is out of the scope.

3. Extensions

[RFC4379] defines two message types, Echo Request and Echo Reply. This document defines a new message type, Relayed Echo Reply. The Relayed Echo Reply message is used in place of the Echo Reply message when an LSR is replying LSR to a relay node.

A new TLV named Relay Node Address Stack TLV is defined in this document, to carry the IP addresses of the relay nodes for the replying LSR.

In addition, MTU (Maximum Transmission Unit) Exceeded Return Code is defined to indicate to the initiator that one or more TLVs will not be returned due to MTU size.

It should be noted that this document focuses only on detecting the LSP which is set up using a uniform IP address family type. That is, all hops between the source and destination node use the same address family type for their LSP ping control planes. This does not preclude nodes that support both IPv6 and IPv4 addresses simultaneously, but the entire path must be addressable using only one address family type. Supporting for mixed IPv4-only and IPv6-only is beyond the scope of this document.

3.1. Relayed Echo Reply message

The Relayed Echo Reply message is a UDP packet, and the UDP payload has the same format with Echo Request/Reply message. A new message type is requested from IANA.

New Message Type:

Value	Meaning
TBD1	MPLS Relayed Echo Reply

The use of TCP and UDP port number 3503 is described in [RFC4379] and has been allocated by IANA for LSP Ping messages. The Relayed Echo Reply message will use the same port number.

3.2. Relay Node Address Stack

The Relay Node Address Stack TLV is an optional TLV. It MUST be carried in the Echo Request, Echo Reply and Relayed Echo Reply messages if the echo reply relayed mechanism described in this document is required. Figure 3 illustrates the TLV format.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Type           |               Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Initiator Source Port       | Reply Add Type|   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Source Address of Replying Router (0, 4, or 16 octets)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Destination Address Offset   |   Number of Relayed Addresses |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                Stack of Relayed Addresses                     ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Relay Node Address Stack TLV

- Type: value is TBD2. The value should be assigned by IANA from 32768-49161 as suggested by [RFC4379] Section 3.

- Length: the length of the value field in octets.

- Initiator Source Port: the source UDP port that the initiator uses in the Echo Request message, and also the port that is expected to receive the Echo Reply message.

- Reply Address Type: address type of replying router. This value also implies the length of the address field as shown below.

Type#	Address Type	Address Length
0	Null	0
1	IPv4	4
2	IPv6	16

- Reserved: This field is reserved and MUST be set to zero.

- Source Address of Replying Router: source IP address of the originator of Echo Reply or Replay Echo Reply message.

- Destination Address Offset: the offset in octets from the top-of-stack to the destination address entry. Each entry size has been listed in this section. Please also refer to section 4.2 for more detail of the operation.

- Number of Relayed Addresses: an integer indicating the number of relayed addresses in the stack.

- Stack of Relayed Addresses: a list of relay node address entries. This stack grows downward, with relay nodes address further along the LSP appearing lower down in the stack. Please refer to section 4.2 for the relay node discovery mechanism.

The format of each relay node address entry is as below:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Address  Type |K|  Reserved   |          Reserved             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~           Relayed Address (0, 4, or 16 octets)                ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Relay Node Address Entry

Type#	Address Type	Address Length	Size of the Entry
0	Null	0	4
1	IPv4	4	8
2	IPv6	16	20

Reserved: The two fields are reserved and MUST be set to zero.

K bit: if the K bit is set to 1, then this address stack entry MUST NOT be stripped from the Relay Node Address Stack during processing described in Section 4.2. If the K bit is clear, the entry might be stripped according to the processing described in Section 4.2.

Having the K bit set in the relay node address entry causes that entry to be preserved in the Relay Node Address Stack TLV for the entire traceroute operation. A responder node MAY set the K bit to ensure its relay node address entry remains as one of the relay nodes in the Relay Node Address Stack TLV. The address with K bit set will

always be a relay node address for the Relayed Echo Reply, see section 4.3.

Relayed Address: this field specifies the node address, either IPv4 or IPv6.

3.3. MTU Exceeded Return Code

Return Code is defined to indicate that one or more TLVs were omitted from the Echo Reply or Relayed Echo Reply message to avoid exceeding the message's effective MTU size. These TLVs MAY be included in an Errored TLV's Object with their lengths set to 0 and no value. The return sub-code MUST be set to the value that otherwise would have been sent. For more detail, please refer to section 4.2.

MTU Exceeded Return Code:

Value	Meaning
TBD2	One or more TLVs not returned due to MTU size

Then section 4.4 of [RFC4379], step 7 will be updated to integrate the processing of MTU Exceeded Return Code.The following text will be added:

Before sending Echo Reply, the new packet size should be checked. If Best-return-code is 3 ("Replying router is an egress for the FEC at stack depth"), or 8 ("Label switched at stack-depth"), and if the packet size exceeds MTU size, then Best-return-code is TBD3 ("One or more TLVs not returned due to MTU size")

4. Procedures

To preform a ping operation, the initiator first discovers the relay nodes. Once those nodes have been discovered, the initiator includes the Relay Node Address Stack TLV into any Echo Request message. The node can then ping as normal. Note that in some cases, the repeated lack of replies to Echo Request messages may be due to a route change that has impacted the necessary stack of relay nodes. In this case the initiator may need to re-discover the relay nodes. The following sections describe the procedures for sending and receiving Echo Request messages with the the Relay Node Address Stack TLV. These procedures can be used in "trace route" mode to discover the relay nodes.

4.1. Sending an Echo Request

In addition to the procedures described in section 4.3 of [RFC4379], a Relay Node Address Stack TLV MUST be carried in the Echo Request message if the relay functionality is required. The relay function initiation is out of the scope of this document, one possible way is that the operator will explicitly add an option to the ping command.

When the initiator sends the first Echo Request with a Relay Node Address Stack TLV, the TLV MUST contain the initiator address as the first entry of the stack of relayed addresses, the Destination Address Offset set to this entry, and the source address of the replying router set to null. The Initiator Source Port field MUST be set to the source UDP port. Note that the first relay node address in the stack will always be the initiator’s address.

When sending subsequent Echo Request message, refer to section 4.6.

4.2. Receiving an Echo Request

The Type of the Relay Node Address Stack TLV is chosen from the range defined in [RFC4379] as "optional TLVs that can be silently dropped if not recognized. An LSR that does not recognize the TLV SHOULD ignore it.

In addition to the processes in section 4.4 of [RFC4379], the procedures of the Relay Node Address Stack TLV are defined here.

Upon receiving a Relay Node Address Stack TLV in an Echo Request message, the receiver updates the "Source Address of Replying Router". The address MUST be same as the source IP address of Relay Echo Reply (section 4.3) or Echo Reply message (section 4.5) being sent.

Those address entries with K bit set to 1 MUST be kept in the stack. The receiver MUST check the addresses of the stack in sequence from bottom to top to find the last address in the stack with the K bit set (or the top of the stack if no K bit was found). The receiver then checks the stack beginning with this entry, proceeding towards the bottom to find the first routable address IP address. The Destination Address Offset MUST be set to this entry which is also the resolved destination address. Address entries below the first routable IP address MUST be deleted. At least one address entries of the replying LSR MUST be added at the bottom of the stack. A second or more address entries MAY also be added if necessary, depending on implementation. The final address added MUST be an address that is reachable through the interface that the Echo Request Message would have been forwarded if it had not TTL expired at this node. The

updated Relay Node Address Stack TLV MUST be carried in the response message.

If the replying LSR is configured to hide its routable address information, the address entry added in the stack MUST be a NIL entry with Address Type set to NULL.

If a node spans two addressing domains (with respect to this message) where nodes on either side may not be able to reach nodes in the other domain, then the final address added SHOULD set the K bit. One example of spanning two address domains is the ASBR node. Other cases are also possible, and out of the scope of this document.

K bit applies in the case of a NULL address, to serve as a warning to the initiator that further Echo Request messages may not result in receiving Echo Reply messages.

If the full reply message would exceed the MTU size, the Relay Node Address Stack TLV SHOULD be included in the Echo Reply message (i.e., other optional TLVs are excluded).

4.3. Originating an Relayed Echo Reply

The destination address determined in section 4.2 is used as the next relay node address. If the resolved next relay node address is not routable, then sending of Relayed Echo Reply or Echo Reply will fail.

If the first IP address in the Relay Node Address Stack TLV is not the next relay node address, the replying LSR SHOULD send a Relayed Echo Reply message to the next relay node. The processing of Relayed Echo Reply is the same with the procedure of the Echo Reply described in Section 4.5 of [RFC4379], except the destination IP address and the destination UDP port. The destination IP address of the Relayed Echo Reply is set to the next relay node address from the Relay Node Address Stack TLV, and both the source and destination UDP port are set to 3503. The updated Relay Node Address Stack TLV described in section 4.2 MUST be carried in the Relayed Echo Reply message. The Source Address of Replying Router field is kept unmodified, and Source IP address field of the IP header is set to an address of the sending node.

When the next relay node address is the first one in the address list, please refer to section 4.5.

4.4. Relaying an Relayed Echo Reply

Upon receiving an Relayed Echo Reply message with its own address as the destination address in the IP header, the relay node MUST determine the next relay node address as described in section 4.2, with the modification that the location of the received destination address is used instead of the bottom of stack in the algorithm. The Destination Address Offset in Relay Node Address Stack TLV will be set to the next relay node address. Note that unlike section 4.2 no changes are made to the Stack of Relayed Addresses.

If the next relay node address is not the first one in the address list, i.e., another intermediate relay node, the relay node MUST send an Relayed Echo Reply message to the determined upstream node with the payload unchanged other than the Relay Node Address Stack TLV. The TTL SHOULD be copied from the received Relay Echo Reply and decremented by 1. The Source Address of Replying Router field is kept unmodified, and Source IP address field of the IP header is set to an address of the sending node.

When the next relay node address is the first one in the address list, please refer to section 4.5.

4.5. Sending an Echo Reply

The Echo Reply is sent in two cases:

1. When the replying LSR receives an Echo Request, and the first IP address in the Relay Node Address Stack TLV is the next relay node address (section 4.3), the replying LSR would send an Echo Reply to the initiator. In addition to the procedure of the Echo Reply described in Section 4.5 of [RFC4379], the updated Relay Node Address Stack TLV described in section 4.2 MUST be carried in the Echo Reply.

If the receiver does not recognize the Relay Node Address Stack TLV, as per section 3 and 4.5 of [RFC4379], it will send an Echo Reply without including the TLV.

2. When the intermediate relay node receives a Relayed Echo Reply, and the first IP address in the Relay Node Address Stack TLV is the next relay node address (section 4.4), the intermediate relay node would send the Echo Reply to the initiator, and update the Message Type field from type of Relayed Echo Reply to Echo Reply. The updated Relay Node Address Stack TLV described in section 4.4 MUST be carried in the Echo Reply. The destination IP address of the Echo Reply is set to the first IP address in the stack, and the destination UDP port would be copied from the Initiator Source Port field of the Relay Node Address Stack TLV. The source UDP port

should be 3503. The TTL of the Echo Reply SHOULD be copied from the received Relay Echo Reply and decremented by 1. The Source Address of Replying Router field is kept unmodified, and Source IP address field of the IP header is set to an address of the sending node.

4.6. Sending Subsequent Echo Requests

During a traceroute operation, multiple Echo Request messages are sent. Each time the TTL is increased, the initiator SHOULD copy the Relay Node Address Stack TLV received in the previous Echo Reply to the next Echo Request. The Relay Node Address Stack TLV MUST NOT be modified except as follows. A NIL entry with K bit unset MAY be removed. A NIL entry with K bit serves as a warning that further Echo Request messages are likely to not result in a reply. If, however, the initiator decides to continue a traceroute operation, the NIL entry with the K bit set MUST be removed. The Source Address of Replying Router and Destination Address Offset fields may be preserved or reset since these fields are ignored in received MPLS Echo Request.

4.7. Impact to Traceroute

Source IP address in Echo Reply and Relay Echo Reply is to be of the address of the node sending those packets, not the original responding node. Then the traceroute address output module will print the source IP address as below:

```
if (Relay Node Address Stack TLV exists) {
      Print the Source Address of Replying Router in
      Relay Node Address Stack TLV;
} else {
      Print the source IP address of Echo Reply message;
}
```

5. LSP Ping Relayed Echo Reply Example

Considering the inter-AS scenario in Figure 5 below. AS1 and AS2 are two independent address domains. In the example, an LSP has been created between PE1 to PE2, but PE1 in AS1 is not reachable by P2 in AS2.

```
+-------+   +-------+   +------+   +------+   +------+   +------+
|       |   |       |   |      |   |      |   |      |   |      |
|  PE1  +---+   P1  +---+ ASBR1+---+ ASBR2+---+  P2  +---+  PE2 |
|       |   |       |   |      |   |      |   |      |   |      |
+-------+   +-------+   +------+   +------+   +------+   +------+
<---------------AS1-------------><---------------AS2------------>
<---------------------------- LSP ------------------------------->
```

Figure 5: Example Inter-AS LSP

When performing LSP traceroute on the LSP, the first Echo Request sent by PE1 with outer-most label TTL=1, contains the Relay Node Address Stack TLV with PE1's address as the first relayed address.

After processed by P1, P1's interface address facing ASBR1 without the K bit set will be added in the Relay Node Address Stack TLV address list following PE1's address in the Echo Reply.

PE1 copies the Relay Node Address Stack TLV into the next Echo Request when receiving the Echo Reply.

Upon receiving the Echo Request, ASBR1 checks the address list in the Relay Node Address Stack TLV, and determines that PE1's address is the next relay address. Then deletes P1's address, and adds its interface address facing ASBR2 with the K bit set. As a result, there would be PE1's address followed by ASBR1's interface address facing ASBR2 in the Relay Node Address Stack TLV of the Echo Reply sent by ASBR1.

PE1 then sends an Echo Request with outer-most label TTL=3, containing the Relay Node Address Stack TLV copied from the received Echo Reply message. Upon receiving the Echo Request message, ASBR2 checks the address list in the Relay Node Address Stack TLV, and determines ASBR1's interface address is the next relay address in the stack TLV. ASBR2 adds its interface address facing P2 with the K bit set. Then ASBR2 sets the next relay address as the destination address of the Relayed Echo Reply, and sends the Relayed Echo Reply to ASBR1.

Upon receiving the Relayed Echo Reply from ASBR2, ASBR1 checks the address list in the Relay Node Address Stack TLV, and determines that PE1's address is the next relay node. Then ASBR1 sends an Echo Reply to PE1.

For the Echo Request with outer-most label TTL=4, P2 checks the address list in the Relay Node Address Stack TLV, and determines that ASBR2's interface address is the next relay address. Then P2 sends

an Relayed Echo Reply to ASBR2 with the Relay Node Address Stack TLV containing four addresses, PE1's, ASBR1's interface address, ASBR2's interface address and P2's interface address facing PE2 in sequence.

Then according to the process described in section 4.4, ASBR2 sends the Relayed Echo Reply to ASBR1. Upon receiving the Relayed Echo Reply, ASBR1 sends an Echo Reply to PE1. And as relayed by ASBR2 and ASBR1, the Echo Reply would finally be sent to the initiator PE1.

For the Echo Request with outer-most label TTL=5, the Echo Reply would relayed to PE1 by ASBR2 and ASBR1, similar to the case of TTL=4.

The Echo Reply from the replying node which has no IP reachable route to the initiator is thus transmitted to the initiator by multiple relay nodes.

6. Security Considerations

The Relayed Echo Reply mechanism for LSP Ping creates an increased risk of DoS by putting the IP address of a target router in the Relay Node Address Stack. These messages then could be used to attack the control plane of an LSR by overwhelming it with these packets. A rate limiter SHOULD be applied to the well-known UDP port on the relay node as suggested in [RFC4379]. The node which acts as a relay node SHOULD validate the relay reply against a set of valid source addresses and discard packets from untrusted border router addresses. An implementation SHOULD provide such filtering capabilities.

If an operator wants to obscure their nodes, it is RECOMMENDED that they may replace the replying node address that originated the Echo Reply with NIL address entry in Relay Node Address Stack TLV.

A receiver of an MPLS Echo Request could verify that the first address in the Relay Node Address Stack TLV is the same address as the source IP address field of the received IP header.

The Relay Node Address Stack TLV has the path information of the LSP, and such information may be maliciously used by any uncontrolled LSR/LER. We have two ways to reduce the path information in the TLV:

1. it is recommended to clear the K bit in the relay address entry unless you have to.

2. it is encouraged to use NIL address entry to hide node information if possible.

Other security considerations discussed in [RFC4379], are also applicable to this document.

7. Backward Compatibility

When one of the nodes along the LSP does not support the mechanism specified in this document, the node will ignore the Relay Node Address Stack TLV as described in section 4.2. Then the initiator may not receive the Relay Node Address Stack TLV in Echo Reply message from that node. In this case, an indication should be reported to the operator, and the Relay Node Address Stack TLV in the next Echo Request message should be copied from the previous Echo Request, and continue the ping process. If the node described above is located between the initiator and the first relay node, the ping process could continue without interruption.

8. IANA Considerations

IANA is requested to assign one new Message Type, one new TLV and one Return Code.

8.1. New Message Type

This document requires allocation of one new message type from "Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters" registry, the "Message Type" registry:

Value	Meaning
TBD1	MPLS Relayed Echo Reply

The value should be assigned from the "Standards Action" range (0-191), and using the lowest free value within this range.

8.2. New TLV

This document requires allocation of one new TLV from "Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters" registry, the "TLVs" registry:

Type	Meaning
TBD2	Relay Node Address Stack TLV

A value should be assigned from "Standards Action" range (32768-49161) as suggested by [RFC4379] Section 3, using the first free value within this range.

8.3. MTU Exceeded Return Code

This document requires allocation of MTU Exceeded return code from "Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters" registry, the "Return Codes" registry:

Value	Meaning
TBD3	One or more TLVs not returned due to MTU size

The value should be assigned from the "Standards Action" range (0-191), and using the lowest free value within this range.

9. Acknowledgement

The authors would like to thank Carlos Pignataro, Xinwen Jiao, Manuel Paul, Loa Andersson, Wim Henderickx, Mach Chen, Thomas Morin, Gregory Mirsky, Nobo Akiya and Joel M. Halpern for their valuable comments and suggestions.

10. Contributors

Ryan Zheng
JSPTPD
371, Zhongshan South Road
Nanjing, 210006, China
Email: ryan.zhi.zheng@gmail.com

11. References

11.1. 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>.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, DOI 10.17487/RFC4379, February 2006, <http://www.rfc-editor.org/info/rfc4379>.

11.2. Informative References

[I-D.ietf-mpls-seamless-mpls]
Leymann, N., Decraene, B., Filsfils, C., Konstantynowicz, M., and D. Steinberg, "Seamless MPLS Architecture", draft-ietf-mpls-seamless-mpls-07 (work in progress), June 2014.

Authors' Addresses

Jian Luo (editor)
ZTE
50, Ruanjian Avenue
Nanjing, 210012, China

Email: luo.jian@zte.com.cn

Lizhong Jin (editor)
Individual
Shanghai, China

Email: lizho.jin@gmail.com

Thomas Nadeau (editor)
Lucidvision

Email: tnadeau@lucidvision.com

George Swallow (editor)
Cisco
300 Beaver Brook Road
Boxborough , MASSACHUSETTS 01719, USA

Email: swallow@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 4, 2015

S. Kini, Ed.
Ericsson
K. Kompella
Juniper
S. Sivabalan
Cisco
S. Litkowski
Orange
R. Shakir
B.T.
X. Xu
Huawei
W. Hendrickx
Alcatel-Lucent
J. Tantsura
Ericsson
March 3, 2015

Entropy labels for source routed stacked tunnels
draft-kini-mpls-spring-entropy-label-03

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 4, 2015.

Copyright Notice

Copyright (c) 2015 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

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

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

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

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

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

1.1. Requirements Language

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

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

2. Abbreviations and Terminology

EL - Entropy Label

ELI - Entropy Label Identifier

ELC - Entropy Label Capability

SR - Segment Routing

ECMP - Equal Cost Multi Paths

MPLS - Multiprotocol Label Switching

SID - Segment Identifier

RLD - Readable Label Depth

OAM - Operation, Administration and Maintenance

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

```
                        +------+
                        |      |
                +-------|  P3  |-----+
                | +-----|      |---+ |
              L3| |L4   +------+ L1| |L2     +----+
                | |                | |    +--| P4 |--+
   +-----+     +-----+            +-----+  |  +----+  |  +-----+
   |  S  |-----| P1  |------------| P2  |--+          +--|  D  |
   |     |     |     |            |     |--+          +--|     |
   +-----+     +-----+            +-----+  |  +----+  |  +-----+
                                           +--| P5 |--+
                                              +----+
```

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

Figure 1: Traffic engineering use-case

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

4. Recommended EL solution for SPRING

The solution described in this section follows [RFC6790].

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

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

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

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

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

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

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

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

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

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

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

5. Options considered

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

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

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

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

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

5.2. An EL per tunnel in the stack

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

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

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

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

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

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

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

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

5.3.1. EL at top of stack

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

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

5.4. ELs at readable label stack depths

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

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

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

6. Acknowledgements

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

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

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

9. References

9.1. Normative References

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

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

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

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

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

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

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

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

9.2. Informative References

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

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

[I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions-04 (work in progress), February 2015.

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

Authors' Addresses

Sriganesh Kini (editor)
Ericsson

Email: sriganesh.kini@ericsson.com

Kireeti Kompella
Juniper

Email: kireeti@juniper.net

Siva Sivabalan
Cisco

Email: msiva@cisco.com

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Rob Shakir
B.T.

Email: rob.shakir@bt.com

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Wim Hendrickx
Alcatel-Lucent

Email: wim.henderickx@alcatel-lucent.com

Jeff Tantsura
Ericsson

Email: jeff.tantsura@ericsson.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: 16 June 2022

K. Kompella
R. Balaji
Juniper Networks
R. Thomas
Cohesity

13 December 2021

Label Distribution Using ARP
draft-kompella-mpls-larp-11

Abstract

This document describes extensions to the Address Resolution Protocol to distribute MPLS labels for IPv4 and IPv6 host addresses. Distribution of labels via ARP enables simple plug-and-play operation of MPLS, which is key to deploying MPLS in data centers and enterprises.

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 16 June 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 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

This document describes extensions to the Address Resolution Protocol (ARP) [RFC0826] to advertise label bindings for IP host addresses. While there are well-established protocols, such as LDP [RFC5036], RSVP [RFC3209], BGP [RFC3107] and SPRING-MPLS [RFC8660], that provide robust mechanisms for label distribution, these protocols tend to be relatively complex, and often require detailed configuration for proper operation. There are situations where a simpler protocol may be more suitable from an operational standpoint. An example is the case where an MPLS Fabric is the underlay technology in a Data Center; here, MPLS tunnels originate from host machines. The host thus needs a mechanism to acquire label bindings to participate in the MPLS Fabric, but in a simple, plug-and-play manner. Existing signaling/routing protocols do not always meet this need. Labeled

ARP (L-ARP) is a proposal to fill that gap.

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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The term "server" will be used in this document to refer to an ARP/L-ARP server; the term "host" will be used to refer to a compute server or other device acting as an ARP/L-ARP client.

1.2. Approach

ARP is a nearly ubiquitous protocol; every device with an Ethernet interface, from hand-helds to hosts, have an implementation of ARP. ARP is plug-and-play; ARP clients do not need configuration to use ARP. That suggests that ARP may be a good fit for devices that want to source and sink MPLS tunnels, but do so in a zero-config, plug-and-play manner, with minimal impact to their code.

The approach taken here is to create a minor variant of the ARP protocol, labeled ARP (L-ARP), which is distinguished by a new hardware type, MPLS-over-Ethernet. Regular (Ethernet) ARP (E-ARP) and L-ARP can coexist; a device, as an ARP client, can choose to send out an E-ARP or an L-ARP request, depending on whether it needs Ethernet or MPLS connectivity. Another device may choose to function as an E-ARP server and/or an L-ARP server, depending on its ability to provide an IP-to-Ethernet and/or IP-to-MPLS mapping.

2. Overview of Ethernet ARP

In the most straightforward mode of operation [RFC0826], ARP queries are sent to resolve "directly connected" IP addresses. The ARP request is broadcast, with the Target Protocol Address field (see Section 5 for a description of the fields in an ARP message) carrying the IP address of another node in the same subnet. All the nodes in the LAN receive this ARP request. All the nodes, except the node that owns the IP address, ignore the ARP request. The IP address owner learns the MAC address of the sender from the Source Hardware Address field in the ARP request, and unicasts an ARP reply to the sender. The ARP reply carries the replying node’s MAC address in the Source Hardware Address field, thus enabling two-way communication between the two nodes.

A variation of this scheme, known as "proxy ARP" [RFC2002], allows a node to respond to an ARP request with its own MAC address, even when the responding node does not own the requested IP address. Generally, the proxy ARP response is generated by routers to attract traffic for prefixes they can forward packets to. This scheme requires the host to send ARP queries for the IP address the host is trying to reach, rather than the IP address of the router. When there is more than one router connected to a network, proxy ARP enables a host to automatically select an exit router without running any routing protocol to determine IP reachability. Unlike regular ARP, a proxy ARP request can elicit multiple responses, e.g., when more than one router has connectivity to the address being resolved. The sender must be prepared to select one of the responding routers.

Yet another variation of the ARP protocol, called 'Gratuitous ARP' [RFC2002], allows a node to update the ARP cache of other nodes in an unsolicited fashion. Gratuitous ARP is sent as either an ARP request or an ARP reply. In either case, the Source Protocol Address and Target Protocol Address contain the sender's address, and the Source Hardware Address is set to the sender's hardware address. In case of a gratuitous ARP reply, the Target Hardware Address is also set to the sender's address.

3. L-ARP Protocol Operation

The L-ARP protocol builds on the proxy ARP model, and also leverages gratuitous ARP model for asynchronous updates.

In this memo, we will refer to L-ARP clients (that make L-ARP requests) and L-ARP servers (that send L-ARP responses). In Figure 1, H1, H2 and H3 are L-ARP clients, and T1, T2 and T3 are L-ARP servers. T4 is a member of the MPLS Fabric that may not be an L-ARP server. Within the MPLS Fabric, the usual MPLS protocols (IGP (i.e., SPRING-MPLS), LDP, RSVP-TE) are run. Say H1, H2 and H3 want to establish MPLS tunnels to each other (for example, they are using BGP MPLS VPNs as the overlay virtual network technology). H1 might also want to talk to a member of the MPLS Fabric, say T. Also, the "protocol" addresses in L-ARP requests are either IPv4 or IPv6 addresses; note that while it is common to use Neighbor Discovery (ND) [RFC4861] for "regular" ARP requests when dealing with IPv6 (i.e., to obtain Ethernet MAC addresses corresponding to an IPv6 address), ND is not used when the ARP request is for an MPLS label.

```
            . . . . . . .
           .             .
   H1 --- T1             T4
      \   .     MPLS      .
       \  .               .
        \ .    Fabric     .
   H2 --- T2             T3 --- H3
           .             .
            . . . . . . .
```

Figure 1: MPLS Fabric

3.1. Setup

In Figure 1, the nodes T1-T4, and those in between making up the "MPLS Fabric" are assumed to be running some protocol whereby they can signal MPLS reachability to themselves and to other nodes (like hosts H1-H3). T1-T3 are L-ARP servers; T4 need not be, since it doesn't have an attached L-ARP client. H1-H3 are L-ARP clients.

3.2. Egress Operation

A node (say T3) that wants an attached node (say H3) to have MPLS reachability allocates a label L3 to reach H3 and advertises this label into the MPLS Fabric. This can be triggered by configuration on T3, or when T3 first receives an L-ARP request from H3 (indicating that H3 wants MPLS connectivity), or via some other protocol. T3 then advertises (H3, L3) to its peers in the MPLS Fabric so that all members of the Fabric have connectivity to H3. This advertisement can be one of the following:

* a "proxy" LDP message (sent on behalf of H3) with prefix H3 and label L3; or

* a node Segment ID (SID) advertised on behalf of H3; or

* a labeled BGP advertisement, with prefix H3, label L3 and next hop self.

  On receiving a packet with label L3, T3 pops the label and send the packet to H3. (In the case of labeled BGP, there would be a two-label stack, with outer label to reach T3 and inner label of L3.) This is the usual operation of an MPLS Fabric, with the addition of advertising labels for nodes outside the fabric.

3.3. Ingress Operation

A node (say H1, an L-ARP client) that needs an MPLS tunnel to another node (say H3) identified by a host address (either IPv4 or IPv6) broadcasts over all its interfaces an L-ARP request with the Target Protocol Address set to H3 and Hardware Type set to "MPLS-over-Ethernet". A node receiving the L-ARP request (say T1, an L-ARP server) does the following:

1. checks if it has MPLS reachability to H3. If not, it ignores the L-ARP request.

2. if it does, T1 allocates a label TL3 to reach H3 (if it doesn't already have such a label) and installs an L-FIB entry to swap L1 with the label (stack) to reach H3.

3. sends a (proxy) L-ARP reply to H1 with the Source Hardware Address (SHA) set to (L, M), where M is T1's metric to H3. T1 may also set some attribute bits in the SHA.

3.4. Data Plane

To send a packet to H3 over an MPLS tunnel, H1 pushes L1 onto the packet, sets the destination MAC address to M1 and sends it to T1. On receiving this packet, T1 swaps the top label with the label(s) for its MPLS tunnel to H3. If T1's reachability to H3 is via a SPRING label stack, the label L1 acts as an implicit binding SID.

If H1 and H3 have an overlay connection (say an IPVPN [RFC4364] VPN-foo) whereby VM1 on H1 wishes to talk to VM3 on H3 over VPN-foo, H1 does the following:

1. H1 learns information about VPN-foo via BGP (or an SDN controller), including the VPN label VL3 to use to talk to VM3;

2. H1 installs a VRF for VPN-foo, with prefix VM3, label VL3 and next hop H3;

3. H1 binds the local "veth" interface to VM1 to this VRF.

4. When VM1 sends a packet to dest IP address VM3 over its veth interface, H1 looks up VM3 in the corresponding VRF, gets label VL3. It then sends an L-ARP request for next hop H3, and gets TL3.

5. Finally, H1 pushes the label pair (TL3, VL3) onto the packet from VM1 and sends this to T1. This packet will then end up at VM3 on H3.

Note that H1 broadcasts its L-ARP request over its attached interfaces. H1 may receive several L-ARP replies; in that case, H1 can select any subset of these to send MPLS packets destined to H3. As described later, the L-ARP response may contain certain parameters that enable the client to make an informed choice. If the target H3 belongs to one of the subnets that H1 participates in, and H3 is capable of sending L-ARP replies, H1 can use H3's response to send MPLS packets to H3.

4. Attributes

In addition to carrying a label stack to be used in the data plane, an L-ARP reply carries some attributes that are typically used in the control plane. One of these is a metric. The metric is the distance from the L-ARP server to the destination. This allows an L-ARP client that receives multiple responses to decide which ones to use, and whether to load-balance across some of them. The metric typically will be the IGP shortest path distance from server to the destination; this makes comparing metrics from different servers meaningful.

Another attribute is Entropy Label (EL) Capability. This attribute says whether the destination is EL capable (ELC). In Figure 1, if T3 advertises a label to reach H3 and T3 is ELC, T3 can include in its signaling to T1 that it is ELC. In that case, T1's L-ARP reply to H1 can have ELC bit set. This tells H1 that it may include (below the outermost label) an Entropy Label Indicator followed by an Entropy Label. This will help improve load balancing across the MPLS Fabric, and possibly on the last hop to H3.

4.1. Secondary Attributes

Beyond the basic attributes that are carried with every L-ARP request, there more optional attributes, for example, to ask for certain characteristics of the path traffic takes to the destination. These attributes are carried in TLVs that are carried in L-ARP requests and replies.

One such TLV is the Classful Transport (CT: see [I-D.kaliraj-idr-bgp-classful-transport-planes]) TLV. This TLV allows the L-ARP client to request a label to a destination over a tunnel of the given Transport Class. To satisfy this request, the L-ARP server creates (or finds) a tunnel to the destination that is routed over the CT Transport Plane, allocates a label L, inserts an entry in the LFIB to swap L to the tunnel, and sends L to the L-ARP client in its reply.

5. L-ARP Message Format

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           ar$hrd              |            ar$pro             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ar$hln    |    ar$pln     |            ar$op              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                     ar$sha (ar$hln octets)                  //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                     ar$spa (ar$pln octets)                  //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                     ar$tha (ar$hln octets)                  //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                     ar$tpa (ar$pln octets)                  //
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            Type               |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                            Value                            //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type               |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                            Value                            //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ...                                                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: L-ARP Packet Format

ar$hrd: Hardware Type: MPLS-over-Ethernet. The value of the field used here is HTYPE-MPLS. To start with, we will use the experimental value HW_EXP2 (256).

ar$pro: Protocol Type: IPv4/IPv6. The value of the field used here is 0x0800 to resolve an IPv4 address and 0x86DD to resolve an IPv6 address.

ar$hln: Hardware Address Length: 6

ar$pln: Protocol Address Length: for an IPv4 address, the length is 4 octets; for an IPv6 address, it is 16.

ar$op: Operation Code: set to 1 for request, 2 for reply, and 10 for ARP-NAK. Other op codes may be used as needed.

ar$sha: Source Hardware Address: In an L-ARP request, this is usually all zeros. In an L-ARP reply, Source Hardware Address is the label to reach ar$spa, as specified in Figure 3 below.

ar$spa: Source Protocol Address: In an L-ARP request, this field carries the sender's IP address. In an L-ARP reply, this field carries the requested IP address (which may not be the sender's IP address).

ar$tha: Target Hardware Address: In an L-ARP message, this is all zeros.

ar$tpa: Target Protocol Address: In an L-ARP request, this field carries the IP address for which the client is seeking an MPLS label.

Type: a 2-octet field defining the Type of the TVL

Length: a 2-octet field defining the Length L of the TVL

Value: an L-octet field with the Value of the TLV

5.1. Hardware Address Format

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          MPLS Label (20 bits)         |E|Z|Z|Z|    Metric ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        ... (3 octets)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Label Format in L-ARP

MPLS Label: The 20-bit label

E-bit: Entropy Label Capable: this flag indicates whether the corresponding label in the label stack can be followd by an Entropy Label. If this flag is set, the client has the option of inserting ELI and EL as specified in [RFC6790]. The client can choose not to insert ELI/EL pair. If this flag is clear, the client must not insert ELI/EL after the corresponding label.

Z: These bits are not used, and SHOULD be set to zero on sending and ignored on receipt.

Metric: The 3-octet IGP metric to ar$tha from the point of view of the L-ARP replier.

5.2. CT TLV

The CT TLV has Type (TBD) and Length 4 octets; the Value field consists of the CT attribute.

6. Security Considerations

There are many possible attacks on ARP: ARP spoofing, ARP cache poisoning and ARP poison routing, to name a few. These attacks use gratuitous ARP as the underlying mechanism, a mechanism used by L-ARP. Thus, these types of attacks are applicable to L-ARP. Furthermore, ARP does not have built-in security mechanisms; defenses rely on means external to the protocol.

It is well outside the scope of this document to present a general solution to the ARP security problem. One simple answer is to add a TLV that contains a digital signature of the contents of the ARP message. This TLV would be defined for use only in L-ARP messages, although in principle, other ARP messages could use it as well. Such an approach would, of course, need a review and approval by the Security Directorate. If approved, the type of this TLV and its procedures would be defined in this document. If some other technique is suggested, the authors would be happy to include the relevant text in this document, and refer to some other document for the full solution.

7. IANA Considerations

IANA is requested to allocate a new ARP hardware type (from registry hrd) for HTYPE-MPLS [IANA].

IANA is further requested to create a registry for Types of L-ARP Secondary Attributes. This registry should contain an entry for the CT Type Section 5.2.

8. Acknowledgments

Many thanks to Shane Amante for his detailed comments and suggestions. Many thanks to the team in Juniper prototyping this work for their suggestions on making this variant workable in the context of existing ARP implementations. Thanks too to Luyuan Fang, Alex Semenyaka and Dmitry Afanasiev for their comments and encouragement.

9. References

9.1. Normative References

[IANA] IANA, "Address Resolution Protocol (ARP) Parameters/ Hardware Types", n.d., <https://www.iana.org/assignments/arp-parameters/arp-parameters.xhtml>.

[RFC0826] Plummer, D., "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <https://www.rfc-editor.org/info/rfc826>.

[RFC2002] Perkins, C., Ed., "IP Mobility Support", RFC 2002, DOI 10.17487/RFC2002, October 1996, <https://www.rfc-editor.org/info/rfc2002>.

[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>.

[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>.

[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>.

9.2. Informative References

[I-D.kaliraj-idr-bgp-classful-transport-planes] Vairavakkalai, K., Venkataraman, N., Rajagopalan, B., Mishra, G., Khaddam, M., Xu, X., Szarecki, R. J., and D. J. Gowda, "BGP Classful Transport Planes", Work in Progress, Internet-Draft, draft-kaliraj-idr-bgp-classful-transport-planes-12, 25 August 2021, <https://www.ietf.org/archive/id/draft-kaliraj-idr-bgp-classful-transport-planes-12.txt>.

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, DOI 10.17487/RFC3107, May 2001, <https://www.rfc-editor.org/info/rfc3107>.

[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>.

[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>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[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>.

[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>.

Authors' Addresses

Kireeti Kompella
Juniper Networks
1133 Innovation Way
Sunnyvale, 94089
United States of America

Phone: +1-408-745-2000
Email: kireeti.ietf@gmail.com

Balaji Rajagopalan
Juniper Networks
Survey No.111/1 to 115/4, Wing A & B
Bangalore 560103
India

Email: balajir@juniper.net

Reji Thomas
Cohesity

Email: rejithomas.d@gmail.com

MPLS WG
Internet-Draft
Intended status: Standards Track
Expires: March 11, 2016

K. Kompella
Juniper Networks, Inc.
L. Contreras
Telefonica I+D

September 8, 2015

Resilient MPLS Rings
draft-kompella-mpls-rmr-02

Abstract

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

Requirements Language

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at 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 March 11, 2016.

Copyright Notice

Copyright (c) 2015 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

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

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

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

1.1. Definitions

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

A ring is a subgraph of G. A ring consists of a subset of n nodes {R_i, 0 <= i < n} of V. The directed edges {(R_i, R_i+1) and (R_i+1, R_i), 0 <= i < n-1} must be a subset of E (note that index arithmetic is done modulo n). We define the direction from node R_i to R_i+1 as "clockwise" (CW) and the reverse direction as "anticlockwise" (AC). As there may be several rings in a graph, we number each ring with a distinct ring ID RID.

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

Figure 1: Ring with 8 nodes

The following terminology is used for ring LSPs:

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

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

Node index: A logical numbering of nodes in a ring, from zero upto 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 connnect ring neighbors.

Bypass links: Links that connnect 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

BY: 11 bypass link

Ring Identification: The process of discovering ring nodes, ring links, link directions, and bypass 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 (AL_jk): A label allocated by R_j for RL_k in the CW (AC) direction.

P_jk (Q_jk): A Path (Resv) message sent by R_j for RL_k.

2. Motivation

A ring is the simplest topology that offers resilience. This is perhaps the main reason to lay out fiber in a ring. Thus, effective mechanisms for fast failover on rings are needed. Furthermore, there

are large numbers of rings. Thus, configuration of rings needs to be as simple as possible. Finally, bandwidth management on access rings is very important, as bandwidth is generally quite constrained here.

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

3. Theory of Operation

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

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

3.1. Provisioning

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

3.2. Ring Nodes

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

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

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

3.3. Ring Links and Directions

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

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

3.3.1. Bypass Links

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

3.4. Ring LSPs

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

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

3.5. Installing Primary LFIB Entries

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

Clearly, R_k does not act as ingress for its own LSPs. However, if these LSPs use UHP, then R_k installs LFIB entries to pop CL_k,k for packets received from R_k-1 and to pop AL_k,k for packets received from R_k+1.

3.6. Installing FRR LFIB Entries

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

3.7. Protection

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

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

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

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

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

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

2. A ring node sends protected traffic (i.e., traffic switched from CW to AC or vice versa) with TTL just large enough to reach the egress.

3. A ring node sends protected traffic with a special purpose label below the ring LSP label. A protecting node first checks for the presence of this label; if present, it means that the traffic is looping and MUST be dropped.

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

4. Autodiscovery

4.1. Overview

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

```
                 S1
                /   \
               |     R0 . . . R1          R0 has MV = 11
               |  .    \        .         R1 has MV = 10
               R7       \________ R2      All other nodes have MV = 00
   Anti-     |  .                   .  |
   clockwise |  .        Ring       .  | Clockwise
             v  .      RID = 17     .  v
                R6                  R3
                   .            .
                     R5 . . . R4
                       \      /
                        \    /
                          An
```

Figure 2: Ring with non-ring nodes and links

In what follows, we refer to a ring Type-Length-Value (TLV). This is a new TLV that contains an RID and associated flags. A ring link TLV is a ring TLV that appears as a sub-TLV of a traffic engineering TLV (TE TLV) of each link that is identified as a ring link or a bypass link. For IS-IS, the TE TLV is the extended reachability TLV; for OSPF, it is the Link TLV in the opaque TE LSA. A ring node TLV is a ring TLV that appears as a sub-TLV of a "node TLV" once for each ring this node is participating in. In IS-IS, the node TLV is the Router ID TLV; in OSPF, it is a new top-level TLV of the TE LSA. The ring direction field is ignored in ring node TLVs.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type (TBD)  |   Length = 8  |     Ring ID (4 octets) ...    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ... (RID continued)      |     Ring Flags (2 octets)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IS-IS Ring TLV Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type (TBD)          |          Length = 12          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Ring ID (4 octets)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Ring Flags (2 octets)     |         Pad (2 octets)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Pad is set to zero when sending and ignored on receipt.
```

OSPF Ring TLV Format

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|MV |RD |SP |OP |M|     MBZ     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
MV: Mastership Value
RD: Ring Direction
SP: Signaling Protocols (10 = RSVP-TE; 01 = LDP)
OP: OAM Protocols (10 = BFD; 01 = EFM)
M : Elected Master (0 = no, 1 = yes)
```

Ring Flags Format

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.

A node in promiscuous mode doesn’t advertise any ring node TLVs. If it hears exactly one non-zero RID from its IGP neighbors, it joins that ring, and sends one ring node TLV with that RID. If it hears more than one RID from its IGP neighbors, it doesn’t join any rings, and withdraws any ring node TLVs it may have advertised.

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

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. If it is the node with the lowest loopback address of all nodes with the highest mastership values, N declares itself master 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 exaclty one; if not, each node restarts timer T2 and tries again. The nodes that set their M bit should be extra careful in advertising their M bit in subsequent tries.

4.4. Ring Identification Phase

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

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

4.5. Ring Changes

The main changes to a ring are:

ring link addition;

ring link deletion;

ring node addition; and

ring node deletion.

The main goal of handling ring changes is (as much as possible) not to perturb existing ring operation. Thus, if the ring master hasn't changed, all of the above changes should be local to the point of change. Link adds just update the IGP; signaling should take advantage of the new capacity as soon as it learns. Link deletions in the case of parallel links also show up as a change in capacity (until the last link in the bundle is removed.)

The removal of the last ring link between two nodes, or the removal of a ring node is an event that triggers protection switching. In a simple ring, the result is a broken ring. However, if a ring has bypass links, then it may be able to converge to a smaller ring with protection. Details of this process will be given in a future version.

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

5. Ring Signaling

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

6. Ring OAM

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

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

7. Security Considerations

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

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, <http://www.rfc-editor.org/info/rfc2119>.

10.2. Informative References

[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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc5305>.

Authors' Addresses

Kireeti Kompella
Juniper Networks, Inc.
1194 N. Mathilda Avenue
Sunnyvale, CA 94089
USA

Email: kireeti.kompella@gmail.com

Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://people.tid.es/LuisM.Contreras/

Network Working Group
Internet-Draft
Updates: 3209, 3473, 5420
(if approved)
Intended status: Standards Track
Expires: September 10, 2015

K. Kompella
Juniper Networks
M. Hellers
LINX
R. Singh
Juniper Networks

March 9, 2015

Multi-path Label Switched Paths Signaled Using RSVP-TE
draft-kompella-mpls-rsvp-ecmp-06.txt

Abstract

This document describes extensions to Resource ReSerVation Protocol - Traffic Engineering for the set up of multi-path Traffic Engineered Label Switched Paths (LSPs) in Multi Protocol Label Switching (MPLS) and Generalized MPLS networks, i.e., LSPs that conform to traffic engineering constraints, but follow multiple independent paths from source to destination.

Status of this Memo

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

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

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

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Copyright (c) 2015 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

In selecting a protocol for setting up and signaling "tunnel" Labeled Switched Paths (LSPs) in Multi Protocol Label Switching (MPLS) and Generalized MPLS (GMPLS) networks, one first chooses whether one wants Equal Cost Multi-Path (ECMP) load balancing or Traffic Engineering (TE). For the former, one uses the Label Distribution Protocol (LDP) ([RFC5036]); for the latter, the Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE) ([RFC3209]). [Two other criteria, the need for fast protection and the desire for less configuration, are no longer the deciding factors they used to be, thanks to "IP fast reroute" ([RFC5286]) and "RSVP-TE automesh" ([RFC4972])].

This document describes how one can set up a tunnel LSP that has both ECMP and TE characteristics using RSVP-TE. The techniques described in this document can be used to create a "Multipath LSP" (MLSP) to a destination, that consists of several "sub-LSPs", each potentially taking a different path through the network to the destination. The techniques can also be used to create a single MLSP to multiple equivalent destinations (such as equidistant BGP nexthops announcing a common set of reachable addresses), such that each destination is served by one or more sub-LSPs.

There are several alternatives to choose from when considering MLSPs. One is whether the ingress Label Switching Router (LSR) computes (or otherwise obtains) the full path for each sub-LSP, or whether LSRs along the various paths can compute paths further downstream (using techniques such as "loose hop expansion", as in [RFC5152]). Another is whether the various paths that make up the MLSP have equal cost (or distance) from ingress to egress (i.e., ECMP), whether they may have differing costs. Finally, one can choose whether to terminate a multi-path LSP on a single egress or on several equivalent egresses. For now, the first of each of these alternatives is assumed; future work can explore other choices.

1.1. Terminology

The term Multipath LSP, or MLSP, will be used to denote the (logical) container LSP from an ingress LSR to one or more egress LSR(s). An MLSP is the unit of configuration and management.

An MLSP consists of one or more "sub-LSPs". A sub-LSP consists of a single path from the ingress of the MLSPs to one of its egresses. A sub-LSP is the unit of signaling of an MLSP. An Explicit Route Object (ERO) will be used to define the path of a sub-LSP.

The "downstream links" of an MLSP Z at LSR X is the union of the

downstream links of all sub-LSPs of Z traversing X. Similarly, the "upstream links" of an MLSP Z at LSR X is the union of upstream links of all sub-LSPs of Z traversing X.

The agent that takes the configuration parameters of a tunnel and computes the corresponding paths is called the Path Computation Agent (PCA). The PCA is responsible for acquiring the tunnel configuration, computing the paths of the sub-LSPs, and, if the PCA is not co-located with the ingress, informing the ingress about the tunnel and the EROs for the sub-LSPs.

1.2. Conventions used in this document

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].

2. Theory of Operation

2.1. Multi-path Label Switched Paths

An MLSP is configured with various constraints associated with TE LSPs, such as destination LSR(s), bandwidth (on a per-class basis, if desired), link colors, Shared Risk Link Groups, etc. [Auto-mesh techniques ([RFC4972]) can be used to reduce configuration; this is not described further here.] In addition, parameters specifically related to MLSPs, such as how many (or the maximum number of) sub-LSPs to create, whether traffic should be split equally across sub-LSPs or not, etc. may also be specified. This configuration lives on the PCA, which is responsible for computing the paths (i.e., the EROs) for the various sub-LSPs. The PCA informs the ingress LSR about the MLSP and the constituent sub-LSPs, including EROs and bandwidths.

The PCA uses the configuration parameters to decide how many sub-LSPs to compute for this MLSP, what paths they should take, and how much bandwidth each sub-LSP is responsible for. Each sub-LSP MUST meet all the constraints of the MLSP (except bandwidth). The bandwidths (per-class, if applicable) of all the sub-LSPs MUST add up to the bandwidth of the MLSP. A Path Computation Element ([RFC4655]) that is multi-path LSP-aware may be used as the PCA.

Having computed (or otherwise obtained) the paths of all the sub-LSPs, the ingress A then signals the MLSP by signaling all the individual sub-LSPs across the MPLS/GMPLS network. To do this, the ingress first picks an MLSP ID, a 16-bit number that is unique in the context of the ingress. This ID is used in an ASSOCIATION object that is placed in each sub-LSP to let all transit LSRs know that the sub-LSPs belong to the same MLSP.

If multiple sub-LSPs of the same MLSP pass through LSR Y, and Y has downstream links YP, YQ and YR for the various sub-LSPs, then Y has to load balance incoming traffic for the MLSP across the three downstream links in proportion to the sum of the bandwidths of the sub-LSPs going to each downstream (see Figure 1).

One must distinguish carefully between the signaled bandwidth of a sub-LSP, a static value capturing the expected or maximum traffic on the sub-LSP, and the instantaneous traffic received on a sub-LSP, a constantly varying quantity. Suppose there are three sub-LSPs traversing Y, with bandwidths 10Gbps, 20Gbps and 30Gbps, going to P, Q and R respectively. Suppose further Y receives some traffic over each of these sub-LSPs. Y must balance this received traffic over the three downstream links YP, YQ and YR in the ratio 1:2:3.

2.2. ECMP

```
                        ------------- M -----
                       /                     \
                      /           --- P --     \
                     /           /         \    \
                   A --- X --- Y --- Q --- T --- B
                          \     \              / |
                           \     --- R -------   |
                            \                    |
                             ------- S ----------
```

An example network illustrating ECMP. Assume that paths AMB, AXYPTB, AXYQTB, AXYRB and AXSB all have the same path length (cost).

Figure 1: Example Network Topology

In an IP or LDP network, incoming traffic arriving at A headed for B will be split equally between M and X at A. Similarly, traffic for B arriving at Y will be split equally among P, Q and R. If the traffic arriving at A for B is 120Gbps, then the AMB path will carry 60Gbps, the paths AXYPTB, AXYQTB and AXYRB will each carry 10Gbps, and the AXSB path will carry 30Gbps. We’ll call this "IP-style" load balancing.

Note: all load balancing is subject to the overriding requirement of mapping the same "flow" to the same downstream. (What constitutes a "flow" is beyond the scope of this document.) This requirement takes precedence over all attempts to balance traffic among downstreams. Thus, the statements above (e.g., "the AMB path will carry 60Gbps") are to be interpreted as ideal targets, not hard requirements, of load balancing.

One can simulate the IP or LDP ECMP behavior with TE-based ECMP by creating an MLSP with five sub-LSPs S1 through S5 taking paths AMB, AXYPTB, AXYQTB, AXYRB and AXSB, with bandwidths 60Gbps, 10Gbps, 10Gbps, 10Gbps and 30Gbps, respectively.

With such an arrangement, the MB link carries 60Gbps while the RB link carries just 10Gbps. If one wishes instead to carry equal amounts of traffic on the links incoming to B, then one could arrange the sub-LSPs S1 to S5 to have bandwidths 30Gbps, 15Gbps, 15Gbps, 30Gbps and 30Gbps, respectively. In this case, the bandwidth on each of the four links going to B is 30Gbps, illustrating some of the capabilities of TE-based ECMP.

Staying with this example, A has one sub-LSP of bandwidth 30Gbps to M and four sub-LSPs of total bandwidth 90Gbps through X. Thus, A should

load balance traffic in the ratio 1:3 between the AM and the AX links. Similarly, X has three sub-LSPs of total bandwidth 60Gbps to Y and one sub-LSP of bandwidth 30Gbps to S, so X should load balance traffic 2:1 between Y and S. Y has a sub-LSP of bandwidth 15Gbps to each of P and Q and one sub-LSP of bandwidth 30Gbps to R, so Y should load balance traffic 1:1:2 among P, Q and R, respectively. Thus, in general, TE-based ECMP does not assume equal distribution of traffic among downstream LSRs, unlike IP- or LDP-style ECMP.

```
                                        ---U---
                                       |       |
                         --L--   --P-- | --V-- |
                        /     \ /     \|/     \|
                       A       S---Q---T---W---B
                        \     / \     /|\     /|
                         --M--   --R-- | --X-- |
                                       |       |
                                        ---Y---
```

Another example network illustrating 30 ECMP paths between A and B.

Figure 2: Another Network Topology

In Figure 2, there are potentially 2x3x5=30 ECMP paths between A and B. With IP or LDP, exploiting all these paths is straightforward, and doesn’t need a lot of state. With an MLSP as seen so far, this would require 30 sub-LSPs to achieve equivalent load balancing. This suggests that a different approach is needed to efficiently achieve IP-style load balancing with TE LSPs. To this end, we introduce the notion of "equi-bandwidth" (EB) sub-LSPs and EB MLSPs. A sub-LSP is equi-bandwidth if its "E" bit is set (see Section 3.1.1). An MLSP is equi-bandwidth if all of its sub-LSPs are equi-bandwidth.

If a set of EB sub-LSPs of the same MLSP traverse an LSR S, say to downstream links SP, SQ and SR, then S MUST attempt to load balance traffic received on these EB sub-LSPs equally among the links SP, SQ and SR, independent of how many sub-LSPs go over each of these links. Furthermore, S MUST redistribute traffic received from each of its upstream LSRs, and SHOULD redistribute all traffic received from upstream as a whole. One can do the former by signaling the same label to each of its upstream LSRs; one can do the latter by signaling the same label to all upstream LSRs (see Section 3.2). For example, in Figure 2, if L sends 12Gbps of traffic to S and M sends 18Gbps to S, S can redistribute L’s traffic by sending 4Gbps to each of P, Q and R; and can similarly send 6Gbps of M’s traffic to each of P, Q and R. Alternatively, S can load balance the aggregate 30Gbps of traffic received from L and M to each of P, Q and R, thus sending 10Gbps to each. EB sub-LSPs have an added benefit of not requiring

unequal load balancing across links, which may pose problems for some hardware.

Given the notion of EB sub-LSPs and EB MLSPs, A can signal an EB MLSP Z comprised of five EB sub-LSPs E1 through E5 with the following paths: ALSPTUB, AMSQTVB, ALSRTWB, AMSPTXB and ALSQTYB (respectively). Then, A has two downstream links for the five sub-LSPs, AL and AM, between which A will load balance equally. Similarly, S has three downstream links, SP, SQ and SR; and T has five downstreams, TU, TV, TW, TX and TY. Thus the load balancing behavior of the MLSP will replicate IP load balancing. The state required for an EB MLSP to achieve IP-style load balancing is somewhat greater than for LDP LSPs, but significantly less than that for multiple "regular" TE LSPs, or for a non-EB MLSP.

2.3. Discussion

Some of the power of TE-based ECMP was illustrated in the above examples. Another is ability to request that all sub-LSPs avoid links colored red. If in the example network in Figure 1, the QT link is colored red but all other links are not, then there are four ECMP paths that satisfy these constraints, and the traffic distribution among them will naturally be different than it would without the link color constraint.

One can also ask whether an MLSP with sub-LSPs is any better than N "regular" LSPs from the same ingress to the same egress. Here are some benefits of an MLSP:

1. With an MLSP, there is a single entity to provision, manage and monitor, versus N separate entities in the case of LSPs. A consequence of this is that with an MLSP, changes in topology can be dealt with easily and autonomously by the ingress LSR, by adding, changing or removing sub-LSPs to rebalance traffic, while maintaining the same TE constraints. With individual LSPs, such changes would require changes in configuration, and thus are harder to automate.

2. An ingress LSR, knowing that an MLSP is for load balancing, can decide on an optimum number of sub-LSPs, and place them appropriately across the network to optimize load balancing. On the other hand, an ingress LSR asked to create N independent LSPs will do so without regard to whether N is a good number of equal cost paths, and, more importantly, may place several of the N LSPs on the same path, defeating the purpose of load balancing.

3. The EB sub-LSP mechanism will, in many cases, result in far fewer sub-LSPs than independent LSPs and thus less control plane state.

4. Finally, an MLSP will usually have less data plane state than N independent LSPs: whenever multiple sub-LSPs traverse a link, a single label will be used for all of them, whereas if multiple LSPs traverse a link, each will need a separate label.

2.4. The Capabilities of TE-based Load Balancing

Definition: Let G=(V, E) be a directed graph (or network), and let A and B in V be two nodes in G. Let T be the traffic arriving at A destined for B. T is said to be "IP-style" load balanced if for every node X on a shortest path from A to B, the portion of T arriving at X is split equally among all nodes Yi that are adjacent to X and are on a shortest path from X to B.

Theorem: An MLSP can accurately mimic IP-style load balancing between any two nodes in any network.

Proof: left to the reader.

Corollary: MLSPs provide a strictly more powerful load balancing mechanism than IP-style load balancing.

3. Operation of MLSPs

3.1. Signaling MLSPs

Sub-LSPs of an MLSP are tied together using ASSOCIATION objects. ASSOCIATION objects have a new Association Type for MLSPs (TBD). The Association ID is chosen by the ingress of the MLSP; the Association Source is the loopback address of the ingress of the MLSP. All sub-LSPs containing an ASSOCIATION object with a given Association Source and Type belong to the same MLSP.

3.1.1. Indicating Equi-bandwidth (EB) nature

A sub-LSP is considered equi-bandwidth if its Path message carries the optional LSP_ATTRIBUTES object ([RFC5420]) with an EBC (equi-bandwidth capability) flag in the Attribute Flags TLV. The bit number for the EBC flag is yet to be assigned by IANA.

3.2. Label Allocation

A LSR S that receives Path messages for several sub-LSPs of the same MLSP from the same upstream LSR SHOULD allocate the same label for all the sub-LSPs. This simplifies load balancing for the aggregate traffic on those sub-LSPs. If the sub-LSPs are EB sub-LSPs, then S SHOULD allocate the same label for all EB sub-LSPs of the same MLSP that pass through S, regardless of which upstream LSR they come from. This allows S to load balance the aggregate traffic received on the MLSP, as all the MLSP traffic arrives at S with the same label. However, an LSR that can achieve the load balancing requirements independent of label allocation strategies is free to do so.

3.3. Bandwidth Accounting

Since MLSPs are traffic engineered, there needs to be strict bandwidth accounting, or admission control, on every link that an MLSP traverses. For non-EB sub-LSPs, this is straightforward, and analogous to regular TE LSPs. However, for EB sub-LSPs, two new procedures are needed, one for signaling bandwidth, and the other for admission control. First, for a given MLSP Z, an LSR X MUST ensure (via signaling) that the total incoming bandwidth of EB sub-LSPs of MLSP Z is divided equally among all the downstream links of X which at least one of the EB sub-LSPs traverses. Second, LSR X MUST ensure that, for each upstream link of X, there is sufficient bandwidth to accommodate all EB sub-LSPs of MLSP Z that traverse that link.

Let’s take the example of Figure 2, with MLSP Z having five EB sub-LSPs E1 to E5, and say that MLSP Z is configured with a bandwidth of 30Gbps. Here are some of the steps involved.

1. LSR A, being the ingress, has no upstream links. A has two downstream links, AL and AM. Three EB sub-LSPs of MLSP Z traverse AL, and two traverse AM. A MUST signal a total of 15Gbps for the sub-LSPs to L, and a total of 15Gbps for the sub-LSPs to M. The required bandwidth may be divided up among the sub-LSPs to L (similarly, to M) in any manner so long as the total is 15Gbps. For example, A can signal sub-LSP E1 with 15Gbps, and sub-LSPs E3 and E5 with 0 bandwidth.

2. LSR L has one upstream link AL with three EB sub-LSPs with a total bandwidth of 15Gbps. L MUST ensure that 15Gbps is available for the AL link. If this bandwidth is not available, L MUST send a PathErr on ALL of the EB sub-LSPs on the AL link. Let’s assume that the AL link has sufficient bandwidth.

3. Next, it is up to L to decide how to divide the incoming 15Gbps among the three downstream EB sub-LSPs to S. Say L signals sub-LSP E1 with 15Gbps, and the others with 0 bandwidth.

4. LSR S has two upstream links: LS with three EB sub-LSPs with a total bandwidth of 15Gbps, and MS with two EB sub-LSPs with a total bandwidth of 15Gbps. S MUST ensure that 15Gbps is available for each of the LS and MS links. S has thus a total incoming bandwidth of 30Gbps on MLSP Z. S has to divide this equally among its downstream links SP, SQ and SR, yielding 10Gbps each. S MUST ensure that the total bandwidth requested on the SP link for sub-LSPs E1 and E4 is 10Gbps. S may choose to signal these sub-LSPs with 5Gbps each. Similarly for the SQ and SR links.

There are two important points to note here. One is that the bandwidth reservation (TSpec) for a given EB sub-LSP can (and usually will) change hop-by-hop. The second is that as new EB sub-LSPs are signaled for an MLSP, the bandwidth reservations for existing EB sub-LSPs belonging to the same MLSP may have to be updated. To minimize these updates, it is RECOMMENDED that the first EB sub-LSP on a link be signaled with the total required bandwidth (as far as is known), and later sub-LSPs on the same link be signaled with 0 bandwidth.

3.4. MLSP Data Plane Actions

Traffic intended to be sent over an MLSP is determined at the ingress LSR by means outside the scope of this document, and at transit LSRs by the label(s) assigned by the transit LSR to its upstream LSRs. In the case of non-EB sub-LSPs, this traffic is load balanced across downstream links in the ratio of the bandwidths of the sub-LSPs that comprise the MLSP. In the case of EB sub-LSPs, the traffic belonging to an MLSP from an upstream LSR (or better still, the aggregate

traffic for the MLSP from all upstream LSRs) is load balanced equally among all downstream links.

As noted above, the overriding concern is that flows are mapped to the same downstream link (except when the MLSP or some constituent sub-LSPs are changing); this is typically done by hashing fields that define a flow, and mapping hash results to different downstream LSRs. Hash-based load balancing typically assumes that the numbers of flows is sufficiently large and the bandwidth per flow is reasonably well-balanced so that the results of hashing yields reasonable traffic distribution.

Entropy labels ([RFC6790] and [RFC6391]) can be used to improve load balancing at intermediate nodes.

4. Manageability

TBD

5. Security Considerations

This document introduces no new security concerns in the setup and signaling of LSPs using RSVP-TE, or in the use of the RSVP protocol. [RFC2205] specifies the message integrity mechanisms for RSVP signaling. These mechanisms apply to RSVP-TE signaling of MLSPs described in this document, and are highly recommended pending newer integrity mechanisms for RSVP.

6. Acknowledgments

The author would like to thank the Routing Protocol group at Juniper Networks for their questions, comments and encouragement for this proposal. While many participated, special thanks go to Yakov Rekhter, John Drake and Rahul Aggarwal. Many thanks too to John for suggesting the use of ASSOCIATION objects.

7. IANA Considerations

IANA is requested to assign the following:

A new Association Type for MLSP. This Association Type is to be used for ASSOCIATION objects with C-Type 1 (IPv4 Source) and 2 (IPv6 Source).

A new flag in the Attribute Flags TLV in the LSP_ATTRIBUTES object ([RFC5420]: a bit number for the EBC (equi-bandwidth capability) to indicate that a specific sub-LSP is an equi-bandwidth sub-LSP.

8. References

8.1. Normative References

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

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC4875] Aggarwal, R., Papadimitriou, D., and S. Yasukawa, "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, May 2007.

[RFC5420] Farrel, A., Papadimitriou, D., Vasseur, JP., and A. Ayyangarps, "Encoding of Attributes for MPLS LSP Establishment Using Resource Reservation Protocol Traffic Engineering (RSVP-TE)", RFC 5420, February 2009.

8.2. Informative References

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006.

[RFC4972] Vasseur, JP., Leroux, JL., Yasukawa, S., Previdi, S., Psenak, P., and P. Mabbey, "Routing Extensions for Discovery of Multiprotocol (MPLS) Label Switch Router (LSR) Traffic Engineering (TE) Mesh Membership", RFC 4972, July 2007.

[RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.

[RFC5152] Vasseur, JP., Ayyangar, A., and R. Zhang, "A Per-Domain Path Computation Method for Establishing Inter-Domain Traffic Engineering (TE) Label Switched Paths (LSPs)", RFC 5152, February 2008.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5286] Atlas, A. and A. Zinin, "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, September 2008.

[RFC6391] Bryant, S., Filsfils, C., Drafz, U., Kompella, V., Regan, J., and S. Amante, "Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network", RFC 6391, November 2011.

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

Authors' Addresses

Kireeti Kompella
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US

Email: kireeti.kompella@gmail.com

Mike Hellers
LINX

Email: mikeh@linx.net

Ravi Singh
Juniper Networks

Email: ravis@juniper.net

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 9, 2016

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

August 8, 2015

Bidirectional Forwarding Detection (BFD) Directed Return Path
draft-mirsky-mpls-bfd-directed-04

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on February 9, 2016.

Copyright Notice

Copyright (c) 2015 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

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

This document defines the extension to LSP Ping [RFC4379], BFD Reverse Path TLV, and proposes that it to be used to instruct the egress BFD peer to use explicit path for its BFD control packets associated with the particular BFD session. The TLV will be allocated from the TLV and sub-TLV registry defined by RFC 4379 [RFC4379]. As a special case, forward and reverse directions of the BFD session can form bi-directional co-routed associated channel.

1.1. Conventions used in this document

1.1.1. Terminology

BFD: Bidirectional Forwarding Detection

MPLS: Multiprotocol Label Switching

LSP: Label Switching Path

LoC: Loss of Continuity

1.1.2. Requirements Language

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

2. Problem Statement

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

- o detection of a failure on the reverse path cannot reliably be interpreted as bi-directional defect and thus trigger, for example, protection switchover of the forward direction;

- o if a failure of the reverse path had been ignored, the ingress node would not receive indication of forward direction failure from its egress peer.

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

3. Direct Reverse BFD Path

3.1. Case of MPLS Data Plane

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

3.1.1. BFD Reverse Path TLV

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

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

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

Figure 1: BFD Reverse Path TLV

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

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

Reverse Path field contains a sub-TLV. Any Target FEC sub-TLV, already or in the future defined, from IANA sub-registry Sub-TLVs for TLV Types 1, 16, and 21 of MPLS LSP Ping Parameters registry MAY be

used in this field. Only one sub-TLV MUST be included in the Reverse Path TLV. If more than one sub-TLVs are present in the Reverse Path TLV, then only the first sub-TLV MUST be used and the rest MUST be silently discarded.

If the egress LSR cannot find path specified in the Reverse Path TLV it MUST send Echo Reply with the received Reverse Path TLV and set the return code to "Failed to establish the BFD session. The specified reverse path was not found" Section 3.4. The egress LSR MAY establish the BFD session over IP network according to [RFC5884].

3.1.2. Static and RSVP-TE sub-TLVs

When explicit path on MPLS data plane set either as Static or RSVP-TE LSP respective sub-TLVs defined in [RFC7110] identify explicit return path.

3.1.3. Segment Routing Tunnel sub-TLV

In addition to Static and RSVP-TE, Segment Routing with MPLS data plane can be used to set explicit path. In this case a new sub-TLV is defined in this document as presented in Figure 2.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  SegRouting MPLS sub-TLV Type |          Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Label Entry 1                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Label Entry 2                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Label Entry N                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Segment Routing MPLS Tunnel sub-TLV

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

The egress LSR MUST use the Value field as label stack for BFD control packets for the BFD session identified by source IP address and value in BFD Discriminator TLV.

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

3.2. Case of IPv6 Data Plane

IPv6 can be data plane of choice for Segment Routed tunnels [I-D.previdi-6man-segment-routing-header]. In such networks the BFD Reverse Path TLV described in Section 3.1.1 can be used as well. To specify reverse path of a BFD session in IPv6 environment the BFD Discriminator TLV MUST be used along with the BFD Reverse Path TLV. The BFD Reverse Path TLV in IPv6 network MUST include sub-TLV.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  SegRouting IPv6 sub-TLV Type |          Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                          IPv6 Prefix                          |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                          IPv6 Prefix                          |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Segment Routing IPv6 Tunnel sub-TLV

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

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

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

Initiator, i.e. ingress LSR, MAY include FECs corresponding to some or all of segments imposed in the label stack by the ingress LSR to communicate the segments traversed. "

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

3.4. Return Codes

This document defines the following Return Codes:

- o "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 LSR, 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 LSR . (Section 3.1.1)

4. Use Case Scenario

In network presented in Figure 4 node A monitors two tunnels to node H: A-B-C-D-G-H and A-B-E-F-G-H. To bootstrap BFD session to monitor the first tunnel, node A MUST include BFD Discriminator TLV with Discriminator value foobar-1 and MAY include BFD Reverse Path TLV that references H-G-D-C-B-A tunnel. To bootstrap BFD session to monitor the second tunnel, node A MUST include BFD Discriminator TLV with Discriminator value foobar-2 [I-D.ietf-bfd-rfc5884-clarifications] and MAY include BFD Reverse Path TLV that references H-G-F-E-B-A tunnel.

```
        C---------D
        |         |
A-------B         G-----H
        |         |
        E---------F
```

Figure 4: Use Case for BFD Reverse Path TLV

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

5. IANA Considerations

5.1. TLV

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

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

Table 1: New BFD Reverse Type TLV

5.2. Sub-TLV

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

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

Table 2: New Segment Routing Tunnel sub-TLV

5.3. Return Codes

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

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

Table 3: New Return Code

6. Security Considerations

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

7. Acknowledgements

8. Normative References

[I-D.ietf-bfd-rfc5884-clarifications]
Govindan, V., Rajaraman, K., Mirsky, G., Akiya, N., and S. Aldrin, "Clarifications to RFC 5884", draft-ietf-bfd-rfc5884-clarifications-02 (work in progress), June 2015.

[I-D.kumarkini-mpls-spring-lsp-ping]
Kumar, N., Swallow, G., Pignataro, C., Akiya, N., Kini, S., Gredler, H., and M. Chen, "Label Switched Path (LSP) Ping/Trace for Segment Routing Networks Using MPLS Dataplane", draft-kumarkini-mpls-spring-lsp-ping-04 (work in progress), July 2015.

[I-D.previdi-6man-segment-routing-header]
Previdi, S., Filsfils, C., Field, B., Leung, I., Vyncke, E., and D. Lebrun, "IPv6 Segment Routing Header (SRH)", draft-previdi-6man-segment-routing-header-07 (work in progress), July 2015.

[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>.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, DOI 10.17487/RFC4379, February 2006, <http://www.rfc-editor.org/info/rfc4379>.

[RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <http://www.rfc-editor.org/info/rfc5586>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7110>.

Authors' Addresses

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Jeff Tantsura
Ericsson

Email: jeff.tantsura@ericsson.com

Ilya Varlashkin
Google

Email: Ilya@nobulus.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2016

G. Mirsky
S. Ruffini
E. Gray
Ericsson
J. Drake
Juniper Networks
S. Bryant
Cisco Systems
A. Vainshtein
ECI Telecom
July 3, 2015

Residence Time Measurement in MPLS network
draft-mirsky-mpls-residence-time-07

Abstract

This document specifies G-ACh based Residence Time Measurement and how it can be used by time synchronization protocols being transported over MPLS domain.

Residence time is the variable part of propagation delay of timing and synchronization messages and knowing what this delay is for each message allows for a more accurate determination of the delay to be taken into account in applying the value included in a PTP event message.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 4, 2016.

Copyright Notice

Copyright (c) 2015 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

Time synchronization protocols, Network Time Protocol version 4 (NTPv4) [RFC5905] and Precision Time Protocol (PTP) Version 2 [IEEE.1588.2008] can be used to synchronize clocks across network domain. Measurement of the time a PTP event message spends traversing a node (using precise times of receipt at an ingress interface and transmission at an egress interface), called Residence Time, can be used to improve the accuracy of clock synchronization. This document defines new Generalized Associated Channel (G-ACh) that can be used in Multi-Protocol Label Switching (MPLS) network to measure Residence Time over Label Switched Path (LSP). Mechanisms for transport of time synchronization protocol packets over MPLS are out of scope in this document.

Though it is possible to use RTM over LSPs instantiated using LDP such scenarios are outside the scope of this document. The scope of this document is on LSPs instantiated using RSVP-TE [RFC3209] because the LSP's path can be determined.

1.1. Conventions used in this document

1.1.1. Terminology

MPLS: Multi-Protocol Label Switching

ACH: Associated Channel

TTL: Time-to-Live

G-ACh: Generic Associated Channel

GAL: Generic Associated Channel Label

NTP: Network Time Protocol

ppm: parts per million

PTP: Precision Time Protocol

LSP: Label Switched Path

LSR: Label Switching Router

OAM: Operations, Administration, and Maintenance

RRO: Record Route Object

RSO: RTM Set Object

RTM: Residence Time Measurement

IGP: Internal Gateway Protocol

1.1.2. Requirements Language

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

2. Residence Time Measurement

Packet Loss and Delay Measurement for MPLS Networks [RFC6374] can be used to measure one-way or two-way end-to-end propagation delay over LSP or PW. But these metrics are insufficient for use in some applications, for example, time synchronization across a network as defined in the Precision Time Protocol (PTP). PTPv2 [IEEE.1588.2008] uses "residence time", the time it takes for a PTPv2 event packet to transit a node. Residence times are accumulated in the correctionField of the PTP event messages, as defined in [IEEE.1588.2008], or of the associated follow-up messages (or Delay_Resp message associated with the Delay_Req message) in case of two-step clocks (detailed discussion in Section 7). The residence time values are specific to each output PTP port and message.

IEEE 1588 uses this residence time to correct the propagated time, effectively making these nodes transparent.

This document proposes mechanism to accumulate packet residence time from all LSRs that support the mechanism across a particular LSP. The values accumulated in scratchpad fields of MPLS RTM messages can be used by the last RTM-capable LSR on an LSP to update the correctionField of the corresponding PTP event packet prior to performing the usual PTP processing.

3. G-ACh for Residence Time Measurement

RFC 5586 [RFC5586] and RFC 6423 [RFC6423] extended applicability of PW Associated Channel (ACH) [RFC5085] to LSPs. G-ACh provides a mechanism to transport OAM and other control messages. Processing by arbitrary transit LSRs can be triggered through controlled use of the Time-to-Live (TTL) value. In a way that is analogous to PTP

operations, the packet residence time can be handled by the RTM capable node either as "one-step clock" or as a "two-step clock".

The packet format for Residence Time Measurement (RTM) is presented in Figure 1

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 1|Version|   Reserved    |          RTM Channel          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                      Scratch Pad                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type               |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Value                             |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: RTM G-ACh packet format for Residence Time Measurement

- o First four octets are defined as G-ACh Header in [RFC5586]

- o The Version field is set to 0, as defined in RFC 4385 [RFC4385].

- o The Reserved field MUST be set to 0 on transmit and ignored on receipt.

- o The RTM G-ACh field, value to be allocated by IANA, identifies the packet as such.

- o The Scratch Pad field is 8 octets in length. The first RTM-capable LSR MUST initialize the Scratch Pad field, it SHOULD set it to zero value. The Scratch Pad is used to accumulate the residence time spent in each RTM capable LSR transited by the packet on its path from ingress LSR to egress LSR. Its format is IEEE double precision and its units are nanoseconds. Note: depending on one-step or two-step operation (Section 7), the residence time might be related to the same packet carried in the Value field or to a packet carried in a different RTM packet.

- o The Type field identifies the type of Value that the TLV carries. IANA will be asked to create a sub-registry in Generic Associated Channel (G-ACh) Parameters Registry called "MPLS RTM TLV Registry".

- o The Length field contains the number of octets of the Value field.

- o The optional Value field may be used to carry a packet of a given time synchronization protocol. If packet data is carried in the RTM message, then this is identified by Type accordingly. The data MAY be NTP [RFC5905] or PTP [IEEE.1588.2008]. It is important to note that the packet may be authenticated or encrypted and carried over MPLS LSP edge to edge unchanged while residence time being accumulated in the Scratch Pad field. Sub-TLVs MAY be included in the Value field.

- o The TLV MUST be included in the RTM message, even if the length of the Value field is zero.

3.1. PTP Packet Sub-TLV

Figure 2 presents format of a PTP sub-TLV that MUST be precede every PTP packet carried in RTM TLV.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Flags                         |PTPType|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Port ID                            |
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |           Sequence ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: PTP Sub-TLV format

where Flags field has format

```
 0                   1                   2
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|                     Reserved                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Flags field format of PTP Packet Sub-TLV

- o The Type field identifies PTP sub-TLV defined in the Table 19 Values of messageType field in [IEEE.1588.2008].

- o The Length field of the PTP sub-TLV contains the number of octets of the Value field and MUST be 20.

- o The Flags field currently defines one bit, the S-bit, that defines whether or not the current message has been processed by a 2-step node, where the flag is cleared if the message has been handled exclusively by 1-step nodes and there is no follow-up message, and set if there has been at least one 2-step node and a follow-up message is forthcoming.

- o The PTPType indicates the type of PTP packet carried in the TLV. PTPType is the messageType field of the PTPv2 packet whose values are defined in the Table 19 [IEEE.1588.2008].

- o The 10 octets long Port ID field contains the identity of the source port.

- o The Sequence ID is the sequence ID of the PTP message carried in the Value field of the message.

4. Control Plane Theory of Operation

The operation of RTM depends upon TTL expiry to deliver an RTM packet from one RTM capable interface to the next along the path from ingress LSR to egress LSR. This means that an LSR with RTM capable interfaces MUST be able to compute a TTL which will cause the expiry of an RTM packet at the next LSR with RTM capable interfaces.

4.1. RTM Capability

Note that RTM capability of a node is with respect to the pair of interfaces that will be used to forward an RTM packet. In general, the ingress interface of this pair must be able to capture the arrival time of the packet and encode it in some way such that this information will be available to the egress interface.

The supported modes (1-step verses 2-step) of any pair of interfaces is then determined by the capability of the egress interface. In both cases, the egress interface implementation MUST be able to determine the precise departure time of the same packet and determine from this, and the arrival time information from the corresponding ingress interface, the difference representing the residence time for the packet.

An interface with the ability to do this and update the associated ScratchPad in real-time (i.e. while the packet is being forwarded) is said to be 1-step capable.

Hence while both ingress and egress interfaces are required to support RTM, for the pair to be RTM-capable, it is the egress interface that determines whether or not the node is 1-step or 2-step capable with respect to the interface-pair.

The RTM capability used in the sub-TLV shown in Figure 4 is thus associated with the egress port of the node making the advertisement, while the ability of any pair of interfaces that includes this egress interface to support any mode of RTM depends on the ability of that interface to record packet arrival time in some way that can be conveyed to and used by that egress interface.

When an LSR uses an IGP to carry the RTM capability sub-TLV, the sub-TLV MUST reflect the RTM capability (1-step or 2-step) associated with egress interfaces and MUST NOT propagate this sub-TLV in IGP LSAs sent from a router which describe a particular interface that does not support the same capability for RTM messages it receives.

4.2. RTM Capability Sub-TLV

The format for the RTM Capabilities sub-TLV is presented in Figure 4

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type(TBA5)          |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| RTM |                       Reserved                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: RTM Capability sub-TLV

- o Type value will be assigned by IANA from appropriate registries.
- o Length MUST be set to 4.
- o RTM (capability) - is a three-bit long bit-map field with values defined as follows:
  - * 0b001 - one-step RTM supported;
  - * 0b010 - two-step RTM supported;
  - * 0b100 - reserved.
- o Reserved field must be set to all zeroes on transmit and ignored on receipt.

[RFC4202] explains that the Interface Switching Capability Descriptor describes switching capability of an interface. For bi-directional links, the switching capabilities of an interface are defined to be the same in either direction. I.e., for data entering the node through that interface and for data leaving the node through that interface". That principle SHOULD be applied when a node advertises RTM Capability.

A node that supports RTM MUST be able to act in two-step mode and MAY also support one-step RTM mode. Detailed discussion of one-step and two-step RTM modes in Section 7.

4.3. RTM Capability Advertisement in OSPFv2

The capability to support RTM on a particular link advertised in the OSPFv2 Extended Link Opaque LSA [I-D.ietf-ospf-prefix-link-attr] as RTM Capability sub-TLV, presented in Figure 4, of the OSPFv2 Extended Link TLV.

Type value will be assigned by IANA from the OSPF Extended Link TLV Sub-TLVs registry that will be created per [I-D.ietf-ospf-prefix-link-attr] request.

4.4. RTM Capability Advertisement in OSPFv3

The capability to support RTM on a particular link in OSPFv3 can be advertised by including an RTM Capability sub-TLV defined in Section 4.3 in the following TLVs defined in [I-D.ietf-ospf-ospfv3-lsa-extend] Intra-Area-Prefix TLV, IPv6 Link-Local Address TLV, or IPv4 Link-Local Address TLV when these are included in E-Link-LSA.

4.5. RTM Capability Advertisement in IS-IS

The RTM capability logically belongs to a group of parameters characterized as "generic information not directly related to the operation of the IS-IS protocol" [RFC6823]. Hence the capability to process RTM messages can be advertised by including RTM Capability sub-TLV in GENINFO TLV [RFC6823].

With respect to the Flags field of the GENINFO TLV:

- o The S bit MUST be cleared to prevent the RTM Capability sub-TLV from leaking between levels.

- o The D bit of the Flags field MUST be cleared as required by [RFC6823].

- o The I bit and the V bit MUST be set accordingly depending on whether RTM capability being advertised for IPv4 or IPv6 interface of the node.

Application ID (TBA6) will be assigned from the Application Identifiers for TLV 251 IANA registry. The RTM Capability sub-TLV, presented in Figure 4, MUST be included in GENINFO TLV in Application Specific Information.

4.6. RSVP-TE Control Plane Operation to Support RTM

Throughout this document we refer to an LSR as RTM capable LSR when at least one of its interfaces is RTM capable. Figure 5 provides an example of relationship between roles a network element may have in PTP over MPLS scenario and RTM capability:

```
 -----     -----     -----     -----     -----     -----     -----
 | A |-----| B |-----| C |-----| D |-----| E |-----| F |-----| G |
 -----     -----     -----     -----     -----     -----     -----
```

Figure 5: RTM capable roles

- o A is a Boundary Clock with its egress port in Master state. Node A transmits PTP messages;
- o B is the ingress LER for the MPLS LSP and is not RTM capable;
- o C is the first RTM capable LSR; it initializes the RTM Scratch Pad field and encapsulates PTP messages in the RTM ACH; the transmitted Scratch Pad information includes the residence time measured by C;
- o D is a transit LSR that is not RTM capable; it passes along the RTM ACH encapsulated PTP message unmodified;
- o E is the last RTM capable LSR; it updates the Correction field of the PTP message with the value in the Scratch Pad field of the RTM ACH, and removes the RTM ACH encapsulation;
- o F is the egress LER for the MPLS LSP and is not RTM capable;
- o G is a Boundary Clock with its ingress port in Slave state. Node G receives PTP messages.

An ingress LSR that is configured to perform RTM along a path through an MPLS network to an egress LSR verifies that the selected egress LSR has an interface that supports RTM via the egress LSR’s advertisement of the RTM Capability sub-TLV. In the Path message

that the ingress LSR uses to instantiate the LSP to that egress LSR it places initialized Record Route Object (RRO) [RFC3209] and RTM Set Object (RSO) [Section 4.7], which tell the egress LSR that RTM is requested for this LSP.

In the Resv message that the egress LSR sends in response to the received Path message, it includes initialized RRO and RSO. The RSO contains an ordered list, from egress LSR to ingress LSR, of the RTM capable LSRs along the LSP's path. Each such LSR will use the ID of the first LSR in the RSO in conjunction with the RRO to compute the hop count to its downstream LSR with reachable RTM capable interface. It will also insert its ID at the beginning of the RTM Set Object before forwarding the Resv upstream.

After the ingress LSR receives the Resv, it MAY begin sending RTM packets to the first RTM capable LSR on the LSP's path. Each RTM packet has its Scratch Pad field initialized and its TTL set to expire on that first subsequent RTM capable LSR.

It should be noted that RTM can also be used for LSPs instantiated using [RFC3209] in an environment in which all interfaces in an IGP support RTM. In this case the RSO MAY be omitted.

4.7. RTM_SET Object

RTM capable interfaces can be recorded via RTM_SET object (RSO). The RTM Set Class is TBA7. This document defines one C_Type, Type TBA8 RTM Set. The RTM_SET object format presented in Figure 6

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                          Sub-objects                          ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: RTM Set object format

The contents of a RTM_SET object are a series of variable-length data items called sub-objects. The sub-objects are defined in Section 4.7.1 below.

The RSO can be present in both RSVP Path and Resv messages. If a Path message contains multiple RSOs, only the first RSO is meaningful. Subsequent RSOs SHOULD be ignored and SHOULD NOT be propagated. Similarly, if in a Resv message multiple RSOs are encountered following a FILTER_SPEC before another FILTER_SPEC is

encountered, only the first RSO is meaningful. Subsequent RSOs SHOULD be ignored and SHOULD NOT be propagated.

4.7.1. RSO Sub-objects

The RTM Set object contains an ordered list, from egress LSR to ingress LSR, of the RTM capable LSRs along the LSP's path.

The contents of a RTM_SET object are a series of variable-length data items called sub-objects. Each sub-object has its own Length field. The length contains the total length of the sub-object in bytes, including the Type and Length fields. The length MUST always be a multiple of 4, and at least 8 (smallest IPv4 sub-object).

Sub-objects are organized as a last-in-first-out stack. The first -out sub-object relative to the beginning of RSO is considered the top. The last-out sub-object is considered the bottom. When a new sub-object is added, it is always added to the top.

Three kinds of sub-objects for RSO are currently defined.

4.7.1.1. IPv4 Sub-object

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type     |     Length    |             Flags             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       IPv4 address                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: IPv4 sub-object format

Type

0x01 IPv4 address

Length

The Length contains the total length of the sub-object in bytes, including the Type and Length fields. The Length is always 8.

IPv4 address

A 32-bit unicast host address.

Flags

TBD

4.7.1.2. IPv6 Sub-object

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type     |     Length    |             Flags             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                          IPv6 address                         |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: IPv6 sub-object format

Type

0x02 IPv6 address

Length

The Length contains the total length of the sub-object in bytes, including the Type and Length fields. The Length is always 20.

IPv6 address

A 128-bit unicast host address.

Flags

TBD

4.7.1.3. Unnumbered Interface Sub-object

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type     |     Length    |             Flags             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Router ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Interface ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: IPv4 sub-object format

Type

0x03 Unnumbered interface

Length

The Length contains the total length of the sub-object in bytes, including the Type and Length fields. The Length is always 12.

Router ID

The Router ID interpreted as discussed in the Section 2 of RFC 3447 [RFC3477].

Interface ID

The identifier assigned to the link by the LSR specified by the Router ID.

Flags

TBD

5. Data Plane Theory of Operation

After instantiating an LSP for a path using RSVP-TE [RFC3209] as described in Section 4.6 or as described in the second paragraph of Section 4 and in Section 4.6, ingress LSR MAY begin sending RTM packets to the first downstream RTM capable LSR on that path. Each RTM packet has its Scratch Pad field initialized and its TTL set to expire on the next downstream RTM-capable LSR. Each RTM-capable LSR on the explicit path receives an RTM packet and records the time at which it receives that packet at its ingress interface as well as the time at which it transmits that packet from its egress interface; this should be done as close to the physical layer as possible to ensure precise accuracy in time determination. The RTM-capable LSR determines the difference between those two times; for 1-step operation, this difference is determined just prior to or while sending the packet, and the RTM-capable egress interface adds it to the value in the Scratch Pad field of the message in progress. Note, for the purpose of calculating a residence time, a common free running clock synchronizing all the involved interfaces may be sufficient, as, for example, 4.6 ppm accuracy leads to 4.6 nanosecond error for residence time on the order of 1 millisecond.

For 2-step operation, the difference between packet arrival time (at an ingress interface) and subsequent departure time (from an egress interface) is determined at some later time prior to sending a

subsequent follow-up message, so that this value can be used to update the correctionField in the follow-up message.

See Section 7 for further details on the difference between 1-step and 2-step operation.

The last RTM-capable LSR on the LSP MAY then use the value in the Scratch Pad field to perform time correction, if there is no follow-up message. For example, the egress LSR may be a PTP Boundary Clock synchronized to a Master Clock and will use the value in the Scratch Pad field to update PTP's correctionField.

6. Applicable PTP Scenarios

The proposed approach can be directly integrated in a PTP network based on the IEEE 1588 delay reqest-response mechanism. The RTM capable LSR nodes act as end-to-end transparent clocks, and typically boundary clocks, at the edges of the MPLS network, use the value in the Scratch Pad field to update the correctionField of the corresponding PTP event packet prior to performing the usual PTP processing.

7. One-step Clock and Two-step Clock Modes

One-step mode refers to the mode of operation where an egress interface updates the correctionField value of an original event message. Two-step mode refers to the mode of operation where this update is made in a subsequent follow-up message.

Processing of the follow-up message, if present, requires the downstream end-point to wait for the arrival of the follow-up message in order to combine correctionField values from both the original (event) message and the subsequent (follow-up) message. In a similar fashion, each 2-step node needs to wait for the related follow-up message, if there is one, in order to update that follow-up message (as opposed to creating a new one. Hence the first node that uses 2-step mode MUST do two things:

1. Mark the original event message to indicate that a follow-up message will be forthcoming (this is necessary in order to

   Let any subsequent 2-step node know that there is already a follow-up message, and

   Let the end-point know to wait for a follow-up message;

2. Create a follow-up message in which to put the RTM determined as an initial correctionField value.

IEEE 1588v2 [IEEE.1588.2008] defines this behaviour for PTP messages.

Thus, for example, with reference to the PTP protocol, the PTPType field identifies whether the message is a Sync message, Follow_up message, Delay_Req message, or Delay_Resp message. The 10 octet long Port ID field contains the identity of the source port, that is, the specific PTP port of the boundary clock connected to the MPLS network. The Sequence ID is the sequence ID of the PTP message carried in the Value field of the message.

PTP messages also include a bit that indicates whether or not a follow-up message will be coming. This bit, once it is set by a 2-step mode device, MUST stay set accordingly until the original and follow-up messages are combined by an end-point (such as a Boundary Clock).

Thus, an RTM packet, containing residence time information relating to an earlier packet, also contains information identifying that earlier packet.

For compatibility with PTP, RTM (when used for PTP packets) must behave in a similar fashion. To do this, a 2-step RTM capable egress interface will need to examine the S-bit in the Flags field of the PTP sub-TLV (for RTM messages that indicate they are for PTP) and - if it is clear (set to zero), it MUST set it and create a follow-up PTP Type RTM message. If the S bit is already set, then the RTM capable node MUST wait for the RTM message with the PTP type of follow-up and matching originator and sequence number to make the corresponding residence time update to the Scratch Pad field.

In practice an RTM operating according to two-step clock behaves like a two-steps transparent clock.

A 1-step capable RTM node MAY elect to operate in either 1-step mode (by making an update to the Scratch Pad field of the RTM message containing the PTP even message), or in 2-step mode (by making an update to the Scratch Pad of a follow-up message when its presence is indicated), but MUST NOT do both.

Two main subcases can be identified for an RTM node operating as a two-step clock:

A) If any of the previous RTM capable node or the previous PTP clock (e.g. the BC connected to the first LSR), is a two-step clock, the residence time is added to the RTM packet that has been created to include the associated PTP packet (i.e. follow-up message in the downstream direction), if the local RTM-capable LSR is also operating as a two-step clock. This RTM packet carries the related accumulated

residence time and the appropriate values of the Sequence Id and Port Id (the same identifiers carried in the packet processed) and the Two-step Flag set to 1.

Note that the fact that an upstream RTM-capable node operating in the two-step mode has created a follow-up message does not require any subsequent RTM capable LSR to also operate in the 2-step mode, as long as that RTM-capable LSR forwards the follow-up message on the same LSP on which it forwards the corresponding previous message.

A one-step capable RTM node MAY elect to update the RTM follow-up message as if it were operating in two-step mode, however, it MUST NOT update both messages.

A PTP event packet (sync) is carried in the RTM packet in order for an RTM node to identify that residence time measurement must be performed on that specific packet.

To handle the residence time of the Delay request message on the upstream direction, an RTM packet must be created to carry the residence time on the associated downstream Delay Resp message.

The last RTM node of the MPLS network in addition to update the correctionField of the associated PTP packet, must also properly handle the two-step flag of the PTP packets.

B) When the PTP network connected to the MPLS and RTM node, operates in one-step clock mode, the associated RTM packet must be created by the RTM node itself. The associated RTM packet including the PTP event packet needs now to indicate that a follow up message will be coming.

The last RTM node of the LSP, modeif it receives an RTM message with a PTP payload indicating a follow-up message will be forthcoming, must generate a follow-up message and properly set the two-step flag of the PTP packets.

8. IANA Considerations

8.1. New RTM G-ACh

IANA is requested to reserve a new G-ACh as follows:

Value	Description	Reference
TBA1	Residence Time Measurement	This document

Table 1: New Residence Time Measurement

8.2. New RTM TLV Registry

IANA is requested to create sub-registry in Generic Associated Channel (G-ACh) Parameters Registry called "MPLS RTM TLV Registry". All code points in the range 0 through 127 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC5226] . Remaining code points are allocated according to the table below. This document defines the following new values RTM TLV type s:

Value	Description	Reference
0	Reserved	This document
1	No payload	This document
2	PTPv2	This document
3	NTP	This document
4-127	Reserved	IETF Consensus
128 - 191	Reserved	First Come First Served
192 - 255	Reserved	Private Use

Table 2: RTM TLV Type

8.3. New RTM Sub-TLV Registry

IANA is requested to create sub-registry in MPLS RTM TLV Registry, requested in Section 8.2, called "MPLS RTM Sub-TLV Registry". All code points in the range 0 through 127 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC5226] . Remaining code points are allocated according to the table below. This document defines the following new values RTM sub-TLV types:

Value	Description	Reference
0	Reserved	This document
1	PTP 2-step	This document
2-127	Reserved	IETF Consensus
128 - 191	Reserved	First Come First Served
192 - 255	Reserved	Private Use

Table 3: RTM Sub-TLV Type

8.4. RTM Capability sub-TLV

IANA is requested to assign a new type for RTM Capability sub-TLV from future OSPF Extended Link TLV Sub-TLVs registry as follows:

Value	Description	Reference
TBA2	RTM Capability	This document

Table 4: RTM Capability sub-TLV

8.5. IS-IS RTM Application ID

IANA is requested to assign a new Application ID for RTM from the Application Identifiers for TLV 251 registry as follows:

Value	Description	Reference
TBA3	RTM	This document

Table 5: IS-IS RTM Application ID

8.6. RTM_SET Object RSVP Class Number, Class Type and Sub-object Types

IANA is requested to assign a new Class Number for RTM_SET object as follows:

Value	Description	Reference
TBA4	RTM_SET object	This document

Table 6: RTM_SET object Class

IANA is requested to assign a new Class Type for RTM_SET object as follows:

Value	Description	Reference
TBA5	RTM Set	This document

Table 7: RTM_SET object Class Type

IANA requested to create new sub-registry for sub-object types of RTM_SET object as follows:

Value	Description	Reference
0	Reserved	
1	IPv4 address	This document
2	IPv6 address	This document
3	Unnumbered interface	This document
4-127	Reserved	IETF Consensus
128 - 191	Reserved	First Come First Served
192 - 255	Reserved	Private Use

Table 8: RTM_SET object sub-object types

9. Security Considerations

Routers that support Residence Time Measurement are subject to the same security considerations as defined in [RFC5586] .

In addition - particularly as applied to use related to PTP - there is a presumed trust model that depends on the existence of a trusted relationship of at least all PTP-aware nodes on the path traversed by PTP messages. This is necessary as these nodes are expected to correctly modify specific content of the data in PTP messages and proper operation of the protocol depends on this ability.

As a result, the content of the PTP-related data in RTM messages that will be modified by intermediate nodes cannot be authenticated, and the additional information that must be accessible for proper operation of PTP 1-step and 2-step modes MUST be accessible to intermediate nodes (i.e. - MUST NOT be encrypted in a manner that makes this data inaccessible).

While it is possible for a supposed compromised LSR to intercept and modify the G-ACh content, this is an issue that exists for LSRs in general - for any and all data that may be carried over an LSP - and is therefore the basis for an additional presumed trust model associated with existing LSPs and LSRs.

The ability for potentially authenticating and/or encrypting RTM and PTP data that is not needed by intermediate RTM/PTP-capable nodes is for further study.

Security requirements of time protocols are provided in RFC 7384 [RFC7384].

10. Acknowledgements

Authors want to thank Loa Andersson for his thorough review and thoghtful comments.

11. References

11.1. Normative References

[I-D.ietf-ospf-ospfv3-lsa-extend]
Lindem, A., Mirtorabi, S., Roy, A., and F. Baker, "OSPFv3 LSA Extendibility", draft-ietf-ospf-ospfv3-lsa-extend-06 (work in progress), February 2015.

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

[IEEE.1588.2008]
"Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Standard 1588, March 2008.

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

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC3477] Kompella, K. and Y. Rekhter, "Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)", RFC 3477, January 2003.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, February 2006.

[RFC5085] Nadeau, T. and C. Pignataro, "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires", RFC 5085, December 2007.

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

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC6423] Li, H., Martini, L., He, J., and F. Huang, "Using the Generic Associated Channel Label for Pseudowire in the MPLS Transport Profile (MPLS-TP)", RFC 6423, November 2011.

[RFC6823] Ginsberg, L., Previdi, S., and M. Shand, "Advertising Generic Information in IS-IS", RFC 6823, December 2012.

11.2. Informative References

[RFC4202] Kompella, K. and Y. Rekhter, "Routing Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4202, October 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, September 2011.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, October 2014.

Authors' Addresses

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Stefano Ruffini
Ericsson

Email: stefano.ruffini@ericsson.com

Eric Gray
Ericsson

Email: eric.gray@ericsson.com

John Drake
Juniper Networks

Email: jdrake@juniper.net

Stewart Bryant
Cisco Systems

Email: stbryant@cisco.com

Alexander Vainshtein
ECI Telecom

Email: Alexander.Vainshtein@ecitele.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2017

M. Taillon
T. Saad, Ed.
R. Gandhi
Cisco Systems, Inc.
A. Deshmukh
M. Jork
V. Beeram
Juniper Networks
March 07, 2017

RSVP-TE Summary Fast Reroute Extensions for LSP Tunnels
draft-mtaillon-mpls-summary-frr-rsvpte-05

Abstract

This document defines Resource Reservation Protocol (RSVP) Traffic-Engineering (TE) signaling extensions that reduce the amount of RSVP signaling required for Fast Reroute (FRR) procedures and subsequently improve the scalability of the RSVP-TE signaling when undergoing FRR convergence after a link or node failure. Such extensions allow the RSVP message exchange between the Point of Local Repair (PLR) and the Merge Point (MP) to be independent of the number of protected Label Switched Paths (LSPs) traversing between them when facility bypass FRR protection is used. The signaling extensions are fully backwards compatible with nodes that do not support them.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 8, 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

The Fast Reroute (FRR) procedures defined in [RFC4090] describe the mechanisms for the Point of Local Repair (PLR) to reroute traffic and signaling of a protected RSVP-TE LSP onto the bypass tunnel in the event of a TE link or node failure. Such signaling procedures are performed individually for each affected protected LSP. This may eventually lead to control plane scalability and latency issues under limited (memory and CPU processing) resources after a failure that

affects a large number of protected LSPs traversing the same PLR and Merge Point (MP) nodes.

For example, in a large RSVP-TE LSPs scale deployment, a single LSR acting as a PLR node may host tens of thousands of protected RSVP-TE LSPs egressing the same link, and also act as a MP node for similar number of LSPs ingressing the same link. In the event of the failure of the link or neighbor node, the RSVP-TE control plane of the node when acting as PLR becomes busy rerouting protected LSPs signaling over the bypass tunnel(s) in one direction, and when acting as an MP node becomes busy merging RSVP states from signaling received over bypass tunnels for LSP(s) in the reverse direction. Subsequently, the head-end LER(s) that are notified of the local repair at downstream LSR will attempt to (re)converge affected RSVP- TE LSPs onto newly computed paths - possibly traversing the same previously affected LSR(s). As a result, the RSVP-TE control plane at the PLR and MP becomes overwhelmed by the amount of FRR RSVP-TE processing overhead following the link or node failure, and the competing other control plane protocol(s) (e.g. the IGP) that undergo their convergence at the same time.

The extensions defined in this document enable a MP node to become aware of the PLR node’s bypass tunnel assignment group and allow FRR procedures between PLR node and MP node to be signaled and processed on groups of LSPs. Further, the MESSAGE_ID for the rerouted PATH and RESV states are exchanged a priori to the fault such that Summary Refresh procedures defined in [RFC2961] can continue to be used to refresh the rerouted state(s) after FRR has occurred.

2. Conventions Used in This Document

2.1. Key Word Definitions

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 BCP 14, RFC 2119 [RFC2119].

2.2. Terminology

The reader is assumed to be familiar with the terminology in [RFC3209] and [RFC4090].

3. Summary FRR Signaling Procedures

The RSVP ASSOCIATION object is defined in [RFC4872] as a means to associate LSPs with each other. For example, in the context of GMPLS-controlled LSP(s), the object is used to associate recovery

LSPs with the LSP they are protecting. The Extended ASSOCIATION object is introduced in [RFC6780] to expand on the possible usage of the ASSOCIATION object and generalize the definition of the Association ID field.

This document proposes the use of the Extended ASSOCIATION object to carry the Summary FRR information and associate the protected LSP(s) with the bypass tunnel that protects them. To this extent, a new Association Type for the Extended ASSOCIATION object, and a new Association ID are proposed in this draft to describe the Bypass Summary FRR (B-SFRR) association.

The PLR creates and manages the Summary FRR LSP groups (Bypass_Group_Identifiers) and shares them with the MP via signaling. Protected LSPs sharing the same egress link and bypass assignment are grouped together and are assigned the same group. The MP maintains the PLR group assignments learned via signaling, and acknowledges the group assignments via signaling. Once the PLR receives the acknowledgment, FRR signaling can proceed as group based.

The PLR node that supports Summary FRR procedures adds the Extended ASSOCIATION object with Bypass Summary FRR Association Type - referred to thereon in this document as B-SFRR Extended ASSOCIATION object- in the RSVP Path message of the protected LSP to inform the MP of the PLR’s assigned bypass tunnel, Summary FRR Bypass_Group_Identifier, and the MESSAGE_ID object that the PLR will use to refresh the protected LSP PATH state after FRR occurs.

The MP node that supports Summary FRR procedures adds the B-SFRR Extended ASSOCIATION object in a RSVP Resv message of the protected LSP to acknowledge the PLR’s bypass tunnel assignment, and provide the MESSAGE_ID object that the MP node will use to refresh the protected LSP RESV state after FRR occurs.

This document also defines a new RSVP FRR_ACTIVE SUMMARY_FRR_BYPASS object that is sent within the RSVP Path message of a bypass LSP to inform the MP node that one or more groups of protected LSPs that are being protected by the bypass tunnel are being rerouted i.e. signaling is rerouted over the bypass tunnel.

3.1. Signaling Procedures Prior to Failure

Before Summary FRR procedures can be used, a handshake MUST be completed between the PLR and MP. This handshake is performed using B-SFRR Extended ASSOCIATION object that is carried in both the RSVP Path and Resv messages of the protected LSP.

3.1.1. Extended ASSOCIATION Object

The B-SFRR Extended ASSOCIATION object is populated using the rules defined below to associate the Summary FRR enabled protected LSP with the bypass LSP that is protecting it.

The Association Type, Association ID, and Association Source MUST be set as defined in [RFC4872] for the ASSOCIATION Object. More specifically:

Association Source:

The Association Source is set to an address selected by the node that originates the association. For Bypass Summary FRR association it is set to an address of the PLR node.

Association Type:

The Association Type is set to indicate the Bypass Summary FRR association. A new Association Type is defined as follows:

Value	Type
(TBD-1)	Bypass Summary FRR Association (B-SFRR)

Extended Association ID:

The Extended Association ID is populated by the node originating the association -- i.e. the PLR for the Bypass Summary FRR association. The rules to populate the Extended Association ID in this case is described below.

3.1.1.1. IPv4 Extended Association ID

The IPv4 Extended Association ID for Summary FRR bypass assignment has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Bypass_Tunnel_ID      |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Bypass_Source_IPv4_Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Bypass_Destination_IPv4_Address                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Bypass_Group_Identifier                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                MESSAGE_ID                                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: The IPv4 Extended Association ID field

Bypass_Tunnel_ID: 16 bits

The bypass tunnel identifier.

Reserved: 16 bits

Reserved for future use.

Bypass_Source_IPv4_Address: 32 bits

The bypass tunnel source IPV4 address.

Bypass_Destination_IPv4_Address: 32 bits

The bypass tunnel destination IPV4 address.

Bypass_Group_Identifier: 32 bits

The bypass tunnel group identifier.

MESSAGE_ID

A MESSAGE_ID object as defined by [RFC2961].

3.1.1.2. IPv6 Extended Association ID

The IPv6 Extended Association ID field for the Summary FRR information has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Bypass_Tunnel_ID      |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                Bypass_Source_IPv6_Address                     +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                Bypass_Destination_IPv6_Address                +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Bypass_Group_Identifier                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                MESSAGE_ID                                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: The IPv6 Extended Association ID field

Bypass_Tunnel_ID: 16 bits

The bypass tunnel identifier.

Reserved: 16 bits

Reserved for future use.

Bypass_Source_IPv6_Address: 128 bits

The bypass tunnel source IPV6 address.

Bypass_Destination_IPv6_Address: 128 bits

The bypass tunnel destination IPV6 address.

Bypass_Group_Identifier: 32 bits

The bypass tunnel group identifier.

MESSAGE_ID

A MESSAGE_ID object as defined by [RFC2961].

The PLR assigns a bypass tunnel and Bypass_Group_Identifier for each protected LSP. The same Bypass_Group_Identifier is used for the set of protected LSPs that share the same bypass tunnel and traverse the same egress link and are not already rerouted. The PLR also generates a MESSAGE_ID object (flags SHOULD be clear, Epoch and Message_Identifier MUST be set according to [RFC2961]).

The PLR MUST generate a new Message_Identifier each time the contents of the B-SFRR Extended ASSOCIATION object change; for example, when PLR node changes the bypass tunnel assignment.

The PLR node notifies the MP node of the bypass tunnel assignment via adding a B-SFRR Extended ASSOCIATION object in the RSVP Path message for the protected LSP using procedures described in Section 3.2.

The MP node acknowledges the PLR node assignment by signaling the B-SFRR Extended Association object within the RSVP Resv message of the protected LSP. With exception of the MESSAGE_ID objects, all other fields of the received B-SFRR Extended ASSOCIATION object in the RSVP Path message are copied into the B-SFRR Extended ASSOCIATION object to be added in the Resv message. The MESSAGE_ID object is set according to [RFC2961] with the Flags being clear. A new Message_Identifier MUST be used to acknowledge an updated PLR assignment.

The PLR considers the protected LSP as Summary FRR capable only if the B-SFRR Extended ASSOCIATION objects sent in the RSVP Path message and the one received in the RSVP Resv message (with exception of the MESSAGE_ID) match. If it does not match, or if B-SFRR Extended Association object is absent in a subsequent refresh, the PLR node MUST consider the protected LSP as not Summary FRR capable.

3.1.2. PLR Summary FRR Signaling Procedure

The B-SFRR Extended ASSOCIATION object is added by each PLR in the RSVP Path message of the protected LSP to record the bypass tunnel assignment. This object is updated every time the PLR updates the bypass tunnel assignment (which triggers an RSVP Path change message).

Upon receiving an RSVP Resv message with B-SFRR Extended ASSOCIATION object, the PLR node checks if the expected subobjects in the B-SFRR Extended ASSOCIATION ID are present. If present, the PLR determines if the MP has acknowledged the current PLR assignment.

To be a valid acknowledgement, the received B-SFRR Extended ASSOCIATION object contents within the RSVP Resv message of the protected LSP MUST match the latest B-SFRR Extended ASSOCIATION object contents that the PLR node had sent within the RSVP Path message (with exception of the MESSAGE_ID).

Note, when forwarding an RSVP Resv message upstream, the PLR node SHOULD remove any/all B-SFRR Extended ASSOCIATION objects whose Association Source matches the PLR node address.

3.1.3. MP Summary FRR Signaling Procedure

Upon receiving an RSVP Path message with an B-SFRR Extended ASSOCIATION object, the MP node processes all (there may be multiple PLRs for a single MP) B-SFRR Extended ASSOCIATION objects that have the MP node address as Bypass Destination address in the Association ID.

The MP node first ensures the existence of the bypass tunnel and that the Bypass_Group_Identifier is not already FRR active. That is, an LSP cannot join a group that is already FRR rerouted.

The MP node builds a mirrored Summary FRR Group database per PLR, which is determined using the Bypass_Source_Address field. The MESSAGE_ID is extracted and recorded for the protected LSP PATH state. The MP node signals a B-SFRR Extended Association object within the RSVP Resv message of the protected LSP. With exception of the MESSAGE_ID objects, all other fields of the received B-SFRR

Extended ASSOCIATION object in the RSVP Path message are copied into the B-SFRR Extended ASSOCIATION object to be added in the Resv message. The MESSAGE_ID object is set according to [RFC2961] with the Flags being clear.

Note, an MP may receive more than one RSVP Path message with the B-SFRR Extended ASSOCIATION object from different upstream PLR node(s). In this case, the MP node is expected to save all the received MESSAGE_IDs from the different upstream PLR node(s). After a failure, the MP node determines and activates the associated Summary Refresh ID to use once it receives and processes the RSVP Path message with FRR_ACTIVE SUMMARY_FRR_BYPASS object over the bypass LSP from the PLR.

When forwarding an RSVP Path message downstream, the MP SHOULD remove any/all B-SFRR Extended ASSOCIATION object(s) whose Association ID contains Bypass_Destination_Address matching the MP node address.

3.2. Signaling Procedures Post Failure

Upon detection of the fault (egress link or node failure) the PLR first performs the object modification procedures described by Section 6.4.3 of [RFC4090] for all affected protected LSPs. For Summary FRR LSPs assigned to the same bypass tunnel a common RSVP_HOP and SENDER_TEMPLATE MUST be used.

The PLR MUST signal non-Summary FRR enabled LSPs over the bypass tunnel before signaling the Summary FRR enabled LSPs. This is needed to allow for the case when the PLR node has recently changed a bypass assignment and the MP has not processed the change yet.

A new object FRR_ACTIVE SUMMARY_FRR_BYPASS is defined in Section 3.2.1 and sent within the RSVP Path message of the bypass LSP to reroute RSVP state of Summary FRR enabled LSPs.

3.2.1. SUMMARY_FRR_BYPASS Object

The SUMMARY_FRR_BYPASS Object with Type FRR_ACTIVE is carried in the Path message of a bypass LSP. This object is added by the PLR node to indicate to the MP node (bypass tunnel destination) that one or more groups of protected LSPs that are being protected by the specified bypass tunnel are being rerouted over the bypass tunnel.

The FRR_ACTIVE SUMMARY_FRR_BYPASS object is assigned the C-Type (TBD-3). The FRR_ACTIVE SUMMARY_FRR_BYPASS object has the below format.

SUMMARY_FRR_BYPASS Class-Num = (TBD-2) (of the form 11bbbbbb) Class-Name = SUMMARY_FRR_BYPASS Class, FRR_ACTIVE C-Type = (TBD-3)

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             | Class-N(TBD-2)| C-Type (TBD-3)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Reserved           |       Num-BGIDs               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Bypass_Group_Identifier                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               :                               |
//                              :                              //
|                               :                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Bypass_Group_Identifier                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           RSVP_HOP_Object                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           TIME_VALUES                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Summary FRR Bypass Object

Reserved: 16 bits

Reserved for future use.

Num-BGIDs: 16 bits

Number of Bypass_Group_Identifier fields.

Bypass_Group_Identifier: 32 bits

The Bypass_Group_Identifier that is previously advertised by the PLR using the Extended Association object. One or more Bypass_Group_Identifiers may be included.

RSVP_HOP_Object: Class 3, as defined by [RFC2205]

Replacement RSVP HOP object to be applied to all LSPs associated with each of the following Bypass_Group_Identifiers. This corresponds to C-Type = 1 for IPv4 RSVP HOP, or C-Type = 2 for IPv6 RSVP HOP depending on the IP address family carried within the object.

TIME_VALUES object: Class 5, as defined by [RFC2205]

Replacement TIME_VALUES object to be applied to all LSPs associated with each of the following Bypass_Group_Identifiers after receiving the FRR_ACTIVE SUMMARY_FRR_BYPASS object.

3.2.2. PLR Summary FRR Signaling Procedure

After a failure event, when using the Summary FRR path signaling procedures, an individual RSVP Path message for each Summary FRR LSP is not signaled. Instead, to reroute Summary FRR LSPs via the bypass tunnel, the PLR adds the FRR_ACTIVE SUMMARY_FRR_BYPASS object in the RSVP Path message of the RSVP session of the bypass tunnel.

The RSVP_HOP_Object field of the FRR_ACTIVE SUMMARY_FRR_BYPASS object is set to the common RSVP_HOP that was used by the PLR in Section 3.2 of this document.

The previously received MESSAGE_ID from the MP is activated. As a result, the MP may refresh the protected rerouted RESV state using Summary Refresh procedures.

For each affected Summary FRR group, its Bypass_Group_Identifier is added to the FRR_ACTIVE SUMMARY_FRR_BYPASS object.

3.2.3. MP Summary FRR Signaling Procedure

Upon receiving an RSVP Path message with a FRR_ACTIVE SUMMARY_FRR_BYPASS object, the MP performs normal merge point processing for each protected LSP associated with each Bypass_Group_Identifier, as if it received individual RSVP Path messages for the LSP.

For each Summary FRR LSP being merged, the MP first modifies the Path state as follows:

1. The RSVP_HOP object is copied from the FRR_ACTIVE SUMMARY_FRR_BYPASS RSVP_HOP_Object field.

2. The TIME_VALUES object is copied from the FRR_ACTIVE SUMMARY_FRR_BYPASS TIMES_VALUE field. The TIME_VALUES object contains the refresh time of the PLR to generate refreshes and that would have exchanged in a Path message sent to the MP after the failure when no SFRR procedures are in effect.

3. The SENDER_TEMPLATE object SrcAddress field is copied from the bypass tunnel SENDER_TEMPLATE object. For the case where PLR is also the head-end, and SENDER_TEMPLATE SrcAddress of the protected LSP and bypass tunnel are the same, the MP MUST use the modified HOP Address field instead.

4. The ERO object is modified as per Section 6.4.4. of [RFC4090]. Once the above modifications are completed, the MP then performs the merge processing as per [RFC4090].

5. The previously received MESSAGE_ID from the PLR is activated, meaning that the PLR may now refresh the protected rerouted PATH state using Summary Refresh procedures.

A failure during merge processing of any individual rerouted LSP MUST result in an RSVP Path Error message.

An individual RSVP Resv message for each successfully merged Summary FRR LSP is not signaled. The MP node SHOULD immediately use Summary Refresh procedures to refresh the protected LSP RESV state.

3.3. Refreshing Summary FRR Active LSPs

Refreshing of Summary FRR active LSPs is performed using Summary Refresh as defined by [RFC2961].

4. Compatibility

The (Extended) ASSOCIATION object is defined in [RFC4872] with a class number in the form 11bbbbbb, which ensures compatibility with non-supporting node(s). Such nodes will ignore the object and forward it without modification.

The new FRR_ACTIVE SUMMARY_FRR_BYPASS object is to be defined with a class number in the form 11bbbbbb, which ensures compatibility with non-supporting nodes. Per [RFC2205], the nodes not supporting this extension will ignore the object but forward it, unexamined and unmodified, in all messages.

5. Security Considerations

This document updates an existing RSVP object, and introduces a new RSVP object. Thus, in the event of the interception of a signaling message, a slightly more information could be deduced about the state of the network than was previously the case. Existing mechanisms for maintaining the integrity and authenticity of RSVP protocol messages [RFC2747] can be applied.

6. IANA Considerations

IANA maintains the "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Parameters" registry (see http://www.iana.org/assignments/gmpls-sig-parameters). The "Association Type" subregistry is included in this registry.

This registry has been updated by new Association Type for Extended ASSOCIATION Object defined in this document as follows:

Value	Name	Reference
TBD-1	Bypass Summary FRR Association (B-SFRR)	Section 2.1.1

IANA also maintains and assigns the values for the RSVP-TE protocol parameters "Resource Reservation Protocol (RSVP) Parameters" (see http://www.iana.org/assignments/rsvp-parameters).

From this registry, a new RSVP Class (TBD-2) and of the form 11bbbbbb and a new C-Type (TBD-3) are requested for the new FRR_ACTIVE SUMMARY_FRR_BYPASS object defined in this document.

Class-Number = (TBD-2), Class-Name = SUMMARY_FRR_BYPASS

C-Type = (TBD-3) Name = FRR_ACTIVE

7. Acknowledgments

The authors would like to thank Loa Andersson, Lou Berger, Eric Osborne, Gregory Mirsky, and Mach Chen for reviewing and providing valuable comments to this document.

8. Contributors

Nicholas Tan
Arista Networks

Email: ntan@arista.com

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, <http://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, <http://www.rfc-editor.org/info/rfc2205>.

[RFC2747] Baker, F., Lindell, B., and M. Talwar, "RSVP Cryptographic Authentication", RFC 2747, DOI 10.17487/RFC2747, January 2000, <http://www.rfc-editor.org/info/rfc2747>.

[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, <http://www.rfc-editor.org/info/rfc2961>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc4090>.

[RFC4872] Lang, J., Ed., Rekhter, Y., Ed., and D. Papadimitriou, Ed., "RSVP-TE Extensions in Support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery", RFC 4872, DOI 10.17487/RFC4872, May 2007, <http://www.rfc-editor.org/info/rfc4872>.

[RFC6780] Berger, L., Le Faucheur, F., and A. Narayanan, "RSVP ASSOCIATION Object Extensions", RFC 6780, DOI 10.17487/RFC6780, October 2012, <http://www.rfc-editor.org/info/rfc6780>.

Authors' Addresses

Mike Taillon
Cisco Systems, Inc.

Email: mtaillon@cisco.com

Tarek Saad (editor)
Cisco Systems, Inc.

Email: tsaad@cisco.com

Rakesh Gandhi
Cisco Systems, Inc.

Email: rgandhi@cisco.com

Abhishek Deshmukh
Juniper Networks

Email: adeshmukh@juniper.net

Markus Jork
Juniper Networks

Email: mjork@juniper.net

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

q

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

J. George
Google
L. Fang
Microsoft
E. Osborne
Level 3
R. Shakir
Jive Communications

October 19, 2015

MPLS / TE Model for Service Provider Networks
draft-openconfig-mpls-consolidated-model-02

Abstract

This document defines a framework for a YANG data model for configuring and managing label switched paths, including the signaling protocols, traffic engineering, and operational aspects based on carrier and content provider operational requirements.

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 April 21, 2016.

Copyright Notice

Copyright (c) 2015 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

This document describes a YANG [RFC6020] data model for MPLS and traffic engineering, covering label switched path (LSP) configuration, as well as signaling protocol configuration. The model is intended to be vendor-neutral, in order to allow operators to manage MPLS in heterogeneous environments with physical or virtual devices (routers, switches, servers, etc.) supplied by multiple vendors. The model is also intended to be readily mapped to existing implementations, to facilitate support from as large a set of routing hardware and software vendors as possible.

1.1. Goals and approach

The focus area of the model in this revision, is to set forth a framework for MPLS, with hooks into which information specific to various signaling-protocols can be added. The framework is built around functionality from a network operator perspective rather than

a signaling protocol-centric approach. For example, a traffic-engineered LSP will have configuration relating to its path computation method, regardless of whether it is signaled with RSVP-TE or with segment routing. Thus, rather than creating separate per-signaling protocol models and trying to stitch them under a common umbrella, this framework focuses on functionality, and adds signaling protocol-specific information under it where applicable.

This model does not aim to be feature complete (i.e., cover all possible aspects or features of MPLS). Rather its development is driven by examination of actual production configurations in use across a number of operator network deployments.

Configuration items that are deemed to be widely available in existing major implementations are included in the model. Those configuration items that are only available from a single implementation are omitted from the model with the expectation they will be available in companion modules that augment the current model. This allows clarity in identifying data that is part of the vendor-neutral model.

An important aspect of the model is the representation of operational state data. This draft takes the approach described in [I-D.openconfig-netmod-opstate] and models configuration and operational state together. Thus, rather than building a separate tree of operational state, the operational state and configuration data are located in parallel containers at the leaves of the data model. This approach allows easy reuse of groupings across models, as well as making it easier to correlate configuration and state.

The consolidated MPLS model encompasses the signaling protocols, label-switched paths (configuration and operational state), and generic TE attributes. The model is designed from an operational and functional perspective, rather than focusing on protocol-centric configuration. This allows protocol-independent functions to be logically separated from protocol-specific details.

One question that arises in this approach is how the consolidated model is integrated with routing instances (e.g., VRFs). This model should be considered as part of a higher level network device model which includes definitions for other routing protocols and system services. For example, in [I-D.openconfig-netmod-model-structure], VRFs and other logical instances are defined with MPLS/TE components within VRFs as appropriate. In particular, some parts of the MPLS model would be instantiated within a VRF, while other parts would have common definitions across VRFs.

Where possible, naming in the model follows conventions used in available standards documents, and otherwise tries to be self-explanatory with sufficient descriptions of the intended behavior. Similarly, configuration data value constraints and default values, where used, are based on recommendations in current standards documentation. Since implementations vary widely in this respect, this version of the model specifies only a limited set of defaults and ranges with the expectation of being more prescriptive in future versions based on actual operator use.

Note that this version of the model is a work-in-progress in several respects. Although we present a complete framework for MPLS and traffic engineering from an operational perspective, some signaling protocol configuration will be completed in future revisions. The current revision has focus on traffic engineered LSPs signaled with RSVP.

2. Model overview

The overall MPLS model is defined across several YANG modules and submodules but at a high level is organized into 4 main sections:

- o global -- configuration affecting MPLS behavior which exists independently of the underlying signaling protocol or label switched path configuration.

- o te-global-attributes -- configuration affecting MPLS-TE behavior which exists independently of the underlying signaling protocol or label switched path configuration.

- o signaling protocols -- configuration specific to signaling protocols used to setup and manage label switched paths.

- o label switched paths -- configuration specific to instantiating and managing individual label switched paths.

The top level of the model is shown in the tree view below:

```
+--rw mpls!
      +--rw global
      |     ...
      +--rw te-global-attributes
      |     ...
      +--rw signaling-protocols
      |     ...
      +--rw lsps
            ...
```

2.1. MPLS global

The global section of the framework provides configuration data for MPLS items which exist independently of an individual label switched path or signaling protocol and are applicable to the MPLS protocol itself. Items such as the depth of the label stack supported, or specific label ranges may be included here.

2.2. TE global attributes

The TE global attributes section of the framework provides configuration control for MPLS-TE items which exist independently of an individual label switched path or signaling protocol. These standalone items are applicable to the entire logical routing device, and establish fundamental configuration such as the threshold for interface bandwidth change that triggers update events into the IGP traffic engineering database (TED). Timers are also specified which determine the length of time an LSP must be present before being considered viable for forwarding use (te-lsp-install-delay), and the length of time between LSP teardown and removal of the LSP from the network element’s forwarding information base (te-lsp-cleanup-delay). Also specified are the name to value mappings of MPLS administrative groups (mpls-admin-groups) and shared risk link groups (mpls-te-srlg).

```
    +--rw te-global-attributes
         |  +--rw mpls-te-srlg
         |  |  +--rw srlg* [srlg-name]
         |  |     +--rw srlg-name       leafref
         |  |     +--rw config
         |  |     |  +--rw srlg-name?    string
         |  |     |  +--rw srlg-value?   uint32
         |  |     |  +--rw srlg-cost?    uint32
         |  |     +--ro state
         ...
         |  |     +--rw members-list* [from-address]
         |  |        +--rw from-address    leafref
         |  |        +--rw config
         |  |        |  +--rw from-address?   inet:ip-address
         |  |        |  +--rw to-address?     inet:ip-address
         ...
         |  +--rw igp-flooding-bandwidth
         |  |  +--rw config
         |  |  |  +--rw threshold-type?            enumeration
         |  |  |  +--rw delta-percentage?          oc-types:percentage
         |  |  |  +--rw threshold-specification?   enumeration
         |  |  |  +--rw up-thresholds*             oc-types:percentage
         |  |  |  +--rw down-thresholds*           oc-types:percentage
         |  |  |  +--rw up-down-thresholds*        oc-types:percentage
         |  |  +--ro state
         ...
         |  +--rw mpls-admin-groups
         |  |  +--rw admin-group* [admin-group-name]
         |  |     +--rw admin-group-name    leafref
         |  |     +--rw config
         |  |     |  +--rw admin-group-name?    string
         |  |     |  +--rw admin-group-value?   uint32
         |  |     +--rw state
         ...
         |  +--rw te-lsp-timers
         |     +--rw config
         |     |  +--rw te-lsp-install-delay?      uint16
         |     |  +--rw te-lsp-cleanup-delay?      uint16
         |     |  +--rw te-lsp-reoptimize-timer?   uint16
         |     +--ro state
         ...
```

2.3. TE interface attributes overview

The TE interface attributes section of the framework provides configuration and state related to traffic engineering such as te-metric or shared risk link group configuration.

```
        +--rw te-intf-attributes
        |  +--rw interface* [interface-name]
        |     +--rw interface-name    leafref
        |     +--rw config
        |     |  +--rw interface-name?          ocif:interface-ref
        |     |  +--rw te-metric?               uint32
        |     |  +--rw srlg* [srlg-name]
                     ...
        |     |  +--rw admin-group* [admin-group-name]
                     ...
        |     |  +--rw igp-flooding-bandwidth
                    ...
        |     +--ro state
        ...
```

2.4. Signaling protocol overview

The signaling protocol section of the framework provides configuration elements for configuring three major methods of signaling label switched paths: RSVP-TE, segment routing, and label distribution protocol (LDP). BGP-LU will be included in a future version of this draft by definitions in the BGP model ([I-D.ietf-idr-bgp-model]) and corresponding augmentations to the MPLS model.

```
+--rw signaling-protocols
      |  +--rw rsvp-te
      |     ...
      |  +--rw segment-routing
      |     ...
      |  +--rw ldp
      |     ...
```

Configuration of RSVP-TE is centered around interfaces on the device which participate in the protocol. A key focus is to expose common RSVP-TE configuration parameters which are used to enhance scale and reliability. Items which are applicable globally in the RSVP-TE protocol such as graceful restart, soft preemption and various statistics are grouped into a global section under the protocol. RSVP neighbor and session state are also available in the RSVP section.

```
         +--rw rsvp-te
         |  |  +--rw rsvp-sessions
         |  |  |  +--rw config
         |  |  |  +--ro state
         |  |  |     +--ro rsvp-session* [source-port destination-port
                            source-address destination-address]
                        ...
         |  |  +--rw rsvp-neighbors
         |  |  |  +--rw config
         |  |  |  +--ro state
         |  |  |     +--ro rsvp-neighbor* [neighbor-address]
                        ...
         |  |  +--rw global
         |  |  |  +--rw graceful-restart
                        ...
         |  |  |  +--rw soft-preemption
                        ...
         |  |  |  +--ro statistics
         |  |  |     +--ro counters
                        ....
         |  |  +--rw interface-attributes
         |  |     +--rw interface* [interface-name]
         |  |        +--rw interface-name    leafref
                        ...
         |  |        +--rw rsvp-hellos
                        ...
         |  |        +--rw authentication
                        ...
         |  |        +--rw subscription
                        ...
         |  |        +--rw protection
                        ...
         ...
```

Containers for specifying signaling via segment routing and LDP are also present. Specific subelements will be added for those protocols, as well as for BGP labeled unicast, in the next revision.

2.5. LSP overview

This part of the framework contains LSP information. At the high level, LSPs are split into three categories: traffic-engineering-capable (constrained-path), non-traffic-engineered determined by the IGP (unconstrained-path), and hop-by-hop configured (static).

```
+--rw mpls!
      +--rw lsps
         +--rw constrained-path
         |     ...
         +--rw unconstrained-path
         |     ...
         +--rw static-lsps
               ...
```

The first two categories, constrained-path and unconstrained-path are the ones for which multiple signaling protocols exist, and are organized in protocol-specific and protocol-independent sections. For example, traffic-engineered (constrained path) LSPs may be set up using RSVP-TE or segment routing, and unconstrained LSPs that follow the IGP path may be signaled with LDP or with segment routing. IGP-determined LSPs may also be signaled by RSVP but this usage is not considered in the current version of the model.

A portion of the data model for constrained path traffic-engineered LSPs signaled with RSVP is shown below. It contains configuration for named explicit paths and for tunnels. Tunnel configuration differs for p2p and p2mp LSPs. In either case, some part of the model is signaling-protocol independent. For example for a p2p LSP, attributes such as the path computation method, the constraints for the the path, the bandwidth allocated to it, and even the frequency of reoptimization are signaling-protocol independent, while other data, such as the setup and hold priorities are protocol-specific and are specified in the protocol specific part of the model.

```
+--rw mpls!
      +--rw lsps
         +--rw constrained-path
       |  +--rw explicit-path* [name]
             ...
         |  +--rw tunnel* [name type]
         |     +--rw name      leafref
         |     +--rw type      leafref
         |     +--rw config
         |     |  +--rw name?                        string
         |     |  +--rw type?                        identityref
         |     |  +--rw local-id?                    union
         |     |  +--rw description?                 string
         |     |  +--rw admin-status?                identityref
         |     |  +--rw preference?                  uint8
         |     |  +--rw metric?                      te-metric-type
         |     |  +--rw (bandwidth)?
                     ...
         |     |  +--rw protection-style-requested?   identityref
         |     |  +--rw te-lsp-reoptimize-timer?      uint16
         |     |  +--rw (signaling-specific-tunnel-attributes)?
         |     |  |  +--:(RSVP)
         |     |  |     +--rw source?                      inet:ip-address
         |     |  |     +--rw soft-preemption?             boolean
         |     |  +--rw (tunnel-type)?
         |     |     +--:(p2p)
         |     |     |  +--rw destination?                 inet:ip-address
         |     |     |  +--rw primary-paths* [name]
         |     |     |  |  +--rw name                      string
         |     |     |  |  +--rw preference?               uint8
         |     |     |  |  +--rw path-computation-method
                              ...
         |     |     |  |  +--rw admin-groups
                              ...
         |     |     |  |  +--rw no-cspf?                  empty
         |     |     |  |  +--rw (sigaling-specific-path-attributes)?
         |     |     |  |     +--:(RSVP)
         |     |     |  |     |  +--rw setup-priority?            uint8
         |     |     |  |     |  +--rw hold-priority?             uint8
         |     |     |  |     |  +--rw retry-timer?               uint16
         |     |     |  |     +--:(SR)
         |     |     |  |        +--rw sid-selection-mode?        enumeration
         |     |     |  |        +--rw sid-protection-required?   boolean
         |     |     |  +--rw secondary-paths* [name]
                        ...
         |     +--ro state
               ...
```

Similarly, the partial model for non-traffic-engineered, or IGP-based, LSPs is shown below:

```
+--rw mpls!
      +--rw lsps
         +--rw unconstrained-path
            +--rw path-setup-protocol
               +--rw ldp!
               |     ...
               +--rw segment-routing!
                     ...
```

3. Example use cases

3.1. Traffic engineered p2p LSP signaled with RSVP

A possible scenario may be the establishment of a mesh of traffic-engineered LSPs where RSVP signaling is desired, and the LSPs use a local constrained path calculation to determine their path. These LSPs would fall into the category of a constrained-path LSP, and the tunnel type is p2p. Attributes such as metric, bandwidth or the style of protection desired are also defined at this (protocol-independent) level in the model. The path is defined to be locally-computed under the path-computation-method container, specifying the use of CSPF (use-cspf). Additional attributes for the path, such as its RSVP priorities are specified at the path level under the protocol-specific stanza.

```
+--rw mpls!
      +--rw lsps
         +--rw constrained-path
            ...
         |  +--rw tunnel* [name type]
         |     +--rw name      leafref
         |     +--rw type      leafref
         |     +--rw config
         |     |  +--rw name?                          string
         |     |  +--rw type?                          identityref
         |     |  +--rw metric?                        te-metric-type
         |     |  +--rw (bandwidth)?
                     ...
         |     |  +--rw protection-style-requested?   identityref
         |     |  +--rw te-lsp-reoptimize-timer?      uint16
                     ...
         |     |  +--rw (tunnel-type)?
         |     |     +--:(p2p)
         |     |     |  +--rw destination?                  inet:ip-address
         |     |     |  +--rw primary-paths* [name]
         |     |     |  |  +--rw name                       string
         |     |     |  |  +--rw preference?                uint8
         |     |     |  |  +--rw path-computation-method
                              ...
         |     |     |  |  +--rw admin-groups
                              ...
         |     |     |  |  +--rw no-cspf?                   empty
         |     |     |  |  +--rw (sigaling-specific-path-attributes)?
         |     |     |  |     +--:(RSVP)
         |     |     |  |     |  +--rw setup-priority?            uint8
         |     |     |  |     |  +--rw hold-priority?             uint8
         |     |     |  |     |  +--rw retry-timer?               uint16
         |     |     |  |     +--:(SR)
         |     |     |  |        +--rw sid-selection-mode?        enumeration
         |     |     |  |        +--rw sid-protection-required?   boolean
         |     |     |  +--rw secondary-paths* [name]
                              ...
         |     +--ro state
```

3.2. Traffic engineered LSP signaled with SR

A possible scenario may be the establishment of disjoint paths in a network where there is no requirement for per-LSP state to be held on midpoint nodes within the network, or RSVP-TE is unsuitable (as described in [I-D.ietf-spring-segment-routing-mpls] and [I-D.shakir-rtgwg-sr-performance-engineered-lsps]). Such LSPs fall in the constrained-path category. Similar to any other traffic engineered LSPs, the path computation method must be specified. Path

attributes, such as the as lsp- placement-constraints (expressed as administrative groups) or metric must be defined. Finally, the path must be specified in a signaling- protocol specific manner appropriate for SR. The same configuration elements from the tree above apply in this case, except that path setup is done by the head-end by building a label stack, rather than signaled.

3.3. IGP-congruent LDP-signaled LSP

A possible scenario may be the establishment of a full mesh of LSPs. When traffic engineering is not an objective, no constraints are placed on the end-to-end path, and the best- effort path can be setup using LDP signaling simply for label distribution. The LSPs follow IGP-computed paths, and fall in the unconstrained-path category in the model. Protocol-specific configuration pertaining to the signaling protocol used, such as the FEC definition and metrics assigned are in the path- setup-protocol portion of the model.

The relevant part of the model for this case is shown below:

```
  +--rw mpls!
      +--rw lsps
         +--rw unconstrained-path
            +--rw path-setup-protocol
               +--rw ldp!
                  +--rw tunnel
                     +--rw tunnel-type?   mplst:tunnel-type
                     +--rw ldp-type?      enumeration
                     +--rw p2p-lsp
                     |  +--rw fec-address*   inet:ip-prefix
                     +--rw p2mp-lsp
                     +--rw mp2mp-lsp
```

A common operational issue encountered when using LDP is traffic blackholing under the following scenario: when an IGP failure occurs, LDP is not aware of it as these are two protocols running independently, resulting in traffic blackholing at the IGP failure point even though LDP is up and running. LDP-IGP synchronization [RFC5443] can be used to cost out the IGP failing point/segment to avoid the blackholing issue. The LDP-IGP synchronization function will be incorporated in a future version of this document.

Note that targeted LDP sessions are not discussed in this use case, and will be incorporated as a separate use case in a future version of this document.

4. Security Considerations

MPLS configuration has a significant impact on network operations, and as such any related protocol or model carries potential security risks.

YANG data models are generally designed to be used with the NETCONF protocol over an SSH transport. This provides an authenticated and secure channel over which to transfer BGP configuration and operational data. Note that use of alternate transport or data encoding (e.g., JSON over HTTPS) would require similar mechanisms for authenticating and securing access to configuration data.

Most of the data elements in the configuration model could be considered sensitive from a security standpoint. Unauthorized access or invalid data could cause major disruption.

5. IANA Considerations

This YANG data model and the component modules currently use a temporary ad-hoc namespace. If and when it is placed on redirected for the standards track, an appropriate namespace URI will be registered in the IETF XML Registry" [RFC3688]. The MPLS YANG modules will be registered in the "YANG Module Names" registry [RFC6020].

6. YANG modules

The modules and submodules comprising the MPLS configuration and operational model are currently organized as depcited below.

```
                              +-------+
             +---------------->| MPLS  |<--------------+
             |                 +-------+               |
             |                     ^                   |
             |                     |                   |
        +----+-----+      +--------+-------+     +-----+-----+
        | TE LSPs  |      | IGP-based LSPs |     |static LSPs|
        |          |      |                |     |           |
        +----------+      +----------------+     +-----------+
            ^  ^                    ^  ^
            |  +----------------+   |  +--------+
            |                   |   |           |
            |   +------+      +-+---+-+      +--+--+
            +---+ RSVP |      |SEGMENT|      | LDP |
                +------+      |ROUTING|      +-----+
                              +-------+
```

The base MPLS module includes submodules describing the three different types of support LSPs, i.e., traffic-engineered (constrained-path), IGP congruent (unconstrained-path), and static. The signaling protocol specific parts of the model are described in separate modules for RSVP, segment routing, and LDP. As mentioned earlier, support for BGP labeled unicast is also planned in a future revision.

A module defining various reusable MPLS types is included, and these modules also make use of the standard Internet types, such as IP addresses, as defined in RFC 6991 [RFC6991].

6.1. MPLS base modules

```
                  <CODE BEGINS> file openconfig-mpls.yang
module openconfig-mpls {

  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/mpls";

  prefix "mpls";


  // import some basic types
  import openconfig-mpls-rsvp { prefix rsvp; }
  import openconfig-mpls-sr { prefix sr; }
  import openconfig-mpls-ldp { prefix ldp; }
  import openconfig-types { prefix oc-types; }
  import openconfig-interfaces { prefix ocif; }

  // include submodules
  include openconfig-mpls-te;
  include openconfig-mpls-igp;
  include openconfig-mpls-static;


  // meta
  organization "OpenConfig working group";

  contact
    "OpenConfig working group
    netopenconfig@googlegroups.com";

  description
    "This module provides data definitions for configuration of
```

```
       Multiprotocol Label Switching (MPLS) and associated protocols for
       signaling and traffic engineering.

       RFC 3031: Multiprotocol Label Switching Architecture

       The MPLS / TE data model consists of several modules and
       submodules as shown below.  The top-level MPLS module describes
       the overall framework.  Three types of LSPs are supported:

       i) traffic-engineered (or constrained-path)

       ii) IGP-congruent (LSPs that follow the IGP path)

       iii) static LSPs which are not signaled

       The structure of each of these LSP configurations is defined in
       corresponding submodules.  Companion modules define the relevant
       configuration and operational data specific to key signaling
       protocols used in operational practice.


                                +-------+
                +---------------->| MPLS  |<--------------+
                |                 +-------+               |
                |                     ^                   |
                |                     |                   |
           +----+-----+      +--------+-------+     +-----+-----+
           | TE LSPs  |      | IGP-based LSPs |     |static LSPs|
           |          |      |                |     |           |
           +----------+      +----------------+     +-----------+
               ^  ^                     ^  ^
               |  +----------------+   |  +--------+
               |                   |   |           |
               |   +------+      +-+---+-+      +--+--+
               +---+ RSVP |      |SEGMENT|      | LDP |
                   +------+      |ROUTING|      +-----+
                                 +-------+
       ";

     revision "2015-10-14" {
       description
         "Work in progress";
       reference "TBD";
     }

     // extension statements

     // feature statements
```

```
     // identity statements


     // grouping statements

     grouping mpls-admin-group_config {
       description
         "configuration data for MPLS link admin groups";

         leaf admin-group-name {
           type string;
           description "name for mpls admin-group";
         }

         leaf admin-group-value {
           type uint32;
           description "value for mpls admin-group";
         }

     }

     grouping mpls-admin-groups-top {

       description "top-level mpls admin-groups config
         and state containers";

       container mpls-admin-groups {
         description
           "Top-level container for admin-groups configuration
           and state";

         list admin-group {
           key admin-group-name;
           description "configuration of value to name mapping
             for mpls affinities/admin-groups";

           leaf admin-group-name {
             type leafref {
               path "../mpls:config/mpls:admin-group-name";
             }
             description
               "name for mpls admin-group";
           }
           container config {
             description "Configurable items for admin-groups";
             uses mpls-admin-group_config;
           }
```

```
          container state {
            description "Operational state for admin-groups";
            uses mpls-admin-group_config;
          }
        }
      }
    }

    grouping mpls-te-igp-flooding-bandwidth_config {
      description
        "Configurable items for igp flooding bandwidth
        threshold configuration.";
      leaf threshold-type {
        type enumeration {
          enum DELTA {
            description "DELTA indicates that the local
              system should flood IGP updates when a
              change in reserved bandwidth >= the specified
              delta occurs on the interface.";
          }
          enum THRESHOLD-CROSSED {
            description "THRESHOLD-CROSSED indicates that
              the local system should trigger an update (and
              hence flood) the reserved bandwidth when the
              reserved bandwidth changes such that it crosses,
              or becomes equal to one of the threshold values.";
          }
        }
        description
          "The type of threshold that should be used to specify the
          values at which bandwidth is flooded. DELTA indicates that
          the local system should flood IGP updates when a change in
          reserved bandwidth >= the specified delta occurs on the
          interface. Where THRESHOLD-CROSSED is specified, the local
          system should trigger an update (and hence flood) the
          reserved bandwidth when the reserved bandwidth changes such
          that it crosses, or becomes equal to one of the threshold
          values";
      }

      leaf delta-percentage {
        when "../threshold-type = 'DELTA'" {
          description
            "The percentage delta can only be specified when the
            threshold type is specified to be a percentage delta of
            the reserved bandwidth";
        }
        type oc-types:percentage;
```

```
          description
            "The percentage of the maximum-reservable-bandwidth
            considered as the delta that results in an IGP update
            being flooded";
        }

        leaf threshold-specification {
          when "../threshold-type = 'THRESHOLD-CROSSED'" {
            description
              "The selection of whether mirrored or separate threshold
              values are to be used requires user specified thresholds to
              be set";
          }
          type enumeration {
            enum MIRRORED-UP-DOWN {
              description
                "MIRRORED-UP-DOWN indicates that a single set of
                threshold values should be used for both increasing
                and decreasing bandwidth when determining whether
                to trigger updated bandwidth values to be flooded
                in the IGP TE extensions.";
            }
            enum SEPARATE-UP-DOWN {
              description
                "SEPARATE-UP-DOWN indicates that a separate
                threshold values should be used for the increasing
                and decreasing bandwidth when determining whether
                to trigger updated bandwidth values to be flooded
                in the IGP TE extensions.";
            }
          }
          description

            "This value specifies whether a single set of threshold
            values should be used for both increasing and decreasing
            bandwidth when determining whether to trigger updated
            bandwidth values to be flooded in the IGP TE extensions.
            MIRRORED-UP-DOWN indicates that a single value (or set of
            values) should be used for both increasing and decreasing
            values, where SEPARATE-UP-DOWN specifies that the increasing
            and decreasing values will be separately specified";
        }

        leaf-list up-thresholds {
          when "../threshold-type = 'THRESHOLD-CROSSED'" +
            "and ../threshold-specification = 'SEPARATE-UP-DOWN'" {
              description
                "A list of up-thresholds can only be specified when the
```

```
                bandwidth update is triggered based on crossing a
                threshold and separate up and down thresholds are
                required";
            }
          type oc-types:percentage;
          description
            "The thresholds (expressed as a percentage of the maximum
            reservable bandwidth) at which bandwidth updates are to be
            triggered when the bandwidth is increasing.";
        }

        leaf-list down-thresholds {
          when "../threshold-type = 'THRESHOLD-CROSSED'" +
            "and ../threshold-specification = 'SEPARATE-UP-DOWN'" {
              description
                "A list of down-thresholds can only be specified when the
                bandwidth update is triggered based on crossing a
                threshold and separate up and down thresholds are
                required";
            }
          type oc-types:percentage;
          description
            "The thresholds (expressed as a percentage of the maximum
            reservable bandwidth) at which bandwidth updates are to be
            triggered when the bandwidth is decreasing.";
        }

        leaf-list up-down-thresholds {
          when "../threshold-type = 'THRESHOLD-CROSSED'" +
            "and ../threshold-specification = 'MIRRORED-UP-DOWN'" {
              description
                "A list of thresholds corresponding to both increasing
                and decreasing bandwidths can be specified only when an
                update is triggered based on crossing a threshold, and
                the same up and down thresholds are required.";
            }
          type oc-types:percentage;
          description
            "The thresholds (expressed as a percentage of the maximum
            reservable bandwidth of the interface) at which bandwidth
            updates are flooded - used both when the bandwidth is
            increasing and decreasing";
        }
      }

      grouping mpls-te-igp-flooding-bandwdith-if {
        description "Interface-level group for traffic engineering
        database flooding options options";
```

```
        container igp-flooding-bandwidth {
         description "Interface bandwidth change percentages
         that trigger update events into the IGP traffic
         engineering database (TED)";
         uses  mpls-te-igp-flooding-bandwidth_config;
        }
     }


     grouping  mpls-te-igp-flooding-bandwidth {
       description "Top level group for traffic engineering
         database flooding options";
       container igp-flooding-bandwidth {
         description "Interface bandwidth change percentages
           that trigger update events into the IGP traffic
           engineering database (TED)";
         container config {
           description "Configuration parameters for TED
             update threshold ";
           uses  mpls-te-igp-flooding-bandwidth_config;
         }
         container state {
           config false;
           description "State parameters for TED update threshold ";
           uses  mpls-te-igp-flooding-bandwidth_config;
         }
       }
     }


     grouping te_lsp_delay_config {
       description "Group for the timers goerning the delay
       in installation and cleanup of TE LSPs";

       leaf te-lsp-install-delay {
         type uint16 {
           range 0..3600;
         }
         units seconds;
         description "delay the use of newly installed te lsp for a
         specified amount of time.";
       }

       leaf te-lsp-cleanup-delay {
         type uint16;
         units seconds;
         description "delay the removal of old te lsp for a specified
         amount of time";
```

```
      }
    }

    grouping te-interface-attributes-top {
      description
        "Top level grouping for attributes
        for TE interfaces.";

      list interface {
        key interface-name;
        description "List of TE interfaces";

        leaf interface-name {
          type leafref {
            path "../config/interface-name";
            require-instance true;
          }
          description "The interface name";
        }

        container config {
          description
            "Configuration parameters related to TE interfaces:";
          uses te-interface-attributes-config;
        }

        container state {
          config false;
          description "State parameters related to TE interfaces";
          uses te-interface-attributes-config;
        }
      }
    }

    grouping te-interface-attributes-config {
      description "global level definitions for interfaces
        on which TE is run";

      leaf interface-name {
        type ocif:interface-ref;
        description "reference to interface name";
      }

      leaf te-metric {
        type uint32;
        description "TE specific metric for the link";
      }
```

```
        list srlg {
          key srlg-name;
          description "list of shared risk link groups on the
            interface";
          leaf srlg-name {
            type string;
            description "The SRLG group identifier";
          }
        }

        list admin-group {
          key admin-group-name;
          description "list of admin groups on the
            interface";
          leaf admin-group-name {
            type string;
            description "The admin group identifier";
          }
        }

        uses  mpls-te-igp-flooding-bandwdith-if;

      }

      grouping mpls-te-lsp-timers {
        description
          "Grouping for traffic engineering timers";
        container te-lsp-timers {
          description
          "definition for delays associated with setup
          and cleanup of TE LSPs";

          container config {
            description
            "Configuration parameters related
            to timers for TE LSPs";

            uses te_lsp_delay_config;
            uses te-tunnel-reoptimize_config;
          }
          container state {
            config false;
            description "State related to timers for TE LSPs";

            uses te_lsp_delay_config;
            uses te-tunnel-reoptimize_config;
          }
        }
```

```
     }

   container mpls {
       presence "top-level container for MPLS config and operational
       state";

       description "Anchor point for mpls configuration and operational
       data";

       container global {
         // entropy label support, label ranges will be added here.
        description "general mpls configuration applicable to any
        type of LSP and signaling protocol - label ranges,
        entropy label supportmay be added here";
       }

       container te-global-attributes {
         description "traffic-engineering global attributes";
         uses mpls-te-srlg-top;
         uses mpls-te-igp-flooding-bandwidth;
         uses mpls-admin-groups-top;
         uses mpls-te-lsp-timers;
       }

       container te-intf-attributes {
         description "traffic engineering attributes specific
         for interfaces";
         uses te-interface-attributes-top;
       }

       container signaling-protocols {
         description "top-level signaling protocol configuration";

         uses rsvp:rsvp-global;
         uses sr:sr-global;
         uses ldp:ldp-global;
       }

       container lsps {
         description "LSP definitions and configuration";

         container constrained-path {
           description "traffic-engineered LSPs supporting different
           path computation and signaling methods";
           uses explicit-paths-top;
           uses te-tunnels-top;

         }
```

```
        container unconstrained-path {
          description "LSPs that use the IGP-determined path, i.e., non
          traffic-engineered, or non constrained-path";

          uses igp-lsp-common;
          uses igp-lsp-setup;

        }

        container static-lsps {
          description "statically configured LSPs, without dynamic
            signaling";

          uses static-lsp-main;
        }
      }
    }

    // augment statements

    // rpc statements

    // notification statements
  }
                  <CODE ENDS>



                  <CODE BEGINS> file openconfig-mpls-types.yang
  module openconfig-mpls-types {

    yang-version "1";

    // namespace
    namespace "http://openconfig.net/yang/mpls-types";

    prefix "mplst";

    // meta
    organization "OpenConfig working group";

    contact
      "OpenConfig working group
      netopenconfig@googlegroups.com";

    description
      "General types for MPLS / TE data model";
```

```
     revision "2015-10-04" {
       description
         "Work in progress";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // using identities rather than enum types to simplify adding new
     // signaling protocols as they are introduced and supported
     identity path-setup-protocol {
       description "base identity for supported MPLS signaling
       protocols";
     }

     identity path-setup-rsvp {
       base path-setup-protocol;
       description "RSVP-TE signaling protocol";
     }

     identity path-setup-sr {
       base path-setup-protocol;
       description "Segment routing";
     }

     identity path-setup-ldp {
       base path-setup-protocol;
       description "lDP - RFC 5036";
     }


     identity protection-type {
       description "base identity for protection type";
     }

     identity unprotected {
       base protection-type;
       description "no protection is desired";
     }

     identity link-protection-requested {
       base protection-type;
       description "link protection is desired";
     }
```

```
identity link-node-protection-requested {
  base protection-type;
  description "node and link protection are both desired";
}

identity lsp-role {
  description
    "Base identity for describing the role of
     label switched path at the current node";
}

identity INGRESS {
  base "lsp-role";
  description
    "Label switched path is an ingress (headend)
     LSP";
}

identity EGRESS {
  base "lsp-role";
  description
    "Label switched path is an egress (tailend)
     LSP";
}

identity TRANSIT {
  base "lsp-role";
  description
    "Label switched path is a transit LSP";
}


identity tunnel-type {
  description
    "Base identity from which specific tunnel types are
    derived.";
}

identity P2P {
  base tunnel-type;
  description
    "TE point-to-point tunnel type.";
}

identity P2MP {
  base tunnel-type;
  description
    "TE point-to-multipoint tunnel type.";
```

```
     }


     identity lsp-oper-status {
       description
         "Base identity for LSP operational status";
     }

     identity DOWN {
       base "lsp-oper-status";
       description
         "LSP is operationally down or out of service";
     }

     identity UP {
       base "lsp-oper-status";
       description
         "LSP is operationally active and available
          for traffic.";
     }

     identity tunnel-admin-status {
       description
         "Base identity for tunnel administrative status";
     }

     identity ADMIN_DOWN {
       base "tunnel-admin-status";
       description
         "LSP is administratively down";
     }

     identity ADMIN_UP {
       base "tunnel-admin-status";
       description
         "LSP is administratively up";
     }

     // typedef statements
     typedef mpls-label {
       type union {
         type uint32 {
           range 16..1048575;
         }
         type enumeration {
           enum IPV4_EXPLICIT_NULL {
             value 0;
             description "valid at the bottom of the label stack,
```

```
            indicates that stack must be popped and packet forwarded
            based on IPv4 header";
          }
          enum ROUTER_ALERT {
            value 1;
            description "allowed anywhere in the label stack except
            the bottom, local router delivers packet to the local CPU
            when this label is at the top of the stack";
          }
          enum IPV6_EXPLICIT_NULL {
            value 2;
            description "valid at the bottom of the label stack,
            indicates that stack must be popped and packet forwarded
            based on IPv6 header";
          }
          enum IMPLICIT_NULL {
            value 3;
            description "assigned by local LSR but not carried in
            packets";
          }
          enum ENTROPY_LABEL_INDICATOR {
            value 7;
            description "Entropy label indicator, to allow an LSR
            to distinguish between entropy label and applicaiton
            labels RFC 6790";
          }
        }
      }
      description "type for MPLS label value encoding";
      reference "RFC 3032 - MPLS Label Stack Encoding";
    }

    typedef tunnel-type {
      type enumeration {
        enum P2P {
          description "point-to-point label-switched-path";
        }
        enum P2MP {
          description "point-to-multipoint label-switched-path";
        }
        enum MP2MP {
          description "multipoint-to-multipoint label-switched-path";
        }
      }
      description "defines the tunnel type for the LSP";
      reference
        "RFC 6388 - Label Distribution Protocol Extensions for
        Point-to-Multipoint and Multipoint-to-Multipoint Label Switched
```

```
         Paths
         RFC 4875 - Extensions to  Resource Reservation Protocol
         - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE
         Label Switched Paths (LSPs)";
     }

     typedef bandwidth-kbps {
       type uint64;
       units kbps;
     }

     typedef bandwidth-mbps {
       type uint64;
       units mbps;
     }

     typedef bandwidth-gbps {
       type uint64;
       units gbps;
     }

     // grouping statements

     // data definition statements

     // augment statements

     // rpc statements

     // notification statements

   }
                    <CODE ENDS>
```

6.2. MPLS LSP submodules

```
                    <CODE BEGINS> file openconfig-mpls-te.yang
   submodule openconfig-mpls-te {

     yang-version "1";

     belongs-to "openconfig-mpls" {
       prefix "mpls";
     }
```

```
// import some basic types
import ietf-inet-types { prefix inet; }
import openconfig-mpls-rsvp { prefix rsvp; }
import openconfig-mpls-sr { prefix sr; }
import openconfig-mpls-types {prefix mplst; }
import openconfig-types { prefix oc-types; }
import ietf-yang-types { prefix yang; }


// meta
organization "OpenConfig working group";

contact
  "OpenConfig working group
  netopenconfig@googlegroups.com";

description
  "Configuration related to constrained-path LSPs and traffic
  engineering.  These definitions are not specific to a particular
  signaling protocol or mechanism (see related submodules for
  signaling protocol-specific configuration).";

revision "2015-10-04" {
  description
    "Work in progress";
  reference "TBD";
}

// extension statements

// feature statements

// identity statements

// using identities for path comp method, though enums may also
// be appropriate if we decided these are the primary computation
// mechanisms in future.
identity path-computation-method {
  description
   "base identity for supported path computation
    mechanisms";
}

identity locally-computed {
  base path-computation-method;
  description
    "indicates a constrained-path LSP in which the
    path is computed by the local LER";
```

```
     }

     identity externally-queried {
       base path-computation-method;
       description
        "constrained-path LSP in which the path is
         obtained by querying an external source, such as a PCE server";
     }

     identity explicitly-defined {
       base path-computation-method;
       description
        "constrained-path LSP in which the path is
         explicitly specified as a collection of strict or/and loose
         hops";
     }

     // typedef statements

     typedef mpls-hop-type {
       type enumeration {
         enum LOOSE {
           description "loose hop in an explicit path";
         }
         enum STRICT {
           description "strict hop in an explicit path";
         }
       }
       description
        "enumerated type for specifying loose or strict
         paths";
     }

     typedef te-metric-type {
       type union {
         type enumeration {
           enum IGP {
             description
              "set the LSP metric to track the underlying
               IGP metric";
           }
         }
         type uint32;
       }
       description
        "union type for setting the LSP TE metric to a
         static value, or to track the IGP metric";
     }
```

```
     typedef cspf-tie-breaking {
       type enumeration {
         enum RANDOM {
           description
            "CSPF calculation selects a random path among
             multiple equal-cost paths to the destination";
         }
         enum LEAST_FILL {
           description
            "CSPF calculation selects the path with greatest
             available bandwidth";
         }
         enum MOST_FILL {
           description "CSPF calculation selects the path with the least
             available bandwidth";
         }
       }
       default RANDOM;
       description
        "type to indicate the CSPF selection policy when
         multiple equal cost paths are available";
     }


     // grouping statements

     grouping te-tunnel-reoptimize_config {
       description "Definition for reoptimize timer configuration";
       leaf te-lsp-reoptimize-timer {
         type uint16;
         units seconds;
         description
          "frequency of reoptimization of
           a traffic engineered LSP";
       }
     }

     grouping path-placement-constraints {
       description
         "Top level grouping for path placement constraints";

         container admin-groups {
           description
             "Include/Exclude constraints for link affinities";
           uses te-lsp-exclude-admin-group_config;
           uses te-lsp-include-any-admin-group_config;
           uses te-lsp-include-all-admin-group_config;
         }
```

```
     }

     grouping te-tunnel-bandwidth_config {
       description "Bandwidth configuration for TE LSPs";
       choice bandwidth {
         default explicit;
         description
          "select how bandwidth for the LSP will be
           specified and managed";
         case explicit {
           leaf set-bandwidth {
             type uint32;
             description
              "set bandwidth explicitly, e.g., using
               offline calculation";
           }
         }
         case auto {
           uses te-lsp-auto-bandwidth_config;
         }
       }
     }

     grouping te-lsp-auto-bandwidth_config {
       description "Configuration parameters related to autobandwidth";
       container auto-bandwidth {
         description
          "configure auto-bandwidth operation in
           which devices automatically adjust bandwidth to meet
           requirements";

         leaf enabled {
           type boolean;
           default false;
           description
            "enables mpls auto-bandwidth on the
             lsp";
         }

         leaf min-bw {
           type uint32;
           description
            "set the minimum bandwidth in Mbps for an
             auto-bandwidth LSP";
         }

         leaf max-bw {
           type uint32;
```

```
          description
           "set the maximum bandwidth in Mbps for an
            auto-bandwidth LSP";
        }

        leaf adjust-interval {
          type uint32;
          description
           "time in seconds between adjustments to
            LSP bandwidth";
        }

        leaf adjust-threshold {
          type oc-types:percentage;
          description
           "percentage difference between the LSP's
            specified bandwidth and its current bandwidth
            allocation -- if the difference is greater than the
            specified percentage, auto-bandwidth adjustment is
            triggered";
        }

        container overflow {
          description
           "configuration of MPLS overflow bandwidth
            adjustement for the LSP";
          uses te-lsp-overflow_config;
        }

        container underflow {
          description
           "configuration of MPLS underflow bandwidth
            adjustement for the LSP";
          uses te-lsp-underflow_config;
        }
      }
    }

    grouping  te-lsp-overflow_config {
      description
       "configuration for mpls lsp bandwidth
        overflow adjustment";

      leaf enabled {
        type boolean;
        default false;
        description
         "enables mpls lsp bandwidth overflow
```

```
            adjustment on the lsp";
        }

        leaf overflow-threshold {
          type oc-types:percentage;
          description
           "bandwidth percentage change to trigger
            an overflow event";

        }

        leaf trigger-event-count {
          type uint16;
          description
           "number of consecutive overflow sample
            events needed to trigger an overflow adjustment";
        }
      }

      grouping te-lsp-underflow_config {
        description
          "configuration for mpls lsp bandwidth
          underflow adjustment";

        leaf enabled {
          type boolean;
          default false;
          description
           "enables bandwidth underflow
            adjustment on the lsp";
        }

        leaf underflow-threshold {
          type oc-types:percentage;
          description
           "bandwidth percentage change to trigger
            and underflow event";
        }

        leaf trigger-event-count {
          type uint16;
          description
           "number of consecutive underflow sample
            events needed to trigger an underflow adjustment";
        }
      }

      grouping te-tunnel-metric_config {
```

```
      description "Configuration parameters related to LSP metric";
      leaf metric {
        type te-metric-type;
        description "LSP metric, either explicit or IGP";
      }
    }

    grouping te-lsp-exclude-admin-group_config {
      description
       "Configuration parameters related to admin-groups
        to exclude in path calculation";
      list exclude-groups {

        key exclude-admin-group-name;

        description
          "list of admin-groups to exclude in path calculation";

        leaf exclude-admin-group-name {
          type leafref {
            path "/mpls/te-global-attributes/mpls-admin-groups/" +
              "admin-group/admin-group-name";
          }
          description
            "name of the admin group -- references a defined admin
            group";
        }
      }
    }

    grouping te-lsp-include-all-admin-group_config {
      description
       "Configuration parameters related to admin-groups
        which all must be included in the path calculation";
      list include-all-groups {

        key all-admin-group-name;
        description
          "list of admin-groups of which all must be included";

        leaf all-admin-group-name {
          type leafref {
            path "/mpls/te-global-attributes/mpls-admin-groups/" +
              "admin-group/admin-group-name";
          }
          description
            "name of the admin group -- references a defined
            admin group";
```

```
        }
      }
    }

    grouping te-lsp-include-any-admin-group_config {
      description
       "Configuration parameters related to admin-groups
        of which one must be included in the path calculation";
      list include-any-groups {

        key any-admin-group-name;
        description
          "list of admin-groups of which one must be included";

        leaf any-admin-group-name {
          type leafref {
            path "/mpls/te-global-attributes/mpls-admin-groups/" +
              "admin-group/admin-group-name";
          }
          description
            "name of the admin group -- references a defined
            admin group";
        }
      }
    }

    grouping te-tunnel-protection_config {
      description
       "Configuration parameters related to LSP
        protection";
      leaf protection-style-requested {
        type identityref {
          base mplst:protection-type;
        }
        default mplst:unprotected;
        description
          "style of mpls frr protection desired: can be
          link, link-node or unprotected.";
      }
    }

    grouping te-lsp-comp-explicit {
      description
       "definitions for LSPs in which hops are explicitly
        specified";

      container explicit-path {
        description "LSP with explicit path specification";
```

```
        leaf path-name {
          type leafref {
            path "/mpls/lsps/constrained-path/"
              + "explicit-path/config/named-explicit-paths/name";
            require-instance true;
          }
          description "reference to a defined path";
        }
      }
    }

    grouping te-lsp-comp-queried {
      description "definitons for LSPs computed by querying a remote
        service, e.g., PCE server";

      container queried-path {
        description "LSP with path queried from an external server";

        leaf path-computation-server {
          type inet:ip-address;
          description
           "Address of the external path computation
            server";
        }
      }
    }

    grouping te-lsp-comp-local {
      description "definitons for locally-computed LSPs";

      container locally-computed {
        description "LSP with path computed by local ingress LSR";

        leaf use-cspf {
          type boolean;
          description "Flag to enable CSPF for locally computed LSPs";
        }
        leaf cspf-tiebreaker {
          type cspf-tie-breaking;
          description
            "Determine the tie-breaking method to choose between
            equally desirable paths during CSFP computation";
        }
      }
    }


    grouping explicit-route-subobject {
```

```
        description
          "The explicit route subobject grouping";
        choice type {
          description
            "The explicit route subobject type";
          case ipv4-address {
            description
              "IPv4 address explicit route subobject";
            leaf address {
              type inet:ip-address;
              description "router hop for the LSP path";
            }

            leaf hop-type {
              type mpls-hop-type;
              description "strict or loose hop";
            }
          }

          case label {
            leaf value {
              type uint32;
              description "the label value";
            }
            description
              "The Label ERO subobject";
          }
        }
      }


      // Explicit paths config somewhat following the IETF model
      grouping named-explicit-path_config {
        description
          "Global explicit path configuration
          grouping";
        list named-explicit-paths {
          key "name";
          description
            "A list of explicit paths";
          leaf name {
            type string;
            description
              "A string name that uniquely identifies
              an explicit path";
          }
          list explicit-route-objects {
            key "index";
```

```
          description
            "List of explicit route objects";
          leaf index {
            type uint8 {
              range "0..255";
            }
            description
              "Index of this explicit route object,
               to express the order of hops in path";
          }
          uses explicit-route-subobject;
        }
      }
    }

    grouping explicit-paths-top {
      description
        "common information for MPLS explicit path definition";
        list explicit-path {
          key name;
          description "Explicit path definition";

          leaf name {
            type leafref {
              path "/mpls/lsps/constrained-path/"
                + "explicit-path/config/named-explicit-paths/name";
              require-instance true;
            }
            description "definition for naming an explicit  path";

          }
          container config {
            description "configuration for an explicit path";
            uses named-explicit-path_config;
          }
          container state {
            config false;
            description "operational state for LSP path name";
            uses named-explicit-path_config;
          }
        }
    }

    grouping mpls-te-srlg_config {
      description
        "Configuration of various attributes associated
        with the SRLG";
```

```
        leaf srlg-name {
          type string;
          description "SRLG group identifier";
        }

        leaf srlg-value {
          type uint32;
          description "group ID for the SRLG";
        }

        leaf srlg-cost {
          type uint32;
          description
            "The cost of the SRLG to the computation
            algorithm";
        }
      }

      grouping mpls-te-srlg-members_config {
        description "Configuration of the membership of the SRLG";

        leaf from-address {
          type inet:ip-address;
          description "IP address of the a-side of the SRLG link";
        }

        leaf to-address {
          type inet:ip-address;
          description "IP address of the z-side of the SRLG link";
        }
      }

      grouping mpls-te-srlg-top {
        description
          "Top level grouping for MPLS shared
          risk link groups.";
        container mpls-te-srlg {
          description
            "Shared risk link groups attributes";
          list srlg {
            key srlg-name;
            description "List of shared risk link groups";

            leaf srlg-name {
              type leafref {
                path "../config/srlg-name";
                require-instance true;
              }
```

```
            description "The SRLG group identifier";
          }

          container config {
            description "Configuration parameters related to the SRLG";
            uses mpls-te-srlg_config;
          }

          container state {
            config false;
            description "State parameters related to the SRLG";
            uses mpls-te-srlg_config;
          }

          list members-list {
            key from-address;
            description
              "List of SRLG members, which are expressed
              as IP address endpoints of links contained in the SRLG";

            leaf from-address {
              type leafref {
                path "../config/from-address";
                require-instance true;
              }
              description "The from address of the link in the SRLG";
            }

            container config {
              description
                "Configuration parameters relating to the
                SRLG members";
              uses mpls-te-srlg-members_config;
            }

            container state {
              config false;
              description
                "State parameters relating to the SRLG
                members";
              uses mpls-te-srlg-members_config;
            }
          }
        }
      }
    }

    grouping tunnel-path_config {
```

```
        description
          "Tunnel path properties grouping";
        container path-computation-method {
          description
            "select and configure the way the LSP path is
            computed";

          leaf path-computation {
            type identityref {
              base path-computation-method;
            }
            description "path computation method to use with the LSP";
          }

          uses te-lsp-comp-explicit;
          uses te-lsp-comp-queried;
          uses te-lsp-comp-local;
        }

        uses path-placement-constraints;

        leaf no-cspf {
          type empty;
          description
            "Indicates no CSPF is to be attempted on this
            path.";
        }

        choice sigaling-specific-path-attributes {
          description "Signaling-protocol specific path attributes.";
          case RSVP {
            uses rsvp:rsvp-p2p-path-attributes_config;
          }
          case SR {
            uses sr:sr-path-attributes_config;
          }
        }
      }

      grouping te-tunnel_config {
        description
          "Configuration parameters relevant to a single
           traffic engineered tunnel.";

        leaf name {
          type string;
          description "The tunnel name";
        }
```

```
        leaf type {
          type identityref {
            base mplst:tunnel-type;
          }
          description "Tunnel type, p2p or p2mp";
        }

        leaf local-id {
          type union {
            type uint32;
            type string;
          }
          description
            "locally signficant optional identifier for the
             tunnel; may be a numerical or string value";
        }

        leaf description {
          type string;
          description "optional text description for the tunnel";
        }

        leaf admin-status {
          type identityref {
            base mplst:tunnel-admin-status;
          }
          default mplst:ADMIN_UP;
          description "TE tunnel administrative state.";
        }

        leaf preference {
          type uint8 {
            range "1..255";
          }
          description "Specifies a preference for this tunnel.
            A lower number signifies a better preference";
        }

        uses te-tunnel-metric_config;
        uses te-tunnel-bandwidth_config;
        uses te-tunnel-protection_config;
        uses te-tunnel-reoptimize_config;

        choice signaling-specific-tunnel-attributes {
          description "Signaling-protocol specific path attributes.";
          case RSVP {
            uses rsvp:rsvp-p2p-tunnel-attributes_config;
          }
```

```
       }

       choice tunnel-type {
         description
           "Describes tunnel by type type";
         case p2p {
           leaf destination {
             type inet:ip-address;
             description
               "P2P tunnel destination address";
           }
           /* P2P list of path(s) */
           list primary-paths {
             key "name";
             leaf name {
               type string;
               description "Path name";
             }
             description
               "List of primary paths for this
               tunnel.";
             leaf preference {
               type uint8 {
                 range "1..255";
               }
               description
                 "Specifies a preference for
                 this path. The lower the
                 number higher the
                 preference";
             }
             uses tunnel-path_config;
           }

           list secondary-paths {
             key "name";
             description
               "List of secondary paths for this
                 tunnel.";
             leaf name {
               type string;
               description "Path name";
             }
             leaf preference {
               type uint8 {
                 range "1..255";
               }
               description
```

```
                 "Specifies a preference for
                   this path. The lower the
                   number higher the
                   preference";
             }
             uses tunnel-path_config;
           }
         }
         case p2mp {
           // TODO - complete
         }
       }
     }

     grouping te-tunnel_state {
       description
         "Counters and statistical data relevent to a single
          tunnel.";


       leaf oper-status {
         type identityref {
           base mplst:lsp-oper-status;
         }
         description
           "The operational status of the TE tunnel";
       }

       leaf role {
         type identityref {
           base mplst:lsp-role;
         }
         description
           "The lsp role at the current node, whether it is headend,
           transit or tailend.";
       }

       container counters {
         description
           "State data for MPLS label switched paths. This state
           data is specific to a single label switched path.";

         leaf bytes {
           type yang:counter64;
           description
             "Number of bytes that have been forwarded over the
              label switched path.";
         }
```

```
         leaf packets {
           type yang:counter64;
           description
             "Number of pacets that have been forwarded over the
              label switched path.";
         }

         leaf path-changes {
           type yang:counter64;
           description
             "Number of path changes for the label switched path";
         }

         leaf state-changes {
           type yang:counter64;
           description
             "Number of state changes for the label switched path";
         }

         leaf online-time {
           type yang:date-and-time;
           description
             "Indication of the time the label switched path
              transitioned to an Oper Up or in-service state";
         }

         leaf current-path-time {
           type yang:date-and-time;
           description
             "Indicates the time the LSP switched onto its
              current path. This is reset upon a LSP path
              change.";
         }

         leaf next-reoptimization-time {
           type yang:date-and-time;
           description
             "Indicates the next scheduled time the LSP
              will be reoptimized.";
         }
       }
     }

     grouping te-tunnels-top {
       description
         "Top level grouping for TE tunnels";

       list tunnel {
```

```
         key "name type";
         description "List of TE tunnels";

         leaf name {
           type leafref {
             path "../config/name";
             require-instance true;
           }
           description "The tunnel name";
         }

         leaf type {
           type leafref {
             path "../config/type";
             require-instance true;
           }
           description "The tunnel type, p2p or p2mp.";
         }

         container config {
           description
             "Configuration parameters related to TE tunnels:";
           uses te-tunnel_config;
         }

         container state {
           config false;
           description "State parameters related to TE interfaces";
           uses te-tunnel_config;
           uses te-tunnel_state;
         }
       }
     }

   // data definition statements

   // augment statements

   // rpc statements

   // notification statements
   }
```

<CODE ENDS>

6.3. MPLS signaling protocol modules

```
                    <CODE BEGINS> file openconfig-mpls-rsvp.yang
module openconfig-mpls-rsvp {

  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/rsvp";

  prefix "rsvp";

  // import some basic types
  import ietf-inet-types { prefix inet; }
  import openconfig-mpls-types { prefix mplst; }
  import ietf-yang-types { prefix yang; }
  import openconfig-types { prefix oc-types; }


  // meta
  organization "OpenConfig working group";

  contact
    "OpenConfig working group
     netopenconfig@googlegroups.com";

  description
    "Configuration for RSVP-TE signaling, including global protocol
     parameters and LSP-specific configuration for constrained-path
     LSPs";

  revision "2015-09-18" {
    description
      "Initial revision";
    reference "TBD";
  }

  // extension statements

  // feature statements

  // identity statements

  // typedef statements

  // grouping statements
```

```
     grouping mpls-rsvp-soft-preemption_config {
       description "Configuration for MPLS soft preemption";
       leaf enable {
         type boolean;
         default false;
         description "Enables soft preemption on a node.";
       }

       leaf soft-preemption-timeout {
         type uint16 {
           range 0..max;
         }
         // The RFC actually recommends 30 seconds as default.
         default 0;
         description
           "Timeout value for soft preemption to revert
            to hard preemption";
         reference "RFC5712 MPLS-TE soft preemption";
       }
     }

     grouping mpls-rsvp-soft-preemption {
       description "Top level group for MPLS soft preemption";
       container soft-preemption {
         description
           "Protocol options relating to RSVP
            soft preemption";
         container config {
           description
             "Configuration parameters relating to RSVP
              soft preemption support";
           uses mpls-rsvp-soft-preemption_config;
         }
         container state {
           config false;
           description
             "State parameters relating to RSVP
              soft preemption support";
           uses mpls-rsvp-soft-preemption_config;
         }
       }
     }

     grouping mpls-rsvp-hellos_config {
       description "RSVP protocol options configuration.";

       leaf hello-interval {
         type uint16 {
```

```
          range 1000..60000;
        }
        units milliseconds;
        default 9000;
        description
          "set the interval in ms between RSVP hello
           messages";
        reference
          "RFC 3209: RSVP-TE: Extensions to RSVP for
           LSP Tunnels.
           RFC 5495: Description of the Resource
           Reservation Protocol - Traffic-Engineered
           (RSVP-TE) Graceful Restart Procedures";
      }

      leaf refresh-reduction {
        type boolean;
        default true;
        description
          "enables all RSVP refresh reduction message
           bundling, RSVP message ID, reliable message delivery
           and summary refresh";
        reference
          "RFC 2961 RSVP Refresh Overhead Reduction
           Extensions";
      }
    }

    grouping mpls-rsvp-hellos {
      description "Top level grouping for RSVP hellos parameters";
      // TODO: confirm that the described semantics are supported
      // on various implementations. Finer grain configuration
      // will be vendor-specific

      container rsvp-hellos {
        description "Top level container for RSVP hello parameters";
        container config {
          description
            "Configuration parameters relating to RSVP
             hellos";
          uses mpls-rsvp-hellos_config;
        }
        container state {
          config false;
          description "State information associated with RSVP hellos";
          uses mpls-rsvp-hellos_config;
        }
      }
```

```
     }

     grouping mpls-rsvp-subscription_config {
       description "RSVP subscription configuration";
       leaf subscription {
         type oc-types:percentage;
         description
           "percentage of the interface bandwidth that
            RSVP can reserve";
       }
     }
     grouping mpls-rsvp-subscription {
       description "Top level group for RSVP subscription options";
       container subscription {
         description
           "Bandwidth percentage reservable by RSVP
            on an interface";
         container config {
           description
             "Configuration parameters relating to RSVP
              subscription options";
           uses mpls-rsvp-subscription_config;
         }
         container state {
           config false;
           description
           "State parameters relating to RSVP
              subscription options";
           uses mpls-rsvp-subscription_config;
         }
       }
     }

     grouping mpls-rsvp-graceful-restart_config {
       description
         "Configuration parameters relating to RSVP Graceful-Restart";

       leaf enable {
         type boolean;
         default false;
         description "Enables graceful restart on the node.";
       }

       leaf restart-time {
         type uint32;
         description
           "Graceful restart time (seconds).";
         reference
```

```
            "RFC 5495: Description of the Resource
             Reservation Protocol - Traffic-Engineered
             (RSVP-TE) Graceful Restart Procedures";
        }
        leaf recovery-time {
          type uint32;
          description
            "RSVP state recovery time";
        }
      }

      grouping mpls-rsvp-graceful-restart {
        description
          "Top level group for RSVP graceful-restart
           parameters";
        container graceful-restart {
          description "TODO";
          container config {
            description
              "Configuration parameters relating to
               graceful-restart";
            uses mpls-rsvp-graceful-restart_config;
          }
          container state {
            config false;
            description
              "State information associated with
               RSVP graceful-restart";
            uses mpls-rsvp-graceful-restart_config;
          }
        }
      }

      grouping mpls-rsvp-authentication_config {
        description "RSVP authentication parameters container.";
        leaf enable {
          type boolean;
          default false;
          description "Enables RSVP authentication on the node.";
        }
        leaf authentication-key {
          type string {
            // Juniper supports 1..16 while
            // Cisco has a much bigger range, up to 60.
            length "1..32";
          }
          description
            "authenticate RSVP signaling
```

```
            messages";
        reference
          "RFC 2747: RSVP Cryptographic Authentication";
      }
    }

    grouping mpls-rsvp-authentication {
      description
        "Top level group for RSVP authentication,
         as per RFC2747";
      container authentication {
        description "TODO";
        container config {
          description
            "Configuration parameters relating
             to authentication";
          uses mpls-rsvp-authentication_config;
        }
        container state {
          config false;
          description
            "State information associated
             with authentication";
          uses mpls-rsvp-authentication_config;
        }
      }
    }

    grouping mpls-rsvp-protection_config {
      description "RSVP facility (link/node) protection configuration";

      leaf link-protection-style-requested {
        type identityref {
          base mplst:protection-type;
        }
        default mplst:link-node-protection-requested;
        description
          "style of mpls frr protection desired:
          link, link-node, or unprotected";
      }

      leaf bypass-optimize-interval {
        type uint16;
        units seconds;
        description
          "interval between periodic optimization
          of the bypass LSPs";
        // note: this is interface specific on juniper
```

```
        // on iox, this is global. need to resolve.
      }
      // to be completed, things like enabling link protection,
      // optimization times, etc.
    }

    grouping mpls-rsvp-link-protection {
      description "Top level group for RSVP protection";
      container protection {
        description "link-protection (NHOP) related configuration";
        container config {
          description "Configuration for link-protection";
          uses mpls-rsvp-protection_config;
        }
        container state {
          config false;
          description "State for link-protection";
          uses mpls-rsvp-protection_config;
        }
      }
    }

    grouping mpls-rsvp-error-statistics {
      description "RSVP-TE packet statistics";
      container error {
        description "RSVP-TE error statistics";
        leaf authentication-failure {
          type yang:counter32;
          description
            "Authentication failure count";
        }

        leaf path-error {
          type yang:counter32;
          description
            "Path error to client count";
        }

        leaf resv-error {
          type yang:counter32;
          description
            "Resv error to client count";
        }

        leaf path-timeout {
          type yang:counter32;
          description
            "Path timeout count";
```

```
        }

        leaf resv-timeout {
          type yang:counter32;
          description
            "Resv timeout count";
        }

        leaf rate-limit {
          type yang:counter32;
          description
            "Count of packets that were rate limited";
        }

        // TODO - complete the other error statistics
      }
    }

    grouping mpls-rsvp-protocol-statistics {
      description "RSVP protocol statistics";
      container protocol {
        description "RSVP-TE protocol statistics";
        leaf hello-sent {
          type yang:counter32;
          description
            "Hello sent count";
        }

        leaf hello-rcvd {
          type yang:counter32;
          description
            "Hello received count";
        }

        leaf path-sent {
          type yang:counter32;
          description
            "Path sent count";
        }

        leaf path-rcvd {
          type yang:counter32;
          description
            "Path received count";
        }

        // TODO - To be completed the other packet statistics
      }
```

```
     }

     grouping mpls-rsvp-statistics {
       description "Top level grouping for RSVP protocol state";
       uses mpls-rsvp-protocol-state;
     }

     grouping rsvp-global {
       description "Global RSVP protocol configuration";
       container rsvp-te {
         description "RSVP-TE global signaling protocol configuration";

         container rsvp-sessions {
           description "Configuration and state of RSVP sessions";

           container config {
             description
               "Configuration of RSVP sessions on the device";
           }

           container state {
             config false;
             description
               "State information relating to RSVP sessions
               on the device";
             uses mpls-rsvp-session-state;
           }
         }

         container rsvp-neighbors {
           description
             "Configuration and state for RSVP neighbors connecting
             to the device";

           container config {
             description "Configuration of RSVP neighbor information";
           }

           container state {
             config false;
             description
               "State information relating to RSVP neighbors";
             uses mpls-rsvp-neighbor-state;
           }
         }

         container global {
           description "Platform wide RSVP configuration and state";
```

```
          uses mpls-rsvp-graceful-restart;
          uses mpls-rsvp-soft-preemption;

          container statistics {
            config false;
            description "Platform wide RSVP state, including counters";
            // TODO - reconcile global and per-interface
            // protocol-related statistics

            container counters {
              config false;
              description
                "Platform wide RSVP statistics and counters";
              uses mpls-rsvp-global-protocol-state;
              uses mpls-rsvp-statistics;
            }
          }
        }

        container interface-attributes {
          // interfaces, bw percentages, hello timers, etc goes here";

          list interface {
            key interface-name;
            description "list of per-interface RSVP configurations";

            // TODO: update to interface ref -- move to separate
            // augmentation.
            leaf interface-name {
              type leafref {
                path "../config/interface-name";
                require-instance true;
              }
              description "references a configured IP interface";
            }


            container config {
              description
                "Configuration of per-interface RSVP parameters";

              leaf interface-name {
                type string;
                description "Name of configured IP interface";
              }
            }

            container state {
```

```
              config false;
              description
                "Per-interface RSVP protocol and state information";
              uses mpls-rsvp-interfaces-state;

              container counters {
                config false;
                description
                  "Interface specific RSVP statistics and counters";
                uses mpls-rsvp-protocol-state;
              }
            }

            uses mpls-rsvp-hellos;
            uses mpls-rsvp-authentication;
            uses mpls-rsvp-subscription;
            uses mpls-rsvp-link-protection;
          }
        }
      }
    }

    grouping rsvp-p2p-tunnel-attributes_config {
      description "properties of RSPP point-to-point paths";

     leaf source {
       type inet:ip-address;
       description
         "tunnel source address";
     }

     leaf soft-preemption {
        type boolean;
        default false;
        description "enables RSVP soft-preemption on this LSP";
      }
    }


    grouping rsvp-p2p-path-attributes_config {
      description "properties of RSPP point-to-point paths";
      leaf setup-priority {
        type uint8 {
          range 0..7;
        }
        default 7;
        description
          "preemption priority during LSP setup, lower is
```

```
            higher priority; default 7 indicates that LSP will not
            preempt established LSPs during setup";
          reference "RFC 3209 - RSVP-TE: Extensions to RSVP for
            LSP Tunnels";
        }

        leaf hold-priority {
          type uint8 {
            range 0..7;
          }
          default 0;
          description
            "preemption priority once the LSP is established,
             lower is higher priority; default 0 indicates other LSPs
             will not preempt the LSPs once established";
          reference "RFC 3209 - RSVP-TE: Extensions to RSVP for
            LSP Tunnels";
        }

        leaf retry-timer {
          type uint16 {
            range 1..600;
          }
          units seconds;
          description
            "sets the time between attempts to establish the
             LSP";
        }
      }

      grouping mpls-rsvp-neighbor-state {
        description "State information for RSVP neighbors";

        list rsvp-neighbor {
          key "neighbor-address";
          description
            "List of RSVP neighbors connecting to the device,
            keyed by neighbor address";

          leaf neighbor-address {
            type inet:ip-address;
            description "Address of RSVP neighbor";
          }

          leaf detected-interface {
            type string;
            description "Interface where RSVP neighbor was detected";
          }
```

```
         leaf neighbor-status {
           type enumeration {
             enum UP {
               description
                 "RSVP hello messages are detected from the neighbor";
             }
             enum DOWN {
               description
                 "RSVP neighbor not detected as up, due to a
                  communication failure or IGP notification
                  the neighbor is unavailable";
             }
           }
           description "Enumuration of possible RSVP neighbor states";
         }

         leaf neighbor-refresh-reduction {
           type boolean;
           description
             "Suppport of neighbor for RSVP refresh reduction";
           reference
             "RFC 2961 RSVP Refresh Overhead Reduction
              Extensions";
         }
       }
     }

     grouping mpls-rsvp-session-state {
       description "State information for RSVP TE sessions";
       list rsvp-session {
         key "source-port destination-port
          source-address destination-address";
         description "List of RSVP sessions";

         leaf source-address {
           type inet:ip-address;
           description "Origin address of RSVP session";
         }

         leaf destination-address {
           type inet:ip-address;
           description "Destination address of RSVP session";
         }

         leaf source-port {
           type uint16;
           description "RSVP source port";
           reference "RFC 2205";
```

```
         }

         leaf destination-port {
           type uint16;
           description "RSVP source port";
           reference "RFC 2205";

         }

         leaf session-state {
           type enumeration {
             enum UP {
               description "RSVP session is up";
             }
             enum DOWN {
               description "RSVP session is down";
             }
           }
           description "Enumeration of RSVP session states";
         }

         leaf session-type {
           type enumeration {
             enum SOURCE {
               description "RSVP session originates on this device";
             }
             enum TRANSIT {
               description "RSVP session transits this device only";
             }
             enum DESTINATION {
               description "RSVP session terminates on this device";
             }
           }
           description "Enumeration of possible RSVP session types";
         }

         leaf tunnel-id {
           type uint16;
           description "Unique identifier of RSVP session";
         }

         leaf label-in {
           type mplst:mpls-label;
           description
             "Incoming MPLS label associated with this RSVP session";
         }

         leaf label-out {
```

```
          type mplst:mpls-label;
          description
            "Outgoing MPLS label associated with this RSVP session";
        }

        leaf-list associated-lsps {
          type leafref {
            path "/mpls/lsps/constrained-path/tunnel/" +
            "config/name";
          }
          description
            "List of label switched paths associated with this RSVP
             session";
        }
      }
    } //rsvp-session-state

    grouping mpls-rsvp-interfaces-state {
      description "RSVP state information relevant to an interface";

      list bandwidth {
        key priority;
        description
          "Available and reserved bandwidth by priority on
           the interface.";

        leaf priority {
          type uint8 {
            range 0..7;
          }
          description
            "RSVP priority level for LSPs traversing the interface";
        }

        leaf available-bandwidth {
          type mplst:bandwidth-mbps;
          description "Bandwidth currently available";
        }

        leaf reserved-bandwidth {
          type mplst:bandwidth-mbps;
          description "Bandwidth currently reserved";
        }
      }

      leaf highwater-mark {
        type mplst:bandwidth-mbps;
        description "Maximum bandwidth ever reserved";
```

```
        }

        leaf active-reservation-count {
          type yang:gauge64;
          description "Number of active RSVP reservations";
        }
      }

      grouping mpls-rsvp-global-protocol-state {
        description "RSVP protocol statistics which may not apply
          on an interface, but are significant globally.";

        leaf path-timeouts {
          type yang:counter64;
          description "TODO";
        }

        leaf reservation-timeouts {
          type yang:counter64;
          description "TODO";
        }

        leaf rate-limited-messages {
          type yang:counter64;
          description "RSVP messages dropped due to rate limiting";
        }
      }

      grouping mpls-rsvp-protocol-state {
        description "RSVP protocol statistics and message counters";
        leaf in-path-messages {
          type yang:counter64;
          description "Number of received RSVP Path messages";
        }

        leaf in-path-error-messages {
          type yang:counter64;
          description "Number of received RSVP Path Error messages";
        }

        leaf in-path-tear-messages {
          type yang:counter64;
          description "Number of received RSVP Path Tear messages";
        }

        leaf in-reservation-messages {
          type yang:counter64;
          description "Number of received RSVP Resv messages";
```

```
        }

        leaf in-reservation-error-messages {
          type yang:counter64;
          description "Number of received RSVP Resv Error messages";
        }

        leaf in-reservation-tear-messages {
          type yang:counter64;
          description "Number of received RSVP Resv Tear messages";
        }

        leaf in-rsvp-hello-messages {
          type yang:counter64;
          description "Number of received RSVP hello messages";
        }

        leaf in-rsvp-srefresh-messages {
          type yang:counter64;
          description "Number of received RSVP summary refresh messages";
        }

        leaf in-rsvp-ack-messages {
          type yang:counter64;
          description
            "Number of received RSVP refresh reduction ack
             messages";
        }

        leaf out-path-messages {
          type yang:counter64;
          description "Number of sent RSVP PATH messages";
        }

        leaf out-path-error-messages {
          type yang:counter64;
          description "Number of sent RSVP Path Error messages";
        }

        leaf out-path-tear-messages {
          type yang:counter64;
          description "Number of sent RSVP Path Tear messages";
        }

        leaf out-reservation-messages {
          type yang:counter64;
          description "Number of sent RSVP Resv messages";
        }
```

```
        leaf out-reservation-error-messages {
          type yang:counter64;
          description "Number of sent RSVP Resv Error messages";
        }

        leaf out-reservation-tear-messages {
          type yang:counter64;
          description "Number of sent RSVP Resv Tear messages";
        }

        leaf out-rsvp-hello-messages {
          type yang:counter64;
          description "Number of sent RSVP hello messages";
        }

        leaf out-rsvp-srefresh-messages {
          type yang:counter64;
          description "Number of sent RSVP summary refresh messages";
        }

        leaf out-rsvp-ack-messages {
          type yang:counter64;
          description
            "Number of sent RSVP refresh reduction ack messages";
        }
      }



      // data definition statements

      // augment statements

      // rpc statements

      // notification statements

    }
                      <CODE ENDS>



                      <CODE BEGINS> file openconfig-mpls-sr.yang
    module openconfig-mpls-sr {

      yang-version "1";

      // namespace
```

```
     namespace "http://openconfig.net/yang/sr";

     prefix "sr";


     // import some basic types
     import ietf-inet-types { prefix inet; }
     import openconfig-mpls-types { prefix mplst; }

     // meta
     organization "OpenConfig working group";

     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "Configuration for MPLS with segment routing-based LSPs,
       including global parameters, and LSP-specific configuration for
       both constrained-path and IGP-congruent LSPs";

     revision "2015-10-14" {
       description
         "Work in progress";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     grouping srgb_config {

       // Matches the "global" configuration options in
       // draft-litkowski-spring-yang...
       // TODO: request to Stephane for this to be a separate
       // grouping such that it can be included.

       leaf lower-bound {
         type uint32;
         description
           "Lower value in the block.";
       }
       leaf upper-bound {
```

```
        type uint32;
        description
          "Upper value in the block.";
      }
      description
        "List of global blocks to be advertised.";
    }

    grouping srgb_state {
      description
        "State parameters relating to the SRGB";

      leaf size {
        type uint32;
        description
          "Number of indexes in the SRGB block";
      }
      leaf free {
        type uint32;
        description
          "Number of SRGB indexes that have not yet been allocated";
      }
      leaf used {
        type uint32;
        description
          "Number of SRGB indexes that are currently allocated";
      }

      // TODO: where do we put LFIB entries?

    }

    grouping adjacency-sid_config {
      description
        "Configuration related to an Adjacency Segment Identifier
        (SID)";

      // tuned from draft-litkowski-spring-yang
      // TODO: need to send a patch to Stephane

      leaf-list advertise {
        type enumeration {
          enum "PROTECTED" {
            description
              "Advertise an Adjacency-SID for this interface, which is
              eligible to be protected using a local protection
              mechanism on the local LSR. The local protection
              mechanism selected is dependent upon the configuration
```

```
                of RSVP-TE FRR or LFA elsewhere on the system";
            }
            enum UNPROTECTED {
              description
                "Advertise an Adajcency-SID for this interface, which is
                explicitly excluded from being protected by any local
                protection mechanism";
            }
          }
          description
            "Specifies the type of adjacency SID which should be
            advertised for the specified entity.";
        }

        leaf-list groups {
          type uint32;
          description
            "Specifies the groups to which this interface belongs.
            Setting a value in this list results in an additional AdjSID
            being advertised, with the S-bit set to 1. The AdjSID is
            assumed to be protected";
        }
      }

      grouping interface_config {
        description
          "Configuration parameters relating to a Segment Routing
          enabled interface";

        leaf interface {
          type string;
          // TODO: this should be changed to a leafref.
          description
            "Reference to the interface for which segment routing
            configuration is to be applied.";
        }
      }

      // grouping statements

      grouping sr-global {
        description "global segment routing signaling configuration";

        container segment-routing {
          description "SR global signaling config";

          list srgb {
            key "lower-bound upper-bound";
```

```
          uses srgb_config;
          container config {
            description
              "Configuration parameters relating to the Segment Routing
              Global Block (SRGB)";
            uses srgb_config;
          }
          container state {
            config false;
            description
              "State parameters relating to the Segment Routing Global
              Block (SRGB)";
            uses srgb_config;
            uses srgb_state;
          }
          description
            "List of Segment Routing Global Block (SRGB) entries. These
            label blocks are reserved to be allocated as domain-wide
            entries.";
        }

        list interfaces {
          key "interface";
          uses interface_config;
          container config {
            description
              "Interface configuration parameters for Segment Routing
              relating to the specified interface";
            uses interface_config;
          }
          container state {
            config false;
            description
              "State parameters for Segment Routing features relating
              to the specified interface";
            uses interface_config;
          }
          container adjacency-sid {
            description
              "Configuration for Adjacency SIDs that are related to
              the specified interface";
            container config {
              description
                "Configuration parameters for the Adjacency-SIDs
                that are related to this interface";
              uses adjacency-sid_config;
            }
            container state {
```

```
                config false;
                description
                  "State parameters for the Adjacency-SIDs that are
                  related to this interface";
                uses adjacency-sid_config;
              }
            }
            description
              "List of interfaces with associated segment routing
              configuration";
          }
        }
      }

      grouping sr-path-attributes_config {
        description
          "Configuration parameters relating to SR-TE LSPs";

        leaf sid-selection-mode {
          type enumeration {
            enum "ADJ-SID-ONLY" {
              description
                "The SR-TE tunnel should only use adjacency SIDs
                to build the SID stack to be pushed for the LSP";
            }
            enum "MIXED-MODE" {
              description
                "The SR-TE tunnel can use a mix of adjacency
                and prefix SIDs to build the SID stack to be pushed
                to the LSP";
            }
          }
          default "MIXED-MODE";
          description
            "The restrictions placed on the SIDs to be selected by the
            calculation method for the SR-TE LSP";
        }

        leaf sid-protection-required {
          type boolean;
          default "false";
          description
            "When this value is set to true, only SIDs that are
            protected are to be selected by the calculating method
            for the SR-TE LSP.";
        }
      }
```

```
     grouping sr_fec-address_config {
       description
         "Configuration parameters relating to a FEC that is to be
         advertised by Segment Routing";

       leaf fec-address {
         type inet:ip-prefix;
         description
           "FEC that is to be advertised as part of the Prefix-SID";
       }
     }

     grouping sr_fec-prefix-sid_config {
       description
         "Configuration parameters relating to the nature of the
         Prefix-SID that is to be advertised for a particular FEC";

       leaf type {
         type enumeration {
           enum "INDEX" {
             description
               "Set when the value of the prefix SID should be specified
               as an off-set from the SRGB's zero-value. When multiple
               SRGBs are specified, the zero-value is the minimum
               of their lower bounds";
           }
           enum "ABSOLUTE" {
             description
               "Set when the value of a prefix SID is specified as the
               absolute value within an SRGB. It is an error to specify
               an absolute value outside of a specified SRGB";
           }
         }
         default "INDEX";
         description
           "Specifies how the value of the Prefix-SID should be
           interpreted - whether as an offset to the SRGB, or as an
           absolute value";
       }

       leaf node-flag {
         type boolean;
         description
           "Specifies that the Prefix-SID is to be treated as a Node-SID
           by setting the N-flag in the advertised Prefix-SID TLV in the
           IGP";
       }
```

```
        leaf last-hop-behavior {
          type enumeration {
            enum "EXPLICIT-NULL" {
              description
                "Specifies that the explicit null label is to be used
                when the penultimate hop forwards a labelled packet to
                this Prefix-SID";
            }
            enum "UNCHANGED" {
              description
                "Specicies that the Prefix-SID's label value is to be
                left in place when the penultimate hop forwards to this
                Prefix-SID";
            }
            enum "PHP" {
              description
                "Specicies that the penultimate hop should pop the
                Prefix-SID label before forwarding to the eLER";
            }
          }
          description
            "Configuration relating to the LFIB actions for the
            Prefix-SID to be used by the penultimate-hop";
        }
      }


      grouping igp-tunnel-sr {
        description "defintiions for SR-signaled, IGP-based LSP tunnel
        types";

        container tunnel {
          description "contains configuration stanzas for different LSP
          tunnel types (P2P, P2MP, etc.)";

          leaf tunnel-type {
            type mplst:tunnel-type;
            description "specifies the type of LSP, e.g., P2P or P2MP";
          }

          container p2p-lsp {
            when "tunnel-type = 'P2P'" {
              description "container active when LSP tunnel type is
              point to point";
            }
            description "properties of point-to-point tunnels";

            list fec {
```

```
            key "fec-address";
            uses sr_fec-address_config;

            description
              "List of FECs that are to be originated as SR LSPs";

            container config {
              description
                "Configuration parameters relating to the FEC to be
                advertised by SR";
              uses sr_fec-address_config;
            }
            container state {
              config false;
              description
                "Operational state relating to a FEC advertised by SR";
              uses sr_fec-address_config;
            }
            container prefix-sid {
              description
                "Parameters relating to the Prefix-SID
                used for the originated FEC";

              container config {
                description
                  "Configuration parameters relating to the Prefix-SID
                  used for the originated FEC";
                uses sr_fec-prefix-sid_config;
              }
              container state {
                config false;
                description
                  "Operational state parameters relating to the
                  Prefix-SID used for the originated FEC";
                uses sr_fec-prefix-sid_config;
              }
            }
          }
        }
      }
    }

    grouping igp-lsp-sr-setup {
      description "grouping for SR-IGP path setup for IGP-congruent
      LSPs";

      container segment-routing {
```

```
         presence "Presence of this container sets the LSP to use
         SR signaling";

         description "segment routing signaling extensions for
         IGP-confgruent LSPs";

         uses igp-tunnel-sr;

       }
     }

     // data definition statements

     // augment statements

     // rpc statements

     // notification statements

   }
                    <CODE ENDS>



                    <CODE BEGINS> file openconfig-mpls-ldp.yang
   module openconfig-mpls-ldp {

     yang-version "1";

     // namespace
     namespace "http://openconfig.net/yang/ldp";

     prefix "ldp";

     // import some basic types
     import ietf-inet-types { prefix inet; }
     import openconfig-mpls-types { prefix mplst; }


     // meta
     organization "OpenConfig working group";

     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "Configuration of Label Distribution Protocol global and LSP-
```

```
       specific parameters for IGP-congruent LSPs";

     revision "2015-07-04" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     // grouping statements

      grouping ldp-global {
       description "global LDP signaling configuration";

       container ldp {
         description "LDP global signaling configuration";

         container timers {
           description "LDP timers";
         }
       }
     }


     grouping igp-tunnel-ldp {
       description "common defintiions for LDP-signaled LSP tunnel
       types";

       container tunnel {
         description "contains configuration stanzas for different LSP
         tunnel types (P2P, P2MP, etc.)";

         leaf tunnel-type {
           type mplst:tunnel-type;
             description "specifies the type of LSP, e.g., P2P or P2MP";
         }

         leaf ldp-type {
           type enumeration {
             enum BASIC {
               description "basic hop-by-hop LSP";
```

```
            }
            enum TARGETED {
              description "tLDP LSP";
            }
          }
          description "specify basic or targeted LDP LSP";
        }

        container p2p-lsp {
          when "tunnel-type = 'P2P'" {
            description "container active when LSP tunnel type is
            point to point";
          }

          description "properties of point-to-point tunnels";

          leaf-list fec-address {
            type inet:ip-prefix;
            description "Address prefix for packets sharing the same
            forwarding equivalence class for the IGP-based LSP";
          }
        }

        container p2mp-lsp {
          when "tunnel-type = 'P2MP'" {
            description "container is active when LSP tunnel type is
            point to multipoint";
          }

          description "properties of point-to-multipoint tunnels";

          // TODO: specify group/source, etc.

        }

        container mp2mp-lsp {
          when "tunnel-type = 'MP2MP'" {
            description "container is active when LSP tunnel type is
            multipoint to multipoint";
          }

          description "properties of multipoint-to-multipoint tunnels";

          // TODO: specify group/source, etc.

        }
      }
    }
```

```
     grouping igp-lsp-ldp-setup {
       description "grouping for LDP setup attributes";

       container ldp {

         presence "Presence of this container sets the LSP to use
         LDP signaling";

         description "LDP signaling setup for IGP-congruent LSPs";

         // include tunnel (p2p, p2mp, ...)

         uses igp-tunnel-ldp;

       }
     }

     // data definition statements

     // augment statements

     // rpc statements

     // notification statements

   }
                    <CODE ENDS>



                    <CODE BEGINS> file openconfig-mpls-igp.yang
   submodule openconfig-mpls-igp {

     yang-version "1";

     belongs-to "openconfig-mpls" {
       prefix "mpls";
     }


     // import some basic types
     import openconfig-mpls-ldp { prefix ldp; }
     import openconfig-mpls-sr { prefix sr; }



     // meta
     organization "OpenConfig working group";
```

```
     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "Configuration generic configuration parameters for IGP-congruent
       LSPs";

     revision "2015-07-04" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     // grouping statements


     grouping igp-lsp-common {
       description "common definitions for IGP-congruent LSPs";

       // container path-attributes {
       //  description "general path attribute settings for IGP-based
       //  LSPs";

       //}

     }


     grouping igp-lsp-setup {
       description "signaling protocol definitions for IGP-based LSPs";

       container path-setup-protocol {
         description "select and configure the signaling method for
             the LSP";

         // uses path-setup-common;
         uses ldp:igp-lsp-ldp-setup;
         uses sr:igp-lsp-sr-setup;
       }
```

```
     }


     // data definition statements

     // augment statements

     // rpc statements

     // notification statements

   }
                    <CODE ENDS>



                    <CODE BEGINS> file openconfig-mpls-static.yang
   submodule openconfig-mpls-static {

     yang-version "1";

     belongs-to "openconfig-mpls" {
       prefix "mpls";
     }

     // import some basic types
     import openconfig-mpls-types {prefix mplst; }
     import ietf-inet-types { prefix inet; }


     // meta
     organization "OpenConfig working group";

     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "Defines static LSP configuration";

     revision "2015-07-04" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements
```

```
     // feature statements

     // identity statements

     // typedef statements

     // grouping statements

     grouping static-lsp-common {
       description "common definitions for static LSPs";

       leaf next-hop {
         type inet:ip-address;
         description "next hop IP address for the LSP";
       }

       leaf incoming-label {
         type mplst:mpls-label;
         description "label value on the incoming packet";
       }

       leaf push-label {
         type mplst:mpls-label;
         description "label value to push at the current hop for the
         LSP";
       }
     }

     grouping static-lsp-main {
       description "grouping for top level list of static LSPs";


       list label-switched-path {
         key name;
         description "list of defined static LSPs";

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

         // TODO: separation into ingress, transit, egress may help
         // to figure out what exactly is configured, but need to
         // consider whether implementations can support the
         // separation
         container ingress {
           description "Static LSPs for which the router is an
             ingress node";
```

```
          uses static-lsp-common;
        }

        container transit {
          description "static LSPs for which the router is a
            transit node";

          uses static-lsp-common;
        }

        container egress {
          description "static LSPs for which the router is a
            egress  node";

          uses static-lsp-common;
        }
      }
    }

    // data definition statements

    // augment statements

    // rpc statements

    // notification statements

  }
```

<CODE ENDS>

7. Contributing Authors

The following people contributed significantly to this document and are listed below:

Ina Minei
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US
Email: inaminei@google.com

Anees Shaikh
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: aashaikh@google.com

Phil Bedard
Cox Communications
Atlanta, GA 30319
US
Email: phil.bedard@cox.com

8. Acknowledgements

The authors are grateful for valuable contributions to this document and the associated models from: Ebben Aires, Deepak Bansal, Nabil Bitar, Feihong Chen, Mazen Khaddam.

9. References

[I-D.ietf-idr-bgp-model]
Shaikh, A., Shakir, R., Patel, K., Hares, S., D’Souza, K., Bansal, D., Clemm, A., Alex, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-ietf-idr-bgp-model-00 (work in progress), July 2015.

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., rjs@rob.sh, r., Tantsura, J., and E. Crabbe, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-02 (work in progress), October 2015.

[I-D.openconfig-netmod-model-structure]
Shaikh, A., Shakir, R., D’Souza, K., and L. Fang, "Operational Structure and Organization of YANG Models", draft-openconfig-netmod-model-structure-00 (work in progress), March 2015.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

[I-D.shaikh-idr-bgp-model]
Shaikh, A., Shakir, R., Patel, K., Hares, S., D’Souza, K., Bansal, D., Clemm, A., Alex, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-shaikh-idr-bgp-model-02 (work in progress), June 2015.

[I-D.shakir-rtgwg-sr-performance-engineered-lsps]
Shakir, R., Vernals, D., and A. Capello, "Performance Engineered LSPs using the Segment Routing Data-Plane", draft-shakir-rtgwg-sr-performance-engineered-lsps-00 (work in progress), July 2013.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC5443] Jork, M., Atlas, A., and L. Fang, "LDP IGP Synchronization", RFC 5443, DOI 10.17487/RFC5443, March 2009, <http://www.rfc-editor.org/info/rfc5443>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

Authors' Addresses

Joshua George
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: jgeorge@google.com

Luyuan Fang
Microsoft
15590 NE 31st St
Redmond, WA 98052
US

Email: lufang@microsoft.com

Eric Osborne
Level 3

Email: eric.osborne@level3.com

Rob Shakir
Jive Communications, Inc.
1275 West 1600 North, Suite 100
Orem, UT 84057

Email: rjs@rob.sh

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

K. Raza
R. Asati
Cisco Systems, Inc.

X. Liu
Ericsson

S. Esale
Juniper Networks

X. Chen
Huawei Technologies

H. Shah
Ciena Corporation

July 8, 2016

YANG Data Model for MPLS LDP and mLDP
draft-raza-mpls-ldp-mldp-yang-04

Abstract

This document describes a YANG data model for Multi-Protocol Label Switching (MPLS) Label Distribution Protocol (LDP) and Multipoint LDP (mLDP).

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

Copyright (c) 2016 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

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

This document introduces a YANG data model for MPLS Label Distribution Protocol (LDP) [RFC5036] and Multipoint LDP (mLDP) [RFC6388]. For LDP, it also covers LDP IPv6 [RFC7552] and LDP capabilities [RFC5561].

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

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

This document is organized to define the data model for each of the above constructs (configuration, state, action, and notifications) in the sequence as listed earlier. Given that mLDP is tightly coupled with LDP, mLDP data model is defined under LDP tree and in the same sequence as listed above.

2. Specification of Requirements

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

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

3. LDP YANG Model

3.1. Overview

This document defines a new module named "ietf-mpls-ldp" for LDP/mLDP data model where this module augments /rt:routing/rt:control-plane-protocols that is defined in [I-D.ietf-netmod-routing-cfg].

There are four main containers in "ietf-mpls-ldp" module as follows:

- o Read-Write parameters for configuration (Discussed in Section 3.2)
- o Read-only parameters for operational state (Discussed in Section 3.3)
- o Notifications for events (Discussed in Section 3.4)
- o RPCs for executing commands to perform some action (Discussed in Section 3.5)

For the configuration and state data, this model follows the similar approach described in [I-D.openconfig-netmod-opstate] to represent the configuration (intended state) and operational (applied and derived) state. This means that for every configuration (rw) item, there is an associated (ro) item under "state" container to represent the applied state. Furthermore, protocol derived state is also kept under "state" tree corresponding to the protocol area (discovery, peer etc.). [Ed note: This document will be (re-)aligned with [I-D.openconfig-netmod-opstate] once that specification is adopted as a WG document]

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

```
module: ietf-mpls-ldp
     +-- rw routing
       +-- rw control-plane-protocols
          +-- rw mpls-ldp
              +-- rw global
              |   +-- rw config
              |       +-- rw ...
              |   +-- ro state
              |       +-- ro ...
              |       .
              +-- rw ...
              |
              |
              +-- rw ...
              ...

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

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

Figure 1

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

- o This module aims to address only the core LDP/mLDP parameters as per RFC specification, as well as some widely used and deployed non-RFC features (such as label policies, session authentication etc). Any vendor specific feature should be defined in a vendor-specific augmentation of this model.

- o Multi-topology LDP [RFC7307] and Multi-topology mLDP [I-D.iwijnand-mpls-mldp-multi-topology] are beyond the scope of this document.

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

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

hence we are treating the default forwarding table as just another VRF.

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

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

- o 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].

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

- o The use of grouping (templates) for bundling and grouping the configuration items is not employed in current revision, and is a subject for consideration in future.

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

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

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

  * Peer: An LDP session which has successfully progressed beyond its initialization phase and is either already exchanging the bindings or is ready to do so.

  It is to be noted that LDP Graceful Restart mechanisms defined in [RFC3478] allow keeping the exchanged bindings for some time after a session goes down with a peer. We call such a state -- i.e. keeping peer bindings without established or recovered peering -- a "stale" peer. When used in this document, the above terms will refer strictly to the semantics and definitions defined for them.

A graphical representation of LDP YANG data model is presented in Figure 3, Figure 5, Figure 11, and Figure 12. Whereas, the actual model definition in YANG is captured in Section 6.

While presenting the YANG tree view and actual .yang specification, this document assumes the reader is familiar with the concepts of YANG modeling, its presentation and its compilation.

3.2. 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 specification supports VRF-centric configuration. For implementations that support protocol-centric configuration, with provision for inheritance and items that apply to all vrfs, we recommend an augmentation of this model such that any protocol-centric or all-vrf configuration is defined under their designated containers within the standard network-instance (please see Section 3.2.2)

This model augments /rt:routing/rt:control-plane-protocols that is defined in [I-D.ietf-netmod-routing-cfg]. For LDP interfaces, this model refers the MPLS interface as defined under MPLS base specification [I-D.saad-mpls-base-yang]. Furthermore, as mentioned earlier, the configuration tree presents read-write intended configuration leave/items as well as read-only state of the applied configuration. The former is listed under "config" container and latter under "state" container.

Following is high-level configuration organization for LDP/mLDP:

```
module: ietf-mpls-ldp
   +-- routing
    +-- control-plane-protocols
          +-- mpls-ldp
              +-- global
              |    +-- ...
              |    +-- ...
              |    +-- address-family* [afi]
              |        +-- . . .
              |        +-- . . .
              |    +-- discovery
              |        +-- . . .
              +-- peers
                   +-- ...
                   +-- ...
```

Figure 2

Given the configuration hierarchy, the model allows inheritance such that an item in a child tree is able to derive value from a similar or related item in one of the parent. For instance, hello holdtime can be configured per-VRF or per-VRF-interface, thus allowing inheritance as well flexibility to override with a different value at any child level.

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

```
+--rw mpls-ldp!
   +--rw global
   |  +--rw config
   |  |  +--rw capability
   |  |  |  +--rw end-of-lib {capability-end-of-lib}?
   |  |  |  |  +--rw enable?   boolean
   |  |  |  +--rw typed-wildcard-fec {capability-typed-wildcard-fec}?
   |  |  |  |  +--rw enable?   boolean
   |  |  |  +--rw upstream-label-assignment {capability-upstream-label-assign
ment}?
   |  |  |     +--rw enable?   boolean
   |  |  +--rw graceful-restart
   |  |  |  +--rw enable?                boolean
   |  |  |  +--rw helper-enable?         boolean {graceful-restart-helper-mod
e}?
   |  |  |  +--rw reconnect-time?        uint16
   |  |  |  +--rw recovery-time?         uint16
   |  |  |  +--rw forwarding-holdtime?   uint16
   |  |  +--rw igp-synchronization-delay?   uint16
   |  |  +--rw lsr-id?                      yang:dotted-quad
```

```
      |  +--rw address-family* [afi]
      |  |  +--rw afi       ldp-address-family
      |  |  +--rw config
      |  |     +--rw enable?         boolean
      |  |     +--rw label-policy
      |  |     |  +--rw independent-mode
      |  |     |  |  +--rw assign {policy-label-assignment-config}?
      |  |     |  |  |  +--rw (prefix-option)?
      |  |     |  |  |     |  +--rw prefix-list?        prefix-list-ref
      |  |     |  |  |        +--rw host-routes-only?   boolean
      |  |     |  |  +--rw advertise
      |  |     |  |  |  +--rw explicit-null
      |  |     |  |  |  |  +--rw enable?        boolean
      |  |     |  |  |  |  +--rw prefix-list?   prefix-list-ref
      |  |     |  |  |  +--rw prefix-list?     prefix-list-ref
      |  |     |  |  +--rw accept
      |  |     |  |     +--rw prefix-list?   prefix-list-ref
      |  |     |  +--rw ordered-mode {policy-ordered-label-config}?
      |  |     |     +--rw egress-lsr
      |  |     |     |  +--rw prefix-list?   prefix-list-ref
      |  |     |     +--rw advertise
      |  |     |     |  +--rw prefix-list?   prefix-list-ref
      |  |     |     +--rw accept
      |  |     |        +--rw prefix-list?   prefix-list-ref
      |  |     +--rw ipv4
      |  |     |  +--rw transport-address?   inet:ipv4-address
      |  |     +--rw ipv6
      |  |        +--rw transport-address?   inet:ipv6-address
      |  +--rw discovery
      |  |  +--rw interfaces
      |  |  |  +--rw config
      |  |  |  |  +--rw hello-holdtime?   uint16
      |  |  |  |  +--rw hello-interval?   uint16
      |  |  |  +--rw interface* [interface]
      |  |  |     +--rw interface         mpls-interface-ref
      |  |  |     +--rw config
      |  |  |     |  +--rw hello-holdtime?              uint16
      |  |  |     |  +--rw hello-interval?              uint16
      |  |  |     |  +--rw igp-synchronization-delay?   uint16 {per-interface-ti
mer-config}?
      |  |  |     +--rw address-family* [afi]
      |  |  |        +--rw afi       ldp-address-family
      |  |  |        +--rw config
      |  |  |           +--rw enable?   boolean
      |  |  |           +--rw ipv4
      |  |  |           |  +--rw transport-address?   union
      |  |  |           +--rw ipv6
      |  |  |              +--rw transport-address?   union
      |  |  +--rw targeted
```

```
      |  |     +--rw config
      |  |     |  +--rw hello-holdtime?   uint16
      |  |     |  +--rw hello-interval?   uint16
      |  |     |  +--rw hello-accept {policy-extended-discovery-config}?
      |  |     |     +--rw enable?          boolean
      |  |     |     +--rw neighbor-list?   neighbor-list-ref
      |  |     +--rw address-family* [afi]
      |  |        +--rw afi      ldp-address-family
      |  |        +--rw ipv4
      |  |        |  +--rw target* [adjacent-address]
      |  |        |     +--rw adjacent-address    inet:ipv4-address
      |  |        |     +--rw config
      |  |        |        +--rw enable?          boolean
      |  |        |        +--rw local-address?   inet:ipv4-address
      |  |        +--rw ipv6
      |  |           +--rw target* [adjacent-address]
      |  |              +--rw adjacent-address    inet:ipv6-address
      |  |              +--rw config
      |  |                 +--rw enable?          boolean
      |  |                 +--rw local-address?   inet:ipv6-address
      |  +--rw forwarding-nexthop {forwarding-nexthop-config}?
      |  |  +--rw interfaces
      |  |     +--rw interface* [interface]
      |  |        +--rw interface         mpls-interface-ref
      |  |        +--rw address-family* [afi]
      |  |           +--rw afi       ldp-address-family
      |  |           +--rw config
      |  |              +--rw ldp-disable?    boolean
      |  +--rw label-policy
      |     +--rw independent-mode
      |     |  +--rw assign {policy-label-assignment-config}?
      |     |  |  +--rw (prefix-option)?
      |     |  |        +--rw prefix-list?        prefix-list-ref
      |     |  |        +--rw host-routes-only?   boolean
      |     |  +--rw advertise
      |     |  |  +--rw explicit-null
      |     |  |  |  +--rw enable?        boolean
      |     |  |  |  +--rw prefix-list?   prefix-list-ref
      |     |  |  +--rw prefix-list?     prefix-list-ref
      |     |  +--rw accept
      |     |     +--rw prefix-list?   prefix-list-ref
      |     +--rw ordered-mode {policy-ordered-label-config}?
      |        +--rw egress-lsr
      |        |  +--rw prefix-list?   prefix-list-ref
      |        +--rw advertise
      |        |  +--rw prefix-list?   prefix-list-ref
      |        +--rw accept
      |           +--rw prefix-list?   prefix-list-ref
```

```
      +--rw peers
         +--rw config
         |  +--rw session-authentication-md5-password?   string
         |  +--rw session-ka-holdtime?                   uint16
         |  +--rw session-ka-interval?                   uint16
         |  +--rw session-downstream-on-demand {session-downstream-on-demand-con
fig}?
         |     +--rw enable?      boolean
         |     +--rw peer-list?   peer-list-ref
         +--rw peer* [lsr-id]
            +--rw lsr-id    yang:dotted-quad
            +--rw config
               +--rw admin-down?                           boolean
               +--rw capability
               +--rw label-policy
               |  +--rw advertise
               |  |  +--rw prefix-list?   prefix-list-ref
               |  +--rw accept
               |     +--rw prefix-list?   prefix-list-ref
               +--rw session-authentication-md5-password?   string
               +--rw graceful-restart
               |  +--rw enable?           boolean
               |  +--rw reconnect-time?   uint16
               |  +--rw recovery-time?    uint16
               +--rw session-ka-holdtime?                   uint16
               +--rw session-ka-interval?                   uint16
               +--rw address-family
                  +--rw ipv4
                  |  +--rw label-policy
                  |     +--rw advertise
                  |     |  +--rw prefix-list?   prefix-list-ref
                  |     +--rw accept
                  |        +--rw prefix-list?   prefix-list-ref
                  +--rw ipv6
                     +--rw label-policy
                        +--rw advertise
                        |  +--rw prefix-list?   prefix-list-ref
                        +--rw accept
                           +--rw prefix-list?   prefix-list-ref
```

Figure 3

3.2.1. Configuration Hierarchy

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

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

Figure 4

Following subsections briefly explain these configuration areas.

3.2.1.1. Per-VRF parameters

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

3.2.1.1.1. Per-VRF global parameters

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

3.2.1.1.2. Per-VRF Capabilities parameters

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

container that is provided to override a capability that is enabled/ specified at VRF level.

3.2.1.1.3. Per-VRF Per-Address-Family parameters

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

3.2.1.1.4. Per-VRF Hello Discovery parameters

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

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

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

3.2.1.1.5. Per-VRF Peer parameters

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

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

3.2.1.1.6. Per-VRF Forwarding parameters

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

3.2.2. All-VRFs Configuration

[Ed note: TODO]

3.3. Operational State

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

Please note this state tree refers both the configuration "applied" state as well as the "derived" state related to the protocol. [Ed note: This is where this model differs presently from [I-D.openconfig-netmod-opstate] and subject to alignment in later revisions]

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

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
   +--rw mpls-ldp!
      +--rw global
      |  +--ro state
      |  |  +--ro capability
      |  |  |  +--ro end-of-lib {capability-end-of-lib}?
      |  |  |  |  +--ro enable?   boolean
      |  |  |  +--ro typed-wildcard-fec {capability-typed-wildcard-fec}?
      |  |  |  |  +--ro enable?   boolean
      |  |  |  +--ro upstream-label-assignment {capability-upstream-label-assign
ment}?
      |  |  |  |  +--ro enable?   boolean
      |  |  +--ro graceful-restart
      |  |  |  +--ro enable?                boolean
      |  |  |  +--ro helper-enable?         boolean {graceful-restart-helper-mod
e}?
      |  |  |  +--ro reconnect-time?        uint16
      |  |  |  +--ro recovery-time?         uint16
      |  |  |  +--ro forwarding-holdtime?   uint16
```

```
      |  |  +--ro igp-synchronization-delay?   uint16
      |  |  +--ro lsr-id?                      yang:dotted-quad
      |  +--rw address-family* [afi]
      |  |  +--rw afi       ldp-address-family
      |  |  +--ro state
      |  |     +--ro enable?         boolean
      |  |     +--ro label-policy
      |  |     |  +--ro independent-mode
      |  |     |  |  +--ro assign {policy-label-assignment-config}?
      |  |     |  |  |  +--ro (prefix-option)?
      |  |     |  |  |     +--:(prefix-list)
      |  |     |  |  |     |  +--ro prefix-list?        prefix-list-ref
      |  |     |  |  |     +--:(host-routes-only)
      |  |     |  |  |        +--ro host-routes-only?   boolean
      |  |     |  |  +--ro advertise
      |  |     |  |  |  +--ro explicit-null
      |  |     |  |  |  |  +--ro enable?        boolean
      |  |     |  |  |  |  +--ro prefix-list?   prefix-list-ref
      |  |     |  |  |  +--ro prefix-list?     prefix-list-ref
      |  |     |  |  +--ro accept
      |  |     |  |     +--ro prefix-list?   prefix-list-ref
      |  |     |  +--ro ordered-mode {policy-ordered-label-config}?
      |  |     |     +--ro egress-lsr
      |  |     |     |  +--ro prefix-list?   prefix-list-ref
      |  |     |     +--ro advertise
      |  |     |     |  +--ro prefix-list?   prefix-list-ref
      |  |     |     +--ro accept
      |  |     |        +--ro prefix-list?   prefix-list-ref
      |  |     +--ro ipv4
      |  |     |  +--ro transport-address?   inet:ipv4-address
      |  |     |  +--ro bindings
      |  |     |     +--ro address* [address]
      |  |     |     |  +--ro address      inet:ipv4-address
      |  |     |     |  +--ro advertisement-type?   advertised-received
      |  |     |     |  +--ro peer?        leafref
      |  |     |     +--ro fec-label* [fec]
      |  |     |        +--ro fec     inet:ipv4-prefix
      |  |     |        +--ro peer* [peer advertisement-type]
      |  |     |           +--ro peer                  leafref
      |  |     |           +--ro advertisement-type    advertised-received
      |  |     |           +--ro label?                mpls:mpls-label
      |  |     |           +--ro used-in-forwarding?   boolean
      |  |     +--ro ipv6
      |  |        +--ro transport-address?   inet:ipv6-address
      |  |        +--ro binding
      |  |           +--ro address* [address]
      |  |           |  +--ro address      inet:ipv6-address
      |  |           |  +--ro advertisement-type?   advertised-received
```

```
      |  |           |  +--ro peer?        leafref
      |  |           +--ro fec-label* [fec]
      |  |              +--ro fec     inet: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
      |  +--rw discovery
      |  |  +--rw interfaces
      |  |  |  +--ro state
      |  |  |  |  +--ro hello-holdtime?   uint16
      |  |  |  |  +--ro hello-interval?   uint16
      |  |  |  +--rw interface* [interface]
      |  |  |     +--ro state
      |  |  |     |  +--ro hello-holdtime?              uint16
      |  |  |     |  +--ro hello-interval?              uint16
      |  |  |     |  +--ro igp-synchronization-delay?   uint16 {per-interface-ti
mer-config}?
      |  |  |     |  +--ro next-hello?                  uint16
      |  |  |     +--rw address-family* [afi]
      |  |  |        +--rw afi       ldp-address-family
      |  |  |        +--ro state
      |  |  |           +--ro enable?   boolean
      |  |  |           +--ro ipv4
      |  |  |           |  +--ro transport-address?   union
      |  |  |           |  +--ro hello-adjacencies* [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?               leafref
      |  |  |           +--ro ipv6
      |  |  |              +--ro transport-address?   union
      |  |  |              +--ro hello-adjacencies* [adjacent-address]
      |  |  |                 +--ro adjacent-address    inet:ipv6-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?               leafref
      |  |  +--rw targeted
      |  |     +--ro state
      |  |     |  +--ro hello-holdtime?   uint16
      |  |     |  +--ro hello-interval?   uint16
      |  |     |  +--ro hello-accept {policy-extended-discovery-config}?
      |  |     |     +--ro enable?          boolean
      |  |     |     +--ro neighbor-list?   neighbor-list-ref
      |  |     +--rw address-family* [afi]
      |  |        +--rw afi      ldp-address-family
      |  |        +--ro state
      |  |        |  +--ro ipv4
      |  |        |  |  +--ro hello-adjacencies* [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?               leafref
      |  |        |  +--ro ipv6
      |  |        |     +--ro hello-adjacencies* [local-address adjacent-address
]
      |  |        |        +--ro local-address       inet:ipv6-address
      |  |        |        +--ro adjacent-address    inet:ipv6-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?               leafref
      |  |        +--rw ipv4
      |  |        |  +--rw target* [adjacent-address]
      |  |        |     +--rw adjacent-address    inet:ipv4-address
```

```
      |  |        |     +--ro state
      |  |        |        +--ro enable?          boolean
      |  |        |        +--ro local-address?   inet:ipv4-address
      |  |        +--rw ipv6
      |  |           +--rw target* [adjacent-address]
      |  |              +--rw adjacent-address    inet:ipv6-address
      |  |              +--ro state
      |  |                 +--ro enable?          boolean
      |  |                 +--ro local-address?   inet:ipv6-address
      |  +--rw forwarding-nexthop {forwarding-nexthop-config}?
      |  |  +--rw interfaces
      |  |     +--rw interface* [interface]
      |  |        +--rw interface         mpls-interface-ref
      |  |        +--rw address-family* [afi]
      |  |           +--rw afi       ldp-address-family
      |  |           +--ro state
      |  |              +--ro ldp-disable?    boolean
      +--rw peers
         +--ro state
         |  +--ro session-authentication-md5-password?   string
         |  +--ro session-ka-holdtime?                   uint16
         |  +--ro session-ka-interval?                   uint16
         |  +--ro session-downstream-on-demand {session-downstream-on-demand-con
fig}?
         |     +--ro enable?      boolean
         |     +--ro peer-list?   peer-list-ref
         +--rw peer* [lsr-id]
            +--rw lsr-id    yang:dotted-quad
            +--ro state
               +--ro admin-down?                           boolean
               +--ro capability
               +--ro label-policy
               |  +--ro advertise
               |  |  +--ro prefix-list?   prefix-list-ref
               |  +--ro accept
               |     +--ro prefix-list?   prefix-list-ref
               +--ro session-authentication-md5-password?   string
               +--ro graceful-restart
               |  +--ro enable?           boolean
               |  +--ro reconnect-time?   uint16
               |  +--ro recovery-time?    uint16
               +--ro session-ka-holdtime?                   uint16
               +--ro session-ka-interval?                   uint16
               +--ro address-family
               |  +--ro ipv4
               |  |  +--ro label-policy
               |  |  |  +--ro advertise
               |  |  |  |  +--ro prefix-list?   prefix-list-ref
               |  |  |  +--ro accept
```

```
                  |  |  |     +--ro prefix-list?   prefix-list-ref
                  |  |  +--ro hello-adjacencies* [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?          mpls-interface-ref
                  |  +--ro ipv6
                  |     +--ro label-policy
                  |     |  +--ro advertise
                  |     |  |  +--ro prefix-list?   prefix-list-ref
                  |     |  +--ro accept
                  |     |     +--ro prefix-list?   prefix-list-ref
                  |     +--ro hello-adjacencies* [local-address adjacent-address]
                  |        +--ro local-address       inet:ipv6-address
                  |        +--ro adjacent-address    inet:ipv6-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?          mpls-interface-ref
                  +--ro label-advertisement-mode
                  |  +--ro local?        label-adv-mode
                  |  +--ro peer?         label-adv-mode
                  |  +--ro negotiated?   label-adv-mode
                  +--ro next-keep-alive?                      uint16
                  +--ro peer-ldp-id?                          yang:dotted-quad
                  +--ro received-peer-state
                  |  +--ro graceful-restart
                  |  |  +--ro enable?           boolean
                  |  |  +--ro reconnect-time?   uint16
                  |  |  +--ro recovery-time?    uint16
                  |  +--ro capability
                  |     +--ro end-of-lib
```

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

Figure 5

3.3.1. Derived States

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

Neighbor Adjacencies

Peer

Bindings (FEC-label and address)

Capabilities

3.3.1.1. Adjacency state

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

Following captures high level tree hierarchy for neighbor adjacency state.

```
   +--rw mpls-ldp!
      +--rw discovery
         +--rw interfaces
         |  +--rw interface* [interface]
         |     +--rw address-family* [af]
         |        +--ro state
         |           +--ro ipv4 (or ipv6)
         |              +--ro hello-adjacencies* [adjacent-address]
         |                 +--ro adjacent-address
         |                    . . . .
         |                    . . . .
         +--rw targeted
            +--rw address-family* [afi]
                +--rw afi      address-family
                   +--ro state
                      +--ro ipv4 (or ipv6)
                         +--ro hello-adjacencies* [local-address adjacent-addres
s]
                            +--ro local-address
                            +--ro adjacent-address
                               . . . .
                               . . . .
```

Figure 6

3.3.1.2. Peer state

Peer related derived 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.

```
  +--rw mpls-ldp!
      +--rw peers
         +--rw peer* [lsr-id]
            +--rw lsr-id
            +--ro state
               +--ro session-ka-holdtime?
               +-- . . . .
               +-- . . . .
               +--ro capability
               +  +ro -- . . .
               +--ro address-family
               |  +--ro ipv4 (or ipv6)
               |     +--ro hello-adjacencies* [local-address adjacent-address]
               |        . . . .
               |        . . . .
               +--ro received-peer-state
               |  +--ro . . . .
               |  +--ro capability
               |     +--ro . . . .
               +--ro statistics
                  +-- . . . .
                  +-- . . . .
```

Figure 7

3.3.1.3. 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 200.1.1.1/32:
              advertised: local-label 16000
                peer 192.168.0.2:0
                peer 192.168.0.3:0
                peer 192.168.0.4:0
              received:
                peer 192.168.0.2:0, label 16002, used-in-forwarding=Yes
                peer 192.168.0.3:0, label 17002, used-in-forwarding=No
            FEC 200.1.1.2/32:
               . . . .
            FEC 201.1.0.0/16:
               . . . .


        Address bindings:
            Addr 1.1.1.1:
              advertised
            Addr 1.1.1.2:
              advertised
            Addr 2.2.2.2:
              received, peer 192.168.0.2
            Addr 2.2.2.22:
              received, peer 192.168.0.2
            Addr 3.3.3.3:
              received, peer 192.168.0.3
            Addr 3.3.3.33:
              received, peer 192.168.0.3
```

Figure 8

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.

```
+--rw mpls-ldp!
   +--rw global
      +--rw address-family* [afi]
         +--rw afi       address-family
         +--ro state
            +--ro ipv4 (or ipv6)
               +--ro bindings
                  +--ro address* [address]
                  |  +--ro address
                  |  +--ro direction?   advertised-received
                  |  +--ro peer?        leafref
                  +--ro fec-label* [fec]
                     +--ro fec     inet:ipv4-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 9

3.3.1.4. 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 global
   |  +--ro state
   |     +--ro capability
   |        +--ro . . . .
   |        +--ro . . . .
   +--rw peers
      +--rw peer* [lsr-id]
         +--rw lsr-id    yang:dotted-quad
         +--ro state
            +--ro received-peer-state
               +--ro capability
                  +--ro . . . .
                  +--ro . . . .
```

Figure 10

3.4. Notifications

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

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

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

Figure 11

3.5. Actions

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

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

```
module: ietf-mpls-ldp
rpcs:
   +---x mpls-ldp-clear-peer
   |  +---w input
   |     +---w lsr-id?   union
   +---x mpls-ldp-clear-hello-adjacency
   |  +---w input
   |     +---w hello-adjacency
   |        +---w (hello-adjacency-type)?
   |           +--:(targeted)
   |           |  +---w targeted!
   |           |     +---w target-address?   inet:ip-address
   |           +--:(link)
   |              +---w link!
   |                 +---w next-hop-interface?   mpls-interface-ref
   |                 +---w next-hop-address?     inet:ip-address
   +---x mpls-ldp-clear-peer-statistics
      +---w input
         +---w lsr-id?   union
```

Figure 12

4. mLDP YANG Model

4.1. Overview

Due to tight dependency of mLDP on LDP, mLDP model builds on top of LDP model defined earlier in the document. Following are the main mLDP areas and documents that are within the scope of this model:

- o mLDP Base Specification [RFC6388]
- o mLDP Recursive FEC [RFC6512]
- o Targeted mLDP [RFC7060]
- o mLDP Fast-Reroute (FRR)
  - * Node Protection [RFC7715]
  - * Multicast-only
- o Hub-and-Spoke Multipoint LSPs [RFC7140]
- o mLDP In-band Signaling [RFC6826] (future revision)
- o mLDP In-band signaling in a VRF [RFC7246]

- o mLDP In-band Signaling with Wildcards [RFC7438] (future revision)
- o Configured Leaf LSPs (manually provisioned)

[Ed Note: Some of the topics in the above list are to be addressed/ added in later revision of this document].

4.2. Configuration

4.2.1. Configuration Hierarchy

In terms of overall configuration layout, following figure highlights extensions to LDP configuration model to incorporate mLDP:

```
      +-- mpls-ldp
         +-- ...
         +-- ...
         +-- mldp
         |  +-- ...
         |  +-- ...
         |  +-- address-family* [af]
         |     +-- af
         |        +-- ...
         |        +-- ...
         +-- global
         |  +-- ...
         |  +-- capability
         |     +-- ...
         |     +-- ...
         |     +-- mldp
         |        +-- ...
         |        +-- ...
         +-- discovery
         |  +-- ...
         |  +-- ...
         +-- forwarding-nexthop
         |  +-- interfaces
         |     +-- interface* [interface]
         |        +-- interface
         |        +-- address-family* [af]
         |           +-- af
         |           +-- ...
         |           +-- mldp-disable
         +-- peers
            +-- ...
            +-- ...
            +-- peer* [lsr-id]
                +-- ...
                +-- ...
                +-- capability
                    +-- ...
                    +-- ...
                    +-- mldp
                        +-- ...
                        +-- ...
```

Figure 13

From above hierarchy, we can categorize mLDP configuration parameters into two types:

- o Parameters that leverage/extend LDP containers and parameters

- o Parameters that are mLDP specific

Following subsections first describe mLDP specific configuration parameters, followed by those leveraging LDP.

4.2.2. mldp container

mldp container resides directly under "mpls-ldp" and holds the configuration related to items that are mLDP specific. The main items under this container are:

- o mLDP enabling: To enable mLDP under a (VRF) routing instance, mldp container is enabled under LDP. Given that mLDP requires LDP signalling, it is not sensible to allow disabling LDP control plane under a (VRF) network-instance while requiring mLDP to be enabled for the same. However, if a user wishes only to allow signalling for multipoint FECs on an LDP/mLDP enabled VRF instance, he/she can use LDP label-policies to disable unicast FECs under the VRF.

- o mLDP per-AF features: mLDP manages its own list of IP address-families and the features enabled underneath. The per-AF mLDP configuration items include:

  - * Multicast-only FRR: This enables Multicast-only FRR functionality for a given AF under mLDP. The feature allows route-policy to be configured for finer control/applicability of the feature.

  - * Recursive FEC: The recursive-fec feature [RFC6512] can be enabled per AF with a route-policy.

  - * Configured Leaf LSPs: To provision multipoint leaf LSP manually, a container is provided per-AF under LDP. The configuration is flexible and allows a user to specify MP LSPs of type p2mp or mp2mp with IPv4 or IPv6 root address(es) by using either LSP-Id or (S,G).

Targeted mLDP feature specification [RFC7060] do not require any mLDP specific configuration. It, however, requires LDP upstream-label-assignment capability [RFC6389] to be enabled.

4.2.3. Leveraging LDP containers

mLDP configuration model leverages following configuration areas and containers that are already defined for LDP:

- o Capabilities: A new container "mldp" is defined under Capabilities container. This new container specifies any mLDP specific capabilities and their parameters. Moreover, a new "mldp" container is also added under per-peer capability container to override/control mLDP specific capabilities on a peer level. In the scope of this document, the most important capabilities related to mLDP are p2mp, mp2mp, make-before-break, hub-and-spoke, and node-protection.

- o Discovery and Peer: mLDP requires LDP discovery and peer procedures to form mLDP peering. A peer is treated as mLDP peer only when either P2MP or MP2MP capabilities have been successfully exchanged with the peer. If a user wish to selectively enable or disable mLDP with a LDP-enabled peer, he/she may use per-peer mLDP capabilities configuration. [Ed Note: The option to control mLDP enabling/disabling on a peer-list is being explored for future]. In most common deployments, it is desirable to disable mLDP (capabilities announcements) on a targeted-only LDP peering, where targeted-only peer is the one whose discovery sources are targeted only. In future revision, a configuration option for this support will also be provided.

- o Forwarding: By default, mLDP is allowed to select any of the LDP enabled interface as a downstream interface towards a nexthop (LDP/mLDP peer) for MP LSP programming. However, a configuration option is provided to allow mLDP to exclude a given interface from such a selection. Note that such a configuration option will be useful only when there are more than one interfaces available for the downstream selection.

This goes without saying that mLDP configuration tree follows the same approach as LDP, where the tree comprise leafs for intended configuration.

4.2.4. YANG tree

The following figure captures the YANG tree for mLDP configuration. To keep the focus, the figure has been simplified to display only mLDP items without any LDP items.

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
   +--rw mpls-ldp!
```

```
      +--rw global
      |  +--rw config
      |  |  +--rw capability
      |  |     +--rw mldp {mldp}?
      |  |        +--rw p2mp
      |  |        |  +--rw enable?   boolean
      |  |        +--rw mp2mp
      |  |        |  +--rw enable?   boolean
      |  |        +--rw make-before-break
      |  |        |  +--rw enable?             boolean
      |  |        |  +--rw switchover-delay?   uint16
      |  |        |  +--rw timeout?            uint16
      |  |        +--rw hub-and-spoke {capability-mldp-hsmp}?
      |  |        |  +--rw enable?   boolean
      |  |        +--rw node-protection {capability-mldp-node-protection}?
      |  |           +--rw plr?           boolean
      |  |           +--rw merge-point
      |  |              +--rw enable?                            boolean
      |  |              +--rw targeted-session-teardown-delay?   uint16
      |  +--rw mldp {mldp}?
      |  |  +--rw config
      |  |  |  +--rw enable?   boolean
      |  |  +--rw address-family* [afi]
      |  |     +--rw afi                     ldp-address-family
      |  |     +--rw config
      |  |     |  +--rw multicast-only-frr {mldp-mofrr}?
      |  |     |  |  +--rw prefix-list?   prefix-list-ref
      |  |     |  +--rw recursive-fec
      |  |     |     +--rw prefix-list?   prefix-list-ref
      |  |     +--rw configured-leaf-lsps
      |  |        +--rw p2mp
      |  |        |  +--rw roots-ipv4
      |  |        |  |  +--rw root* [root-address]
      |  |        |  |     +--rw root-address    inet:ipv4-address
      |  |        |  |     +--rw lsp* [lsp-id source-address group-address]
      |  |        |  |        +--rw lsp-id            uint16
      |  |        |  |        +--rw source-address    inet:ipv4-address
      |  |        |  |        +--rw group-address     inet:ipv4-address-no-zone
      |  |        |  +--rw roots-ipv6
      |  |        |     +--rw root* [root-address]
      |  |        |        +--rw root-address    inet:ipv6-address
      |  |        |        +--rw lsp* [lsp-id source-address group-address]
      |  |        |           +--rw lsp-id            uint16
      |  |        |           +--rw source-address    inet:ipv6-address
      |  |        |           +--rw group-address     inet:ipv6-address-no-zone
      |  |        +--rw mp2mp
      |  |           +--rw roots-ipv4
      |  |           |  +--rw root* [root-address]
```

```
      |  |           |     +--rw root-address    inet:ipv4-address
      |  |           |     +--rw lsp* [lsp-id source-address group-address]
      |  |           |        +--rw lsp-id            uint16
      |  |           |        +--rw source-address    inet:ipv4-address
      |  |           |        +--rw group-address     inet:ipv4-address-no-zone
      |  |           +--rw roots-ipv6
      |  |              +--rw root* [root-address]
      |  |                 +--rw root-address    inet:ipv6-address
      |  |                 +--rw lsp* [lsp-id source-address group-address]
      |  |                    +--rw lsp-id            uint16
      |  |                    +--rw source-address    inet:ipv6-address
      |  |                    +--rw group-address     inet:ipv6-address-no-zone
      |  +--rw forwarding-nexthop {forwarding-nexthop-config}?
      |     +--rw interfaces
      |        +--rw interface* [interface]
      |           +--rw interface         if:interface-ref
      |           +--rw address-family* [afi]
      |              +--rw afi       address-family
      |              +--rw config
      |                 +--rw mldp-disable?   boolean {mldp}?
      +--rw peers
         +--rw peer* [lsr-id]
               +--rw capability
                  +--rw mldp {mldp}?
                     +--rw p2mp
                     |  +--rw enable?   boolean
                     +--rw mp2mp
                     |  +--rw enable?   boolean
                     +--rw make-before-break
                     |  +--rw enable?             boolean
                     |  +--rw switchover-delay?   uint16
                     |  +--rw timeout?            uint16
                     +--rw hub-and-spoke {capability-mldp-hsmp}?
                     |  +--rw enable?   boolean
                      +--rw node-protection {capability-mldp-node-protection}?
                        +--rw plr?           boolean
                        +--rw merge-point
                           +--rw enable?                           boolean
                           +--rw targeted-session-teardown-delay?   uint16
```

Figure 14

4.3. Operational State

Operational state of mLDP can be queried and obtained from this read-only container "mldp" which resides under mpls-ldp container.

Please note this state tree refers both the configuration "applied" state as well as the "derived" state related to the mLDP protocol.

Following is a simplified graphical representation of the data model for mLDP operational state:

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
   +--rw mpls-ldp!
      +--rw global
      |  +--ro state
      |  |  +--ro capability
      |  |     +--ro mldp {mldp}?
      |  |        +--ro p2mp
      |  |        |  +--ro enable?   boolean
      |  |        +--ro mp2mp
      |  |        |  +--ro enable?   boolean
      |  |        +--ro make-before-break
      |  |        |  +--ro enable?             boolean
      |  |        |  +--ro switchover-delay?   uint16
      |  |        |  +--ro timeout?            uint16
      |  |        +--ro hub-and-spoke {capability-mldp-hsmp}?
      |  |        |  +--ro enable?   boolean
      |  |        +--ro node-protection {capability-mldp-node-protection}?
      |  |           +--ro plr?           boolean
      |  |           +--ro merge-point
      |  |              +--ro enable?                           boolean
      |  |              +--ro targeted-session-teardown-delay?   uint16
      |  |
      |  +--rw mldp {mldp}?
      |  |  +--ro state
      |  |     +--ro enable?   boolean
      |  |  +--rw address-family* [afi]
      |  |     +--rw afi                     ldp-address-family
      |  |     +--ro state
      |  |        +--ro multicast-only-frr {mldp-mofrr}?
      |  |        |  +--ro prefix-list?   prefix-list-ref
      |  |        +--ro recursive-fec
      |  |        |  +--ro prefix-list?   prefix-list-ref
      |  |        +--ro ipv4
      |  |        |  +--ro roots
      |  |        |  |  +--ro root* [root-address]
      |  |        |  |     +--ro root-address    inet:ipv4-address
      |  |        |  |     +--ro is-self?        boolean
      |  |        |  |     +--ro reachability* [address interface]
      |  |        |  |        +--ro address      inet:ipv4-address
      |  |        |  |        +--ro interface    mpls-interface-ref
```

```
      |  |        |  |        +--ro peer?        leafref
      |  |        |  +--ro bindings
      |  |        |     +--ro opaque-type-lspid
      |  |        |     |  +--ro fec-label* [root-address lsp-id recur-root-addr
ess recur-rd]
      |  |        |     |     +--ro root-address          inet:ipv4-address
      |  |        |     |     +--ro lsp-id                uint32
      |  |        |     |     +--ro recur-root-address    inet:ip-address
      |  |        |     |     +--ro recur-rd              route-distinguisher
      |  |        |     |     +--ro multipoint-type?      multipoint-type
      |  |        |     |     +--ro peer* [direction peer advertisement-type]
      |  |        |     |        +--ro direction             downstream-upstream
      |  |        |     |        +--ro peer                  leafref
      |  |        |     |        +--ro advertisement-type    advertised-received
      |  |        |     |        +--ro label?                mpls:mpls-label
      |  |        |     |        +--ro mbb-role?             enumeration
      |  |        |     |        +--ro mofrr-role?           enumeration
      |  |        |     +--ro opaque-type-src
      |  |        |     |  +--ro fec-label* [root-address source-address group-a
ddress rd recur-root-address recur-rd]
      |  |        |     |     +--ro root-address          inet:ipv4-address
      |  |        |     |     +--ro source-address        inet:ip-address
      |  |        |     |     +--ro group-address         inet:ip-address-no-zon
e
      |  |        |     |     +--ro rd                    route-distinguisher
      |  |        |     |     +--ro recur-root-address    inet:ip-address
      |  |        |     |     +--ro recur-rd              route-distinguisher
      |  |        |     |     +--ro multipoint-type?      multipoint-type
      |  |        |     |     +--ro peer* [direction peer advertisement-type]
      |  |        |     |        +--ro direction             downstream-upstream
      |  |        |     |        +--ro peer                  leafref
      |  |        |     |        +--ro advertisement-type    advertised-received
      |  |        |     |        +--ro label?                mpls:mpls-label
      |  |        |     |        +--ro mbb-role?             enumeration
      |  |        |     |        +--ro mofrr-role?           enumeration
      |  |        |     +--ro opaque-type-bidir
      |  |        |        +--ro fec-label* [root-address rp group-address rd re
cur-root-address recur-rd]
      |  |        |           +--ro root-address          inet:ipv4-address
      |  |        |           +--ro rp                    inet:ip-address
      |  |        |           +--ro group-address         inet:ip-address-no-zon
e
      |  |        |           +--ro rd                    route-distinguisher
      |  |        |           +--ro recur-root-address    inet:ip-address
      |  |        |           +--ro recur-rd              route-distinguisher
      |  |        |           +--ro multipoint-type?      multipoint-type
      |  |        |           +--ro peer* [direction peer advertisement-type]
      |  |        |              +--ro direction             downstream-upstream
      |  |        |              +--ro peer                  leafref
      |  |        |              +--ro advertisement-type    advertised-received
      |  |        |              +--ro label?                mpls:mpls-label
      |  |        |              +--ro mbb-role?             enumeration
      |  |        |              +--ro mofrr-role?           enumeration
```

```
      |  |        +--ro ipv6
      |  |           +--ro roots
      |  |           |  +--ro root* [root-address]
      |  |           |     +--ro root-address    inet:ipv6-address
      |  |           |     +--ro is-self?        boolean
      |  |           |     +--ro reachability* [address interface]
      |  |           |        +--ro address      inet:ipv6-address
      |  |           |        +--ro interface    mpls-interface-ref
      |  |           |        +--ro peer?        leafref
      |  |           +--ro bindings
      |  |              +--ro opaque-type-lspid
      |  |              |  +--ro fec-label* [root-address lsp-id recur-root-addr
ess recur-rd]
      |  |              |     +--ro root-address          inet:ipv6-address
      |  |              |     +--ro lsp-id                uint32
      |  |              |     +--ro recur-root-address    inet:ip-address
      |  |              |     +--ro recur-rd              route-distinguisher
      |  |              |     +--ro multipoint-type?      multipoint-type
      |  |              |     +--ro peer* [direction peer advertisement-type]
      |  |              |        +--ro direction             downstream-upstream
      |  |              |        +--ro peer                  leafref
      |  |              |        +--ro advertisement-type    advertised-received
      |  |              |        +--ro label?                mpls:mpls-label
      |  |              |        +--ro mbb-role?             enumeration
      |  |              |        +--ro mofrr-role?           enumeration
      |  |              +--ro opaque-type-src
      |  |              |  +--ro fec-label* [root-address source-address group-a
ddress rd recur-root-address recur-rd]
      |  |              |     +--ro root-address          inet:ipv6-address
      |  |              |     +--ro source-address        inet:ip-address
      |  |              |     +--ro group-address         inet:ip-address-no-zon
e
      |  |              |     +--ro rd                    route-distinguisher
      |  |              |     +--ro recur-root-address    inet:ip-address
      |  |              |     +--ro recur-rd              route-distinguisher
      |  |              |     +--ro multipoint-type?      multipoint-type
      |  |              |     +--ro peer* [direction peer advertisement-type]
      |  |              |        +--ro direction             downstream-upstream
      |  |              |        +--ro peer                  leafref
      |  |              |        +--ro advertisement-type    advertised-received
      |  |              |        +--ro label?                mpls:mpls-label
      |  |              |        +--ro mbb-role?             enumeration
      |  |              |        +--ro mofrr-role?           enumeration
      |  |              +--ro opaque-type-bidir
      |  |                 +--ro fec-label* [root-address rp group-address rd re
cur-root-address recur-rd]
      |  |                    +--ro root-address          inet:ipv6-address
      |  |                    +--ro rp                    inet:ip-address
      |  |                    +--ro group-address         inet:ip-address-no-zon
e
      |  |                    +--ro rd                    route-distinguisher
      |  |                    +--ro recur-root-address    inet:ip-address
      |  |                    +--ro recur-rd              route-distinguisher
```

```
      |  |                    +--ro multipoint-type?      multipoint-type
      |  |                    +--ro peer* [direction peer advertisement-type]
      |  |                       +--ro direction             downstream-upstream
      |  |                       +--ro peer                  leafref
      |  |                       +--ro advertisement-type    advertised-received
      |  |                       +--ro label?                mpls:mpls-label
      |  |                       +--ro mbb-role?             enumeration
      |  |                       +--ro mofrr-role?           enumeration
      |  +--rw forwarding-nexthop {forwarding-nexthop-config}?
      |     +--rw interfaces
      |        +--rw interface* [interface]
      |           +--rw address-family* [afi]
      |              +--ro state
      |                 +--ro mldp-disable?   boolean {mldp}?
      +--rw peers
         +--rw peer* [lsr-id]
            +--ro state
               +--ro capability
               |   +--ro mldp {mldp}?
               |     +--ro p2mp
               |     |  +--ro enable?   boolean
               |     +--ro mp2mp
               |     |  +--ro enable?   boolean
               |     +--ro make-before-break
               |     |  +--ro enable?             boolean
               |     |  +--ro switchover-delay?   uint16
               |     |  +--ro timeout?            uint16
               |     +--ro hub-and-spoke {capability-mldp-hsmp}?
               |     |  +--ro enable?   boolean
               |     +--ro node-protection {capability-mldp-node-protection}?
               |        +--ro plr?           boolean
               |        +--ro merge-point
               |           +--ro enable?                            boolean
               |           +--ro targeted-session-teardown-delay?   uint16
               +--ro received-peer-state
                  +--ro capability
                     +--ro mldp {mldp}?
                        +--ro p2mp
                        |  +--ro enable?   boolean
                        +--ro mp2mp
                        |  +--ro enable?   boolean
                        +--ro make-before-break
                        |  +--ro enable?   boolean
                        +--ro hub-and-spoke
                        |  +--ro enable?   boolean
                        +--ro node-protection
                           +--ro plr?           boolean
                           +--ro merge-point?   boolean
```

Figure 15

4.3.1. Derived states

Following are main areas for which mLDP operational derived state is defined:

- o Root
- o Bindings (FEC-label)
- o Capabilities

4.3.1.1. Root state

Root address is a fundamental construct for MP FEC bindings and LSPs. The root state provides information on all the known roots in a given address-familty, and their information on the root reachability (as learnt from RIB). In case of multi-path reachability to a root, the selection of upstream path is done on per-LSP basis at the time of LSP setup. Similarly, when protection mechanisms like MBB or MoFRR are in place, the path designation as active/standby or primary/ backup is also done on per LSP basis. It is to be noted that a given root can be shared amongst multiple P2MP and/or MP2MP LSPs. Moreover, an LSP can be signaled to more than one root for RNR purposes.

The following diagram illustrates a root database on a branch/transit LSR:

```
        root 1.1.1.1:
            path1:
                RIB: GigEthernet 1/0, 12.1.0.2;
                LDP: peer 192.168.0.1:0
            path2:
                RIB: GigEthernet 2/0, 12.2.0.2;
                LDP: peer 192.168.0.3:0

        root 2.2.2.2:
            path1:
                RIB: 3.3.3.3;             (NOTE: This is a recursive path)
                LDP: peer 192.168.0.3:0   (NOTE: T-mLDP peer)

         root 9.9.9.9:
            . . . .
```

Figure 16

A root entry on a root LSR itself will be presented as follows:

```
          root 9.9.9.9:
              is-self
```

Figure 17

4.3.1.2. Bindings state

Binding state provides information on mLDP FEC-label bindings for both P2MP and MP2MP FEC types. Like LDP, the FEC-label binding derived state is presented in a FEC-centric view per address-family, and provides information on both inbound (received) and outbound (advertised) bindings. The FEC is presented as (root-address, opaque-type-data) and the direction (upstream or downstream) is picked with respect to root reachability. In case of MBB or/and MoFRR, the role of a given peer binding is also provided with respect to MBB (active or standby) or/and MoFRR (primary or backup).

This document covers following type of opaque values with their keys in the operational model of mLDP bindings:

Opaque Type	Key	RFC
Generic LSP Identifier	LSP Id	[RFC6388]
Transit IPv4 Source	Source, Group	[RFC6826]
Transit IPv6 Source	Source, Group	[RFC6826]
Transit IPv4 Bidir	RP, Group	[RFC6826]
Transit IPv6 Bidir	RP, Group	[RFC6826]
Transit VPNv4 Source	Source, Group, RD	[RFC7246]
Transit VPNv6 Source	Source, Group, RD	[RFC7246]
Transit VPNv4 Bidir	RP, Group, RD	[RFC7246]
Transit VPNv6 Bidir	RP, Group, RD	[RFC7246]
Recursive Opaque	Root	[RFC6512]
VPN-Recursive Opaque	Root, RD	[RFC6512]

Table 1: MP Opaque Types and keys

It is to be noted that there are three basic types (LSP Id, Source, and Bidir) and then there are variants (VPN, recursive, VPN-recursive) on top of these basic types.

Following captures high level tree hierarchy for mLDP bindings state:

```
+--rw mpls-ldp!
    +--rw mldp
      +--rw address-family* [afi]
         +--rw afi       address-family
         +--ro state
            +--ro ipv4 (or ipv6)
               +--ro bindings
                  +--ro opaque-type-xxx [root-address, type-specific-key]
                     +--ro root-address
                     +--ro ...
                     +--ro recur-root-address    inet:ipv4-address
                     +--ro recur-rd              route-distinguisher
                     +--ro multipoint-type?      multipoint-type
                     +--ro peer* [direction peer advertisement-type]
                        +--ro direction             downstream-upstream
                        +--ro peer                  leafref
                        +--ro advertisement-type    advertised-received
                        +--ro label?                mpls:mpls-label
                        +--ro mbb-role?             enumeration
                        +--ro mofrr-role?           enumeration
```

Figure 18

In the above tree, the type-specific-key varies with the base type as listed in earlier Table 1. For example, if the opaque type is Generic LSP Identifier, then the type-specific-key will be a uint32 value corresponding to the LSP. Please see the complete model for all other types.

Moreover, the binding tree defines only three types of sub-trees (i.e. lspid, src, and bidir) which is able to map the respective variants (vpn, recursive, and vpn-recusrive) accordingly. For example, the key for opaque-type-src is [R, S, G, rd, recur-R, recur-RD], where basic type will specify (R, S,G,-, -, -), VPN type will specify (R, S,G, rd, -, -), recursive type will specify [R, S,G, -, recur-R, -] and VPN-recursive type will specify [R, S,G, -, recur-R, recur-rd].

It is important to take note of the following:

- o The address-family ipv4/ipv4 applies to "root" address in the mLDP binding tree. The other addresses (source, group, RP etc) do not have to be of the same address family type as the root.

- o The "recur-root-address" field applies to Recursive opaque type, and (recur-root-address, recur-rd) fields applies to VPN-Recursive opaque types as defined in [RFC6512]

- o In case of a recursive FEC, the address-family of the recur-root-address could be different than the address-family of the root address of original encapsulated MP FEC

The following diagram illustrates the FEC-label binding information structure for a P2MP (Transit IPv4 Source type) LSP on a branch/ transit LSR:

```
FEC (root 2.2.2.2, S=192.168.1.1, G=224.1.1.1):
  type: p2mp
  upstream:
    advertised:
      peer 192.168.0.1:0, label 16000 (local)
  downstream:
    received:
      peer 192.168.0.2:0, label 17000 (remote)
      peer 192.168.0.3:0, label 18000  (remote)
```

Figure 19

The following diagram illustrates the FEC-label binding information structure for a similar MP2MP LSP on a branch/transit LSR:

```
FEC (root 2.2.2.2, RP=192.168.9.9, G=224.1.1.1):
  type: mp2mp
  upstream:
    advertised:
      peer 192.168.0.1:0, label 16000 (local)
    received:
      peer 192.168.0.1:0, label 17000 (remote)
  downstream:
    advertised:
      peer 192.168.0.2:0, label 16001 (local), MBB role=active
      peer 192.168.0.3:0, label 16002 (local), MBB role=standby
    received:
      peer 192.168.0.2:0, label 17001 (remote)
      peer 192.168.0.3:0, label 18001 (remote)
```

Figure 20

4.3.1.3. Capabilities state

Like LDP, mLDP capabilities state comprise two types of information - global information and per-peer information.

4.4. Notifications

mLDP notification module consists of notification related to changes in the operational state of an mLDP FEC. Following is a simplified graphical representation of the data model for mLDP notifications:

```
notifications:
   +---n mpls-mldp-fec-event
      +--ro event-type?       oper-status-event-type
      +--ro tree-type?        multipoint-type
      +--ro root?             inet:ip-address
      +--ro (lsp-key-type)?
         +--:(lsp-id-based)
         |  +--ro lsp-id?           uint16
         +--:(source-group-based)
            +--ro source-address?   inet:ip-address
            +--ro group-address?    inet:ip-address
```

Figure 21

4.5. Actions

Currently, no RPCs/actions are defined for mLDP.

5. Open Items

Following is a list of open items that are to be discussed and addressed in future revisions of this document:

- o Close on augmentation off "mpls" list in "ietf-mpls" defined in [I-D.saad-mpls-base-yang]
- o Align operational state modeling with other routing procols and [I-D.openconfig-netmod-opstate]
- o Complete the section on Protocol-centric implementations and all-vrfs
- o Specify default values for configuration parameters
- o Revisit and cut down on the scope of the document and number of features it is trying to cover
- o Split the model into a base and extended items
- o Add statistics for mLDP root LSPs and bindings
- o Extend the "Configured Leaf LSPs" for various type of opaque-types
- o Extend mLDP notifications for other types of opaque values as well
- o Close on single vs separate document for mLDP Yang

6. YANG Specification

Following are actual YANG definition for LDP and mLDP constructs defined earlier in the document.

```
 <CODE BEGINS> file "ietf-mpls-ldp@2016-07-08.yang" -->

 module ietf-mpls-ldp {
   namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp";
   // replace with IANA namespace when assigned
   prefix ldp;

   import ietf-inet-types {
```

```
    prefix "inet";
  }

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

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-ip {
    prefix "ip";
  }

  import ietf-routing {
    prefix "rt";
  }

  import ietf-mpls {
    prefix "mpls";
  }

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

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

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

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

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

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

     Editor:   Xufeng Liu
               <mailto:xliu@kuatrotech.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) and Multipoint LDP (mLDP).";

   revision 2016-07-08 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: YANG Data Model for MPLS LDP and mLDP.";
   }

   /*
    * Features
    */

   feature admin-down-config {
     description
       "This feature indicates that the system allows to configure
        administrative down on a VRF instance and a peer.";
   }

   feature all-af-policy-config {
     description
       "This feature indicates that the system allows to configure
        policies that are applied to all address families.";
   }

   feature capability-end-of-lib {
     description
       "This feature indicates that the system allows to configure
        LDP end-of-lib capability.";
   }

   feature capability-mldp-hsmp {
     description
       "This feature indicates that the system allows to configure
        mLDP hub-and-spoke-multipoint capability.";
   }
```

```
   feature capability-mldp-node-protection {
     description
       "This feature indicates that the system allows to configure
        mLDP node-protection 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 global-session-authentication {
     description
       "This feature indicates that the system allows to configure
        authentication at global level.";
   }

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

   feature mldp {
     description
       "This feature indicates that the system supports Multicast
        LDP (mLDP).";
   }

   feature mldp-mofrr {
     description
       "This feature indicates that the system supports mLDP
        Multicast only FRR (MoFRR).";
   }
```

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

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

   /*
    * Typedefs
    */
   typedef ldp-address-family {
     type identityref {
```

```
        base rt:address-family;
      }
      description
        "LDP address family type.";
    }

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

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

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

    typedef label-adv-mode {
      type enumeration {
```

```
        enum downstream-unsolicited {
          description "Downstream Unsolicited.";
        }
        enum downstream-on-demand {
          description "Downstream on Demand.";
        }
      }
      description
        "Label Advertisement Mode.";
    }

    typedef mpls-interface-ref {
      type leafref {
        path "/rt:routing/mpls:mpls/mpls:interface/mpls:name";
      }
      description
        "This type is used by data models that need to reference
         mpls interfaces.";
    }

    typedef multipoint-type {
      type enumeration {
        enum p2mp {
          description "Point to multipoint.";
        }
        enum mp2mp {
          description "Multipoint to multipoint.";
        }
      }
      description
        "p2mp or mp2mp.";
    }

    typedef neighbor-list-ref {
      type string;
      description
        "A type for a reference to a neighbor list.";
    }

    typedef peer-list-ref {
      type string;
      description
        "A type for a reference to a peer list.";
    }

    typedef prefix-list-ref {
      type string;
      description
```

```
      "A type for a reference to a prefix list.";
  }

  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.";
  }

  typedef route-distinguisher {
    type string {
    }
    description
      "Type definition for route distinguisher.";
    reference
      "RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs).";
  }

  /*
   * Identities
   */
  identity adjacency-flag-base {
    description "Base type for adjacency flags.";
  }

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

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

  /*
```

```
    * Groupings
    */

   grouping adjacency-state-attributes {
     description
       "Adjacency state attributes.";

     leaf-list flag {
       type identityref {
         base "adjacency-flag-base";
       }
       description "Adjacency flags.";
     }
     container hello-holdtime {
       description "Hello holdtime state.";
       leaf adjacent {
         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.";
       }
     }

     leaf next-hello {
       type uint16;
       units seconds;
       description "Time to send the next hello message.";
     }

     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 received.";
      }
    } // statistics
  } // adjacency-state-attributes

  grouping basic-discovery-timers {
    description
      "Basic discovery timer attributes.";
    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";
    }
    leaf hello-interval {
      type uint16 {
        range 5..1200;
      }
      units seconds;
      description
        "The interval between consecutive LDP link Hello messages
         used in basic LDP discovery";
    }
  } // basic-discovery-timers

  grouping binding-address-state-attributes {
    description
      "Address binding attributes";
    leaf advertisement-type {
      type advertised-received;
```

```
        description
          "Received or advertised.";
      }
      leaf peer {
        type leafref {
          path "../../../../../../../peers/peer/lsr-id";
        }
        must "../advertisement-type = 'received'" {
          description
            "Applicable for received address.";
        }
        description
          "LDP peer from which this address is received.";
      } // peer
    } // binding-address-state-attributes

    grouping binding-label-state-attributes {
      description
        "Label binding attributes";
      list peer {
        key "peer advertisement-type";
        description
          "List of advertised and received peers.";
        leaf peer {
          type leafref {
            path "../../../../../../../../peers/peer/lsr-id";
          }
          description
            "LDP peer from which this binding is received,
             or to which this binding is advertised.";
        }
        leaf advertisement-type {
          type advertised-received;
          description
            "Received or advertised.";
        }
        leaf label {
          type mpls:mpls-label;
          description
            "Advertised (outbound) or received (inbound)
             label.";
        }
        leaf used-in-forwarding {
          type boolean;
          description
            "'true' if the lable is used in forwarding.";
        }
      } // peer
```

```
   } // binding-label-state-attributes

   grouping extended-discovery-policy-attributes {
     description
       "LDP policy to control the acceptance of extended neighbor
        discovery hello messages.";
     container hello-accept {
       if-feature policy-extended-discovery-config;
       description
         "Extended discovery acceptance policies.";

       leaf enable {
         type boolean;
         description
           "'true' to accept; 'false' to deny.";
       }
       leaf neighbor-list {
         type neighbor-list-ref;
         description

           "The name of a peer ACL.";
       }
     } // hello-accept
   } // extended-discovery-policy-attributes

   grouping extended-discovery-timers {
     description
       "Extended discovery timer attributes.";
     leaf hello-holdtime {
       type uint16 {
         range 15..3600;
       }
       units seconds;
       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;
       description
         "The interval between consecutive LDP targeted Hello
          messages used in extended LDP discovery.";
     }
```

```
   } // extended-discovery-timers

   grouping global-attributes {
     description "Configuration attributes at global level.";

     uses instance-attributes;
   } // global-attributes

   grouping graceful-restart-attributes {
     description
       "Graceful restart configuration attributes.";
     container graceful-restart {
       description
         "Attributes for graceful restart.";
       leaf enable {
         type boolean;
         description
           "Enable or disable graceful restart.";
       }
       leaf helper-enable {
         if-feature graceful-restart-helper-mode;
         type boolean;
         description
           "Enable or disable graceful restart helper mode.";
       }
       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.";
       }
       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.";
       }
       leaf forwarding-holdtime {
         type uint16 {
```

```
          range 30..3600;
        }
        units seconds;
        description
          "Specifies the time interval, in seconds, before the
           termination of the recovery phase.";
      }
    } // graceful-restart
  } // graceful-restart-attributes

  grouping graceful-restart-attributes-per-peer {
    description
      "Per peer graceful restart configuration attributes.";
    container graceful-restart {
      description
        "Attributes for graceful restart.";
      leaf enable {
        type boolean;
        description
          "Enable or disable graceful restart.";
      }
      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.";
      }
      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.";
      }
    } // graceful-restart
  } // graceful-restart-attributes-per-peer

  grouping instance-attributes {
    description "Configuration attributes at instance level.";
```

```
     container capability {
       description "Configure capability.";
       container end-of-lib {
         if-feature capability-end-of-lib;
         description
           "Configure end-of-lib capability.";
         leaf enable {
           type boolean;
           description
             "Enable end-of-lib capability.";
         }
       }
       container typed-wildcard-fec {
         if-feature capability-typed-wildcard-fec;
         description
           "Configure typed-wildcard-fec capability.";
         leaf enable {
           type boolean;
           description
             "Enable typed-wildcard-fec capability.";
         }
       }
       container upstream-label-assignment {
         if-feature capability-upstream-label-assignment;
         description
           "Configure upstream label assignment capability.";
         leaf enable {
           type boolean;
           description
             "Enable upstream label assignment.";
         }
       }
       container mldp {
         if-feature mldp;

         description
           "Multipoint capabilities.";
         uses mldp-capabilities;
       }
     } // capability

     uses graceful-restart-attributes;

     leaf igp-synchronization-delay {
       type uint16 {
         range 3..60;
       }
       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.";
      }
      leaf lsr-id {
        type yang:dotted-quad;
        description "Router ID.";
      }
    } // instance-attributes

    grouping ldp-adjacency-ref {
      description
        "An absolute reference to an LDP adjacency.";
      choice hello-adjacency-type {
        description
          "Interface or targeted adjacency.";
        case targeted {
          container targeted {
            description "Targeted adjacency.";
            leaf target-address {
              type inet:ip-address;
              description
                "The target address.";
            }
          } // targeted
        }
        case link {
          container link {
            description "Link adjacency.";
            leaf next-hop-interface {
              type mpls-interface-ref;
              description
                "Interface connecting to next-hop.";
            }
            leaf next-hop-address {
              type inet:ip-address;
              must "../next-hop-interface" {
                description
                  "Applicable when interface is specified.";

              }
              description
                "IP address of next-hop.";
            }
          } // link
        }
```

```
      }
    } // ldp-adjacency-ref

    grouping ldp-fec-event {
      description
        "A LDP FEC event.";
      leaf prefix {
        type inet:ip-prefix;
        description
          "FEC.";
      }
    } // ldp-fec-event

    grouping ldp-peer-ref {
      description
        "An absolute reference to an LDP peer.";
      leaf peer-ref {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/mpls-ldp/"
            + "peers/peer/lsr-id";
        }
        description
          "Reference to an LDP peer.";
      }
    } // ldp-peer-ref

    grouping mldp-capabilities {
      description
        "mLDP capabilities.";
      container p2mp {
        description
          "Configure point-to-multipoint capability.";
        leaf enable {
          type boolean;
          description
            "Enable point-to-multipoint.";
        }
      }
      container mp2mp {
        description
          "Configure multipoint-to-multipoint capability.";
        leaf enable {
          type boolean;
          description
            "Enable multipoint-to-multipoint.";
        }
      }
      container make-before-break {
```

```
        description
          "Configure make-before-break capability.";
        leaf enable {
          type boolean;
          description
            "Enable make-before-break.";
        }
        leaf switchover-delay {
          type uint16;
          units seconds;
          description
            "Switchover delay in seconds.";
        }
        leaf timeout {
          type uint16;
          units seconds;
          description
            "Timeout in seconds.";
        }
      }
      container hub-and-spoke {
        if-feature capability-mldp-hsmp;
        description
          "Configure hub-and-spoke-multipoint capability.";
        reference
          "RFC7140: LDP Extensions for Hub and Spoke Multipoint
           Label Switched Path";
        leaf enable {
          type boolean;
          description
            "Enable hub-and-spoke-multipoint.";
        }
      }
      container node-protection {
        if-feature capability-mldp-node-protection;
        description
          "Configure node-protection capability.";
        reference
          "RFC7715: mLDP Node Protection.";
        leaf plr {
          type boolean;
          description
            "Point of Local Repair capable for MP LSP node
             protection.";
        }
        container merge-point {
          description
            "Merge Point capable for MP LSP node protection.";
```

```
        leaf enable {
          type boolean;
          description
            "Enable merge point capability.";
        }
        leaf targeted-session-teardown-delay {
          type uint16;
          units seconds;
          description
            "Targeted session teardown delay.";
        }
      } // merge-point
    }
  } // mldp-capabilities

  grouping  mldp-configured-lsp-roots {
    description
      "mLDP roots containers.";

    container roots-ipv4 {

      when "../../../af = 'ipv4'" {
        description
          "Only for IPv4.";
      }
      description
        "Configured IPv4 multicast LSPs.";
      list root {
        key "root-address";
        description
          "List of roots for configured multicast LSPs.";

        leaf root-address {
          type inet:ipv4-address;
          description
            "Root address.";
        }

        list lsp {
          must "(lsp-id = 0 and source-address != '0.0.0.0' and "
            + "group-address != '0.0.0.0') or "
            + "(lsp-id != 0 and source-address = '0.0.0.0' and "
            + "group-address = '0.0.0.0')" {
            description
              "A LSP can be identified by either <lsp-id> or
               <source-address, group-address>.";
          }
          key "lsp-id source-address group-address";
```

```
          description
            "List of LSPs.";
          leaf lsp-id {
            type uint16;
            description "ID to identify the LSP.";
          }
          leaf source-address {
            type inet:ipv4-address;
            description
              "Source address.";
          }
          leaf group-address {
            type inet:ipv4-address-no-zone;
            description
              "Group address.";
          }
        } // list lsp
      } // list root
    } // roots-ipv4

    container roots-ipv6 {

      when "../../../af = 'ipv6'" {
        description
          "Only for IPv6.";
      }
      description
        "Configured IPv6 multicast LSPs.";

      list root {
        key "root-address";
        description
          "List of roots for configured multicast LSPs.";

        leaf root-address {
          type inet:ipv6-address;
          description
            "Root address.";
        }

        list lsp {
          must "(lsp-id = 0 and source-address != '::' and "
            + "group-address != '::') or "
            + "(lsp-id != 0 and source-address = '::' and "
            + "group-address = '::')" {
            description
              "A LSP can be identified by either <lsp-id> or
               <source-address, group-address>.";
```

```
          }
          key "lsp-id source-address group-address";
          description
            "List of LSPs.";
          leaf lsp-id {
            type uint16;
            description "ID to identify the LSP.";
          }
          leaf source-address {
            type inet:ipv6-address;
            description
              "Source address.";
          }
          leaf group-address {
            type inet:ipv6-address-no-zone;
            description
              "Group address.";
          }
        } // list lsp
      } // list root
    } // roots-ipv6
  } // mldp-configured-lsp-roots

  grouping mldp-fec-event {
    description
      "A mLDP FEC event.";
    leaf tree-type {
      type multipoint-type;
      description
        "p2mp or mp2mp.";
    }
    leaf root {
      type inet:ip-address;
      description
        "Root address.";
    }
    choice lsp-key-type {
      description
        "LSP ID based or source-group based .";
      case lsp-id-based {
        leaf lsp-id {
          type uint16;
          description
            "ID to identify the LSP.";
        }
      }
      case source-group-based {
        leaf source-address {
```

```
          type inet:ip-address;
          description

            "LSP source address.";
        }
        leaf group-address {
          type inet:ip-address;
          description
            "Multicast group address.";
        }
      } // case source-group-based
    }
  } // mldp-fec-event

  grouping  mldp-binding-label-state-attributes {
    description
      "mLDP label binding attributes.";

    leaf multipoint-type {
      type multipoint-type;
      description
        "The type of mutipoint, p2mp or mp2mp.";
    }
    list peer {
      key "direction peer advertisement-type";
      description
        "List of advertised and received peers.";
      leaf direction {
        type downstream-upstream;
        description
          "Downstream or upstream.";
      }
      leaf peer {
        type leafref {
          path
            "../../../../../../../../../../peers/peer/lsr-id";
        }
        description
          "LDP peer from which this binding is received,
           or to which this binding is advertised.";
      }
      leaf advertisement-type {
        type advertised-received;
        description
          "Advertised or received.";
      }
      leaf label {
        type mpls:mpls-label;
```

```
          description
            "Advertised (outbound) or received (inbound) label.";
        }
        leaf mbb-role {
          when "../direction = 'upstream'" {
            description
              "For upstream.";
          }
          type enumeration {
            enum none {
              description "MBB is not enabled.";
            }
            enum active {
              description "This LSP is active.";
            }
            enum inactive {
              description "This LSP is inactive.";
            }
          }
          description
            "The MBB status of this LSP.";
        }
        leaf mofrr-role {
          when "../direction = 'upstream'" {
            description
              "For upstream.";
          }
          type enumeration {
            enum none {
              description "MOFRR is not enabled.";
            }
            enum primary {
              description "This LSP is primary.";
            }
            enum backup {
              description "This LSP is backup.";
            }
          }
          description
            "The MOFRR status of this LSP.";
        }
      } // peer
    } // mldp-binding-label-state-attributes

    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 outgoing label
                 advertisements.";
          }
        }
        container accept {
          description
            "Label advertisement acceptance policies.";
          leaf prefix-list {
            type prefix-list-ref;
            description
              "Applies the prefix list to incoming label
               advertisements.";
          }
        } // accept
      } // label-policy
    } // peer-af-policy-container

    grouping peer-attributes {
      description "Peer configuration attributes.";

      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.";
      }
      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.";
```

```
      }
    } // peer-attributes

    grouping peer-authentication {
      description
        "Peer authentication attributes.";
      leaf session-authentication-md5-password {
        type string {
          length "1..80";
        }
        description
          "Assigns an encrypted MD5 password to an LDP
           peer";
      } // md5-password
    } // peer-authentication

    grouping peer-state-derived {
      description "Peer derived state attributes.";

      container label-advertisement-mode {
        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;
        description "Time to send the next KeepAlive message.";
      }

      leaf peer-ldp-id {
        type yang:dotted-quad;
        description "Peer LDP ID.";
      }
```

```
     container received-peer-state {
       description "Peer features.";

       uses graceful-restart-attributes-per-peer;

       container capability {
         description "Configure capability.";
         container end-of-lib {
           description
             "Configure end-of-lib capability.";
           leaf enable {
             type boolean;
             description
               "Enable end-of-lib capability.";
           }
         }
         container typed-wildcard-fec {
           description
             "Configure typed-wildcard-fec capability.";
           leaf enable {
             type boolean;
             description
               "Enable typed-wildcard-fec capability.";
           }
         }
         container upstream-label-assignment {
           description
             "Configure upstream label assignment capability.";
           leaf enable {
             type boolean;
             description
               "Enable upstream label assignment.";
           }
         }
         container mldp {
           if-feature mldp;
           description
             "Multipoint capabilities.";

           container p2mp {
             description
               "Configure point-to-multipoint capability.";
             leaf enable {
               type boolean;
               description
                 "Enable point-to-multipoint.";
             }
           }
```

```
          container mp2mp {
            description
              "Configure multipoint-to-multipoint capability.";
            leaf enable {
              type boolean;
              description
                "Enable multipoint-to-multipoint.";
            }
          }
          container make-before-break {
            description
              "Configure make-before-break capability.";
            leaf enable {
              type boolean;
              description
                "Enable make-before-break.";
            }
          }
          container hub-and-spoke {
            description
              "Configure hub-and-spoke-multipoint capability.";
            reference
              "RFC7140: LDP Extensions for Hub and Spoke Multipoint
               Label Switched Path";
            leaf enable {
              type boolean;
              description
                "Enable hub-and-spoke-multipoint.";
            }
          }
          container node-protection {
            description
              "Configure node-protection capability.";
            reference
              "RFC7715: mLDP Node Protection.";
            leaf plr {
              type boolean;
              description
                "Point of Local Repair capable for MP LSP node
                 protection.";
            }
            leaf merge-point {
              type boolean;
              description
                "Merge Point capable for MP LSP node protection.";
            } // merge-point
          } // node-protection
        } // mldp
```

```
        } // capability
      } // received-peer-state

      container session-holdtime {
        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.";
          }
        }
        description
          "Representing the operational status.";
      }

      container tcp-connection {
        description "TCP connection state.";
        leaf local-address {
          type inet:ip-address;
```

```
        description "Local address.";
      }
      leaf local-port {
        type inet:port-number;
        description "Local port.";
      }
      leaf remote-address {
        type inet:ip-address;
        description "Remote address.";
      }
      leaf remote-port {
        type inet:port-number;
        description "Remote port.";
      }
    } // tcp-connection

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

    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.";
      }

      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 policy-container {
    description
      "LDP policy attributes.";
    container label-policy {
      description
        "Label policy attributes.";
      container independent-mode {
        description
          "Independent label policy attributes.";
        container assign {

          if-feature policy-label-assignment-config;
          description
            "Label assignment policies";
          choice prefix-option {
            description
              "Use either prefix-list or host-routes-only.";
            case prefix-list {
              leaf prefix-list {
                type prefix-list-ref;
                description
                  "Assign labels according to certain prefixes.";
              }
            }
            case host-routes-only {
              leaf host-routes-only {
                type boolean;
                description
                  "'true' to apply host routes only.";
              }
            }
          } // prefix-option
```

```
          }
          container advertise {
            description
              "Label advertising policies.";
            container explicit-null {
              description
                "Enables an egress router to advertise an
                 explicit null label (value 0) in place of an
                 implicit null label (value 3) to the
                 penultimate hop router.";
              leaf enable {
                type boolean;
                description
                  "'true' to enable explicit null.";
              }
              leaf prefix-list {
                type prefix-list-ref;
                description
                  "Prefix list name. Applies the filters in the
                   specified prefix list to label
                   advertisements.
                   If the prefix list is not specified, explicit
                   null label advertisement is enabled for all
                   directly connected prefixes.";
              }
            }
            leaf prefix-list {
              type prefix-list-ref;
              description
                "Applies the prefix list to outgoing label
                 advertisements.";

            }
          }
          container accept {
            description
              "Label advertisement acceptance policies.";
            leaf prefix-list {
              type prefix-list-ref;
              description
                "Applies the prefix list to incoming label
                 advertisements.";
            }
          }
        } // independent-mode
        container ordered-mode {
          if-feature policy-ordered-label-config;
          description
```

```
          "Ordered label policy attributes.";
        container egress-lsr {
          description
            "Egress LSR label assignment policies";
          leaf prefix-list {
            type prefix-list-ref;
            description
              "Assign labels according to certain prefixes.";
          }
        }
        container advertise {
          description
            "Label advertising policies.";
          leaf prefix-list {
            type prefix-list-ref;
            description
              "Applies the prefix list to outgoing label
               advertisements.";
          }
        }
        container accept {
          description
            "Label advertisement acceptance policies.";
          leaf prefix-list {
            type prefix-list-ref;
            description
              "Applies the prefix list to incoming label
               advertisements.";
          }
        }
      } // ordered-mode
    } // label-policy
  } // policy-container

  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 messages sent or received.";
     }
   } // statistics-peer-received-sent

   /*
    * Configuration data nodes
    */

   augment "/rt:routing/rt:control-plane-protocols" {
     description "LDP augmentation.";

     container mpls-ldp {
       presence "Container for LDP protocol.";
       description
         "Container for LDP protocol.";

       container global {
         description
           "Global attributes for LDP.";
         container config {
           description
             "Configuration data.";
           uses global-attributes;
         }
         container state {
           config false;
           description
             "Operational state data.";
           uses global-attributes;
         }

         container mldp {
           if-feature mldp;
           description
             "mLDP attributes at per instance level. Defining
              attributes here does not enable any MP capabilities.
              MP capabilities need to be explicitly enabled under
              container capability.";

           container config {
             description
               "Configuration data.";
             leaf enable {
               type boolean;
               description
                 "Enable mLDP.";
             }
           }
```

```
           container state {
             config false;
             description

               "Operational state data.";
             leaf enable {
               type boolean;
               description
                 "Enable mLDP.";
             }
           }

           list address-family {
             key "afi";
             description
               "Per-af params.";
             leaf afi {
               type ldp-address-family;
               description
                 "Address family type value.";
             }

             container config {
               description
                 "Configuration data.";
               container multicast-only-frr {
                 if-feature mldp-mofrr;
                 description
                   "Multicast only FRR (MoFRR) policy.";
                 leaf prefix-list {
                   type prefix-list-ref;
                   description
                     "Enables MoFRR for the specified access list.";
                 }
               } // multicast-only-frr
               container recursive-fec {
                 description
                   "Recursive FEC policy.";
                 leaf prefix-list {
                   type prefix-list-ref;
                   description
                     "Enables recursive FEC for the specified access
                      list.";
                 }
               } // recursive-for
             }
             container state {
               config false;
```

```
                description
                  "Operational state data.";
                container multicast-only-frr {
                  if-feature mldp-mofrr;

                  description
                    "Multicast only FRR (MoFRR) policy.";
                  leaf prefix-list {
                    type prefix-list-ref;
                    description
                      "Enables MoFRR for the specified access list.";
                  }
                } // multicast-only-frr
                container recursive-fec {
                  description
                    "Recursive FEC policy.";
                  leaf prefix-list {
                    type prefix-list-ref;
                    description
                      "Enables recursive FEC for the specified access
                       list.";
                  }
                } // recursive-fec

                container ipv4 {
                  when "../../afi = 'ipv4'" {
                    description
                      "Only for IPv4.";
                  }
                  description
                    "IPv4 state information.";
                  container roots {
                    description
                      "IPv4 multicast LSP roots.";
                    list root {
                      key "root-address";
                      description
                        "List of roots for configured multicast LSPs.";

                      leaf root-address {
                        type inet:ipv4-address;
                        description
                          "Root address.";
                      }

                      leaf is-self {
                        type boolean;
                        description
```

```
                        "This is the root.";
                    }

                    list reachability {
                      key "address interface";
                      description
                        "A next hop for reachability to root,
                         as a RIB view.";
                      leaf address {
                        type inet:ipv4-address;
                        description
                          "The next hop address to reach root.";
                      }
                      leaf interface {
                        type mpls-interface-ref;
                        description
                          "Interface connecting to next-hop.";
                      }
                      leaf peer {
                        type leafref {
                          path
                            "../../../../../../../../../peers/peer/"
                            + "lsr-id";
                        }
                        description
                          "LDP peer from which this next hop can be
                           reached.";
                      }
                    }
                  } // list root
                } // roots
                container bindings {
                  description
                    "mLDP FEC to label bindings.";
                  container opaque-type-lspid {
                    description
                      "The type of opaque value element is
                       the generic LSP identifier";
                    reference
                      "RFC6388: Label Distribution Protocol
                       Extensions for Point-to-Multipoint and
                       Multipoint-to-Multipoint Label Switched
                       Paths.";
                    list fec-label {
                      key
                        "root-address lsp-id "
                        + "recur-root-address recur-rd";
                      description
```

```
                        "List of FEC to label bindings.";
                      leaf root-address {
                        type inet:ipv4-address;
                        description
                          "Root address.";
                      }
                      leaf lsp-id {
                        type uint32;
                        description "ID to identify the LSP.";
                      }
                      leaf recur-root-address {
                        type inet:ip-address;
                        description
                          "Recursive root address.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      leaf recur-rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher in the VPN-Recursive
                           Opaque Value.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      uses mldp-binding-label-state-attributes;
                    } // fec-label
                  } // opaque-type-lspid

                  container opaque-type-src {
                    description
                      "The type of opaque value element is
                       the transit source TLV";
                    reference
                      "RFC6826: Multipoint LDP In-Band Signaling for
                       Point-to-Multipoint and
                       Multipoint-to-Multipoint Label Switched
                       Paths.";
                    list fec-label {
                      key
                        "root-address source-address group-address "
                        + "rd recur-root-address recur-rd";
                      description
                        "List of FEC to label bindings.";
                      leaf root-address {
                        type inet:ipv4-address;
```

```
                        description
                          "Root address.";
                      }
                      leaf source-address {
                        type inet:ip-address;
                        description
                          "Source address.";
                      }
                      leaf group-address {
                        type inet:ip-address-no-zone;
                        description
                          "Group address.";
                      }
                      leaf rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher.";
                        reference
                          "RFC7246: Multipoint Label Distribution
                           Protocol In-Band Signaling in a Virtual
                           Routing and Forwarding (VRF) Table
                           Context.";
                      }
                      leaf recur-root-address {
                        type inet:ip-address;
                        description
                          "Recursive root address.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      leaf recur-rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher in the VPN-Recursive
                           Opaque Value.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      uses mldp-binding-label-state-attributes;
                    } // fec-label
                  } // opaque-type-src

                  container opaque-type-bidir {
                    description
                      "The type of opaque value element is
                       the generic LSP identifier";
```

```
                      reference
                        "RFC6826: Multipoint LDP In-Band Signaling for
                         Point-to-Multipoint and
                         Multipoint-to-Multipoint Label Switched
                         Paths.";
                      list fec-label {
                        key
                          "root-address rp group-address "
                          + "rd recur-root-address recur-rd";
                        description
                          "List of FEC to label bindings.";
                        leaf root-address {
                          type inet:ipv4-address;
                          description
                            "Root address.";
                        }
                        leaf rp {
                          type inet:ip-address;
                          description
                            "RP address.";
                        }
                        leaf group-address {
                          type inet:ip-address-no-zone;
                          description
                            "Group address.";
                        }
                        leaf rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher.";
                          reference
                            "RFC7246: Multipoint Label Distribution
                             Protocol In-Band Signaling in a Virtual
                             Routing and Forwarding (VRF) Table
                             Context.";
                        }
                        leaf recur-root-address {
                          type inet:ip-address;
                          description
                            "Recursive root address.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        leaf recur-rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher in the VPN-Recursive
```

```
                             Opaque Value.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        uses mldp-binding-label-state-attributes;
                      } // fec-label
                    } // opaque-type-bidir
                  } // bindings
                } // ipv4

                container ipv6 {
                  when "../../afi = 'ipv6'" {
                    description
                      "Only for IPv6.";
                  }
                  description
                    "IPv6 state information.";
                  container roots {
                    description
                      "IPv6 multicast LSP roots.";
                    list root {
                      key "root-address";
                      description
                        "List of roots for configured multicast LSPs.";

                      leaf root-address {
                        type inet:ipv6-address;
                        description
                          "Root address.";
                      }

                      leaf is-self {
                        type boolean;
                        description
                          "This is the root.";
                      }

                      list reachability {
                        key "address interface";
                        description
                          "A next hop for reachability to root,
                           as a RIB view.";
                        leaf address {
                          type inet:ipv6-address;
                          description
                            "The next hop address to reach root.";
                        }
```

```
                        leaf interface {
                          type mpls-interface-ref;
                          description
                            "Interface connecting to next-hop.";
                        }
                        leaf peer {
                          type leafref {
                            path
                              "../../../../../../../../../peers/peer/"
                              + "lsr-id";
                          }
                          description
                            "LDP peer from which this next hop can be
                             reached.";
                        }
                      }
                    } // list root
                  } // roots
                  container bindings {
                    description
                      "mLDP FEC to label bindings.";
                    container opaque-type-lspid {
                      description
                        "The type of opaque value element is
                         the generic LSP identifier";
                      reference
                        "RFC6388: Label Distribution Protocol
                         Extensions for Point-to-Multipoint and
                         Multipoint-to-Multipoint Label Switched
                         Paths.";
                      list fec-label {
                        key
                          "root-address lsp-id "
                          + "recur-root-address recur-rd";
                        description
                          "List of FEC to label bindings.";
                        leaf root-address {
                          type inet:ipv6-address;
                          description
                            "Root address.";
                        }
                        leaf lsp-id {
                          type uint32;
                          description "ID to identify the LSP.";
                        }
                        leaf recur-root-address {
                          type inet:ip-address;
                          description
```

```
                    "Recursive root address.";
                  reference
                    "RFC6512: Using Multipoint LDP When the
                     Backbone Has No Route to the Root";
                }
                leaf recur-rd {
                  type route-distinguisher;
                  description
                    "Route Distinguisher in the VPN-Recursive
                     Opaque Value.";
                  reference
                    "RFC6512: Using Multipoint LDP When the
                     Backbone Has No Route to the Root";
                }
                uses mldp-binding-label-state-attributes;
              } // fec-label
            } // opaque-type-lspid

            container opaque-type-src {
              description
                "The type of opaque value element is
                 the transit Source TLV";
              reference
                "RFC6826: Multipoint LDP In-Band Signaling for
                 Point-to-Multipoint and
                 Multipoint-to-Multipoint Label Switched
                 Paths.";
              list fec-label {
                key
                  "root-address source-address group-address "
                  + "rd recur-root-address recur-rd";
                description
                  "List of FEC to label bindings.";
                leaf root-address {
                  type inet:ipv6-address;
                  description
                    "Root address.";
                }
                leaf source-address {
                  type inet:ip-address;
                  description
                    "Source address.";
                }
                leaf group-address {
                  type inet:ip-address-no-zone;
                  description
                    "Group address.";
                }
```

```
                        leaf rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher.";
                          reference
                            "RFC7246: Multipoint Label Distribution
                             Protocol In-Band Signaling in a Virtual
                             Routing and Forwarding (VRF) Table
                             Context.";
                        }
                        leaf recur-root-address {
                          type inet:ip-address;
                          description
                            "Recursive root address.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        leaf recur-rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher in the VPN-Recursive
                             Opaque Value.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        uses mldp-binding-label-state-attributes;
                      } // fec-label
                    } // opaque-type-src

                    container opaque-type-bidir {
                      description
                        "The type of opaque value element is
                         the generic LSP identifier";
                      reference
                        "RFC6826: Multipoint LDP In-Band Signaling for
                         Point-to-Multipoint and
                         Multipoint-to-Multipoint Label Switched
                         Paths.";
                      list fec-label {
                        key
                          "root-address rp group-address "
                          + "rd recur-root-address recur-rd";
                        description
                          "List of FEC to label bindings.";
                        leaf root-address {
                          type inet:ipv6-address;
```

```
                        description
                          "Root address.";
                      }
                      leaf rp {
                        type inet:ip-address;
                        description
                          "RP address.";
                      }
                      leaf group-address {
                        type inet:ip-address-no-zone;
                        description
                          "Group address.";
                      }
                      leaf rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher.";
                        reference
                          "RFC7246: Multipoint Label Distribution
                           Protocol In-Band Signaling in a Virtual
                           Routing and Forwarding (VRF) Table
                           Context.";
                      }
                      leaf recur-root-address {
                        type inet:ip-address;
                        description
                          "Recursive root address.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      leaf recur-rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher in the VPN-Recursive
                           Opaque Value.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      uses mldp-binding-label-state-attributes;
                    } // fec-label
                  } // opaque-type-bidir
                } // bindings
              } // ipv6
            } // state

            container configured-leaf-lsps {
```

```
               description
                 "Configured multicast LSPs.";

               container p2mp {
                 description
                   "Configured point-to-multipoint LSPs.";
                 uses mldp-configured-lsp-roots;
               }
               container mp2mp {
                 description
                   "Configured multipoint-to-multipoint LSPs.";
                 uses mldp-configured-lsp-roots;
               }
             } // configured-leaf-lsps
           } // list address-family
         } // mldp

         list address-family {
           key "afi";
           description
             "Per-vrf per-af params.";
           leaf afi {
             type ldp-address-family;
             description
               "Address family type value.";
           }

           container config {
             description
               "Configuration data.";
             leaf enable {
               type boolean;
               description
                 "'true' to enable the address family.";
             }
             uses policy-container;

             container ipv4 {
               when "../../afi = 'ipv4'" {
                 description
                   "Only for IPv4.";
               }
               description
                 "IPv4 address family.";
               leaf transport-address {
                 type inet:ipv4-address;
                 description
                   "The transport address advertised in LDP Hello
```

```
                 messages.";
            }
          } // ipv4
          container ipv6 {
            when "../../afi = 'ipv6'" {
              description
                "Only for IPv6.";
            }
            description
              "IPv6 address family.";
            leaf transport-address {
              type inet:ipv6-address;
              description
                "The transport address advertised in LDP Hello
                 messages.";
            }
          } // ipv6
        }
        container state {
          config false;
          description
            "Operational state data.";
          leaf enable {
            type boolean;
            description
              "'true' to enable the address family.";
          }

          uses policy-container;

          container ipv4 {
            when "../../afi = 'ipv4'" {
              description
                "Only for IPv4.";
            }
            description
              "IPv4 address family.";
            leaf transport-address {
              type inet:ipv4-address;
              description
                "The transport address advertised in LDP Hello
                 messages.";
            }

            container bindings {
              description
                "LDP address and label binding information.";
              list address {
```

```
                  key "address";
                  description
                    "List of address bindings.";
                  leaf address {
                    type inet:ipv4-address;
                    description
                      "Binding address.";
                  }
                  uses binding-address-state-attributes;
                } // binding-address

                list fec-label {
                  key "fec";
                  description
                    "List of label bindings.";
                  leaf fec {
                    type inet:ipv4-prefix;
                    description
                      "Prefix FEC.";
                  }
                  uses binding-label-state-attributes;
                } // fec-label
              } // binding
            } // ipv4
            container ipv6 {
              when "../../afi = 'ipv6'" {
                description
                  "Only for IPv6.";
              }
              description
                "IPv6 address family.";
              leaf transport-address {
                type inet:ipv6-address;
                description
                  "The transport address advertised in LDP Hello
                   messages.";
              }

              container binding {
                description
                  "LDP address and label binding information.";
                list address {
                  key "address";
                  description
                    "List of address bindings.";
                  leaf address {
                    type inet:ipv6-address;
                    description
```

```
                      "Binding address.";
                  }
                  uses binding-address-state-attributes;
                } // binding-address

                list fec-label {
                  key "fec";
                  description
                    "List of label bindings.";
                  leaf fec {
                    type inet:ipv6-prefix;
                    description
                      "Prefix FEC.";
                  }
                  uses binding-label-state-attributes;
                } // fec-label
              } // binding
            } // ipv6
          } // state
        } // address-family

        container discovery {
          description
            "Neibgbor discovery configuration.";

          container interfaces {
            description
              "A list of interfaces for basic descovery.";
            container config {
              description
                "Configuration data.";
              uses basic-discovery-timers;
            }
            container state {
              config false;
              description

                "Operational state data.";
              uses basic-discovery-timers;
            }

            list interface {
              key "interface";
              description
                "List of LDP interfaces.";
              leaf interface {
                type mpls-interface-ref;
                description
```

```
                    "Interface.";
                }
                container config {
                  description
                    "Configuration data.";
                  uses basic-discovery-timers {
                    if-feature per-interface-timer-config;
                  }
                  leaf igp-synchronization-delay {
                    if-feature per-interface-timer-config;
                    type uint16 {
                      range 3..60;
                    }
                    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.";
                  }
                }
                container state {
                  config false;
                  description
                    "Operational state data.";
                  uses basic-discovery-timers {
                    if-feature per-interface-timer-config;
                  }
                  leaf igp-synchronization-delay {
                    if-feature per-interface-timer-config;
                    type uint16 {
                      range 3..60;
                    }
                    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.";
                  }
                  leaf next-hello {
                    type uint16;
                    units seconds;
                    description "Time to send the next hello message.";
                  }
                } // state
```

```
                list address-family {
                  key "afi";
                  description
                    "Per-vrf per-af params.";
                  leaf afi {
                    type ldp-address-family;
                    description
                      "Address family type value.";
                  }
                  container config {
                    description
                      "Configuration data.";
                    leaf enable {
                      type boolean;
                      description
                        "Enable the address family on the interface.";
                    }

                    container ipv4 {
                      must "/if:interfaces/if:interface"
                        + "[name = current()/../../../interface]/"
                        + "ip:ipv4" {
                        description
                          "Only if IPv4 is enabled on the interface.";
                      }
                      description
                        "IPv4 address family.";
                      leaf transport-address {
                        type union {
                          type enumeration {
                            enum "use-interface-address" {
                              description
                              "Use interface address as the transport
                               address.";
                            }
                          }
                          type inet:ipv4-address;
                        }
                        description
                          "IP address to be advertised as the LDP
                           transport address.";
                      }
                    }

                    container ipv6 {
                      must "/if:interfaces/if:interface"
                        + "[name = current()/../../../interface]/"
                        + "ip:ipv6" {
```

```
                      description
                        "Only if IPv6 is enabled on the interface.";
                    }
                    description
                      "IPv6 address family.";
                    leaf transport-address {
                      type union {
                        type enumeration {
                          enum "use-interface-address" {
                            description
                              "Use interface address as the transport
                               address.";
                          }
                        }
                        type inet:ipv4-address;
                      }
                      description
                        "IP address to be advertised as the LDP
                         transport address.";
                    }
                  } // ipv6
                }
                container state {
                  config false;
                  description
                    "Operational state data.";
                  leaf enable {
                    type boolean;
                    description
                      "Enable the address family on the interface.";
                  }

                  container ipv4 {
                    must "/if:interfaces/if:interface"
                      + "[name = current()/../../../interface]/"
                      + "ip:ipv4" {
                      description
                        "Only if IPv4 is enabled on the interface.";
                    }
                    description
                      "IPv4 address family.";
                    leaf transport-address {
                      type union {
                        type enumeration {

                          enum "use-interface-address" {
                            description
                              "Use interface address as the transport
```

```
                    address.";
                }
              }
              type inet:ipv4-address;
            }
            description
              "IP address to be advertised as the LDP
               transport address.";
          }

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

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

            uses adjacency-state-attributes;

            leaf peer {
              type leafref {
                path "../../../../../../../../../peers/peer/"
                  + "lsr-id";
              }
              description
                "LDP peer from this adjacency.";
            }
          } // hello-adjacencies
        }
        container ipv6 {
          must "/if:interfaces/if:interface"
            + "[name = current()/../../../interface]/"
            + "ip:ipv6" {
            description
              "Only if IPv6 is enabled on the interface.";
          }
          description
            "IPv6 address family.";
          leaf transport-address {
            type union {
              type enumeration {
                enum "use-interface-address" {
                  description
                  "Use interface address as the transport
                   address.";
```

```
                          }
                        }
                        type inet:ipv4-address;
                      }
                      description
                        "IP address to be advertised as the LDP
                         transport address.";
                    }

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

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

                      uses adjacency-state-attributes;

                      leaf peer {
                        type leafref {
                          path "../../../../../../../../../peers/peer/"
                            + "lsr-id";
                        }
                        description
                          "LDP peer from this adjacency.";
                      }
                    } // hello-adjacencies
                  } // ipv6
                }
              } // address-family
            } // list interface
          } // interfaces

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

              description
                "Configuration data.";
              uses extended-discovery-timers;
              uses extended-discovery-policy-attributes;
            }
            container state {
```

```
                config false;
                description
                  "Operational state data.";
                uses extended-discovery-timers;
                uses extended-discovery-policy-attributes;
              }

              list address-family {
                key "afi";
                description
                  "Per-af params.";
                leaf afi {
                  type ldp-address-family;
                  description
                    "Address family type value.";
                }

                container state {
                  config false;
                  description
                    "Operational state data.";

                  container ipv4 {
                    when "../../afi = 'ipv4'" {
                      description
                        "For IPv4.";
                    }
                    description
                      "IPv4 address family.";
                    list hello-adjacencies {
                      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 peer {
                        type leafref {
```

```
                        path "../../../../../../../../../peers/peer/"
                          + "lsr-id";
                      }
                      description
                        "LDP peer from this adjacency.";
                    }
                  } // hello-adjacencies
                } // ipv4

                container ipv6 {
                  when "../../afi = 'ipv6'" {
                    description
                      "For IPv6.";
                  }
                  description
                    "IPv6 address family.";
                  list hello-adjacencies {
                    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 adjacency-state-attributes;

                    leaf peer {
                      type leafref {
                        path "../../../../../../../../../peers/peer/"
                          + "lsr-id";
                      }
                      description
                        "LDP peer from this adjacency.";
                    }
                  } // hello-adjacencies
                } // ipv6
              } // state

              container ipv4 {
                when "../afi = 'ipv4'" {
                  description
```

```
                    "For IPv4.";
                }
                description
                  "IPv4 address family.";
                list target {
                  key "adjacent-address";
                  description
                    "Targeted discovery params.";

                  leaf adjacent-address {
                    type inet:ipv4-address;
                    description
                      "Configures a remote LDP neighbor and enables
                       extended LDP discovery of the specified
                       neighbor.";
                  }
                  container config {
                    description
                      "Configuration data.";
                    leaf enable {
                      type boolean;
                      description
                        "Enable the target.";
                    }
                    leaf local-address {
                      type inet:ipv4-address;
                      description
                        "The local address.";
                    }
                  }
                  container state {
                    config false;
                    description
                      "Operational state data.";
                    leaf enable {
                      type boolean;
                      description
                        "Enable the target.";
                    }
                    leaf local-address {
                      type inet:ipv4-address;
                      description
                        "The local address.";
                    }
                  } // state
                }
              } // ipv4
              container ipv6 {
```

```
                  when "../afi = 'ipv6'" {
                    description
                      "For IPv6.";
                  }
                  description
                    "IPv6 address family.";
                  list target {
                    key "adjacent-address";
                    description
                      "Targeted discovery params.";

                    leaf adjacent-address {
                      type inet:ipv6-address;
                      description
                        "Configures a remote LDP neighbor and enables
                         extended LDP discovery of the specified
                         neighbor.";
                    }
                    container config {
                      description
                        "Configuration data.";
                      leaf enable {
                        type boolean;
                        description
                          "Enable the target.";
                      }
                      leaf local-address {
                        type inet:ipv6-address;
                        description
                          "The local address.";
                      }
                    }
                    container state {
                      config false;
                      description
                        "Operational state data.";
                      leaf enable {
                        type boolean;
                        description
                          "Enable the target.";
                      }
                      leaf local-address {
                        type inet:ipv6-address;
                        description
                          "The local address.";
                      }
                    } // state
                  }
```

```
              } // ipv6
            } // address-family
          } // targeted
        } // discovery

        container forwarding-nexthop {
          if-feature forwarding-nexthop-config;
          description
            "Configuration for forwarding nexthop.";

          container interfaces {
            description
              "A list of interfaces on which forwarding is
               disabled.";

            list interface {
              key "interface";
              description
                "List of LDP interfaces.";
              leaf interface {
                type mpls-interface-ref;
                description
                  "Interface.";
              }
              list address-family {
                key "afi";
                description
                  "Per-vrf per-af params.";
                leaf afi {
                  type ldp-address-family;
                  description
                    "Address family type value.";
                }
                container config {
                  description
                    "Configuration data.";
                  leaf ldp-disable {
                    type boolean;
                    description
                      "Disable LDP forwarding on the interface.";
                  }
                  leaf mldp-disable {
                    if-feature mldp;
                    type boolean;
                    description
                      "Disable mLDP forwarding on the interface.";
                  }
                }
```

```
                 container state {
                   config false;
                   description
                     "Operational state data.";
                   leaf ldp-disable {
                     type boolean;
                     description
                       "Disable LDP forwarding on the interface.";
                   }
                   leaf mldp-disable {
                     if-feature mldp;

                     type boolean;
                     description
                       "Disable mLDP forwarding on the interface.";
                   }
                 }
               } // address-family
             } // list interface
           } // interfaces
         } // forwarding-nexthop
         uses policy-container {
           if-feature all-af-policy-config;
         }
       } // global

       container peers {
         description
           "Peers configuration attributes.";

         container config {
           description
             "Configuration data.";
           uses peer-authentication {
             if-feature global-session-authentication;
           }
           uses peer-attributes;

           container session-downstream-on-demand {
             if-feature session-downstream-on-demand-config;
             description
               "Session downstream-on-demand attributes.";
             leaf enable {
               type boolean;
               description
                 "'true' if session downstream-on-demand is enabled.";
             }
             leaf peer-list {
```

```
               type peer-list-ref;
               description
                 "The name of a peer ACL.";
             }
           }
         }
         container state {
           config false;
           description
             "Operational state data.";
           uses peer-authentication {
             if-feature global-session-authentication;
           }
           uses peer-attributes;

           container session-downstream-on-demand {
             if-feature session-downstream-on-demand-config;
             description
               "Session downstream-on-demand attributes.";
             leaf enable {
               type boolean;
               description
                 "'true' if session downstream-on-demand is enabled.";
             }
             leaf peer-list {
               type peer-list-ref;
               description
                 "The name of a peer ACL.";
             }
           }
         }

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

           leaf lsr-id {
             type yang:dotted-quad;
             description "LSR ID.";
           }

           container config {
             description
               "Configuration data.";
             leaf admin-down {
               type boolean;
               default false;
```

```
              description
                "'true' to disable the peer.";
            }

            container capability {
              description
                "Per peer capability";
              container mldp {
                if-feature mldp;
                description
                  "mLDP capabilities.";
                uses mldp-capabilities;
              }
            }

            uses peer-af-policy-container {
              if-feature all-af-policy-config;
            }

            uses peer-authentication;

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

            uses peer-attributes {
              if-feature per-peer-session-attributes-config;
            }

            container address-family {
              description
                "Per-vrf per-af params.";
              container ipv4 {
                description
                  "IPv4 address family.";
                uses peer-af-policy-container;
              }
              container ipv6 {
                description
                  "IPv6 address family.";
                uses peer-af-policy-container;
              } // ipv6
            } // address-family
          }
          container state {
            config false;
            description
              "Operational state data.";
```

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

              container capability {
                description
                  "Per peer capability";
                container mldp {
                  if-feature mldp;
                  description
                    "mLDP capabilities.";
                  uses mldp-capabilities;
                }
              }

              uses peer-af-policy-container {
                if-feature all-af-policy-config;
              }

              uses peer-authentication;

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

              uses peer-attributes {
                if-feature per-peer-session-attributes-config;
              }

              container address-family {
                description
                  "Per-vrf per-af params.";
                container ipv4 {
                  description
                    "IPv4 address family.";
                  uses peer-af-policy-container;

                  list hello-adjacencies {
                    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 mpls-interface-ref;
                      description "Interface for this adjacency.";
                    }
                  } // hello-adjacencies
                } // ipv4
                container ipv6 {
                  description
                    "IPv6 address family.";
                  uses peer-af-policy-container;

                  list hello-adjacencies {
                    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 adjacency-state-attributes;

                    leaf interface {
                      type mpls-interface-ref;
                      description "Interface for this adjacency.";
                    }
                  } // hello-adjacencies
                } // ipv6
              } // address-family

              uses peer-state-derived;
            } // state
          } // list peer
```

```
      } // peers
    } // container mpls-ldp
  }

  /*
   * RPCs
   */
  rpc mpls-ldp-clear-peer {
    description
      "Clears the session to the peer.";
    input {
      leaf lsr-id {
        type union {
          type yang:dotted-quad;
          type uint32;
        }
        description
          "LSR ID of peer to be cleared. If this is not provided
           then all peers are cleared";
      }
    }
  }

  rpc mpls-ldp-clear-hello-adjacency {
    description
      "Clears the hello adjacency";
    input {
      container hello-adjacency {
        description
          "Link adjacency or targettted adjacency. If this is not
           provided then all hello adjacencies are cleared";
        choice hello-adjacency-type {
          description "Adjacency type.";
          case targeted {
            container targeted {
              presence "Present to clear targeted adjacencies.";
              description
                "Clear targeted adjacencies.";
              leaf target-address {
                type inet:ip-address;
                description
                  "The target address. If this is not provided then
                   all targeted adjacencies are cleared";
              }
            } // targeted
          }
          case link {
            container link {
```

```
               presence "Present to clear link adjacencies.";
               description
                 "Clear link adjacencies.";
               leaf next-hop-interface {
                 type mpls-interface-ref;
                 description

                   "Interface connecting to next-hop. If this is not
                    provided then all link adjacencies are cleared.";
               }
               leaf next-hop-address {
                 type inet:ip-address;
                 must "../next-hop-interface" {
                   description
                     "Applicable when interface is specified.";
                 }
                 description
                   "IP address of next-hop. If this is not provided
                    then adjacencies to all next-hops on the given
                    interface are cleared.";
               } // next-hop-address
             } // link
           }
         }
       }
     }
   }

   rpc mpls-ldp-clear-peer-statistics {
     description
       "Clears protocol statistics (e.g. sent and received
        counters).";
     input {
       leaf lsr-id {
         type union {
           type yang:dotted-quad;
           type uint32;
         }
         description
           "LSR ID of peer whose statistic are to be cleared.
            If this is not provided then all peers statistics are
            cleared";
       }
     }
   }

   /*
    * Notifications
```

```
    */
   notification mpls-ldp-peer-event {

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

   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.";
     }
     uses ldp-adjacency-ref;
   }

   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.";
     }
     uses ldp-fec-event;
   }

   notification mpls-mldp-fec-event {
     description
       "Notification event for a change of FEC status.";
     leaf event-type {
       type oper-status-event-type;
       description "Event type.";
     }
     uses mldp-fec-event;
   }
 }

 <CODE ENDS>
```

Figure 22

7. Security Considerations

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

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

8. IANA Considerations

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

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

9. Acknowledgments

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

10. References

10.1. Normative References

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-22 (work in progress), July 2016.

[I-D.rtgyangdt-rtgwg-ni-model]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "Network Instance Model", draft-rtgyangdt-rtgwg-ni-model-00 (work in progress), May 2016.

[I-D.saad-mpls-base-yang]
Raza, K., Gandhi, R., Liu, X., Beeram, V., Saad, T., Bryskin, I., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for MPLS Base", draft-saad-mpls-base-yang-00 (work in progress), May 2016.

[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>.

[RFC3478] Leelanivas, M., Rekhter, Y., and R. Aggarwal, "Graceful Restart Mechanism for Label Distribution Protocol", RFC 3478, DOI 10.17487/RFC3478, February 2003, <http://www.rfc-editor.org/info/rfc3478>.

[RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <http://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, <http://www.rfc-editor.org/info/rfc5331>.

[RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL. Le Roux, "LDP Capabilities", RFC 5561, DOI 10.17487/RFC5561, July 2009, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc5919>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://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, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <http://www.rfc-editor.org/info/rfc6388>.

[RFC6389] Aggarwal, R. and JL. Le Roux, "MPLS Upstream Label Assignment for LDP", RFC 6389, DOI 10.17487/RFC6389, November 2011, <http://www.rfc-editor.org/info/rfc6389>.

[RFC6512] Wijnands, IJ., Rosen, E., Napierala, M., and N. Leymann, "Using Multipoint LDP When the Backbone Has No Route to the Root", RFC 6512, DOI 10.17487/RFC6512, February 2012, <http://www.rfc-editor.org/info/rfc6512>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6826] Wijnands, IJ., Ed., Eckert, T., Leymann, N., and M. Napierala, "Multipoint LDP In-Band Signaling for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6826, DOI 10.17487/RFC6826, January 2013, <http://www.rfc-editor.org/info/rfc6826>.

[RFC7060] Napierala, M., Rosen, E., and IJ. Wijnands, "Using LDP Multipoint Extensions on Targeted LDP Sessions", RFC 7060, DOI 10.17487/RFC7060, November 2013, <http://www.rfc-editor.org/info/rfc7060>.

[RFC7140] Jin, L., Jounay, F., Wijnands, IJ., and N. Leymann, "LDP Extensions for Hub and Spoke Multipoint Label Switched Path", RFC 7140, DOI 10.17487/RFC7140, March 2014, <http://www.rfc-editor.org/info/rfc7140>.

[RFC7246] Wijnands, IJ., Ed., Hitchen, P., Leymann, N., Henderickx, W., Gulko, A., and J. Tantsura, "Multipoint Label Distribution Protocol In-Band Signaling in a Virtual Routing and Forwarding (VRF) Table Context", RFC 7246, DOI 10.17487/RFC7246, June 2014, <http://www.rfc-editor.org/info/rfc7246>.

[RFC7438] Wijnands, IJ., Ed., Rosen, E., Gulko, A., Joorde, U., and J. Tantsura, "Multipoint LDP (mLDP) In-Band Signaling with Wildcards", RFC 7438, DOI 10.17487/RFC7438, January 2015, <http://www.rfc-editor.org/info/rfc7438>.

[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, <http://www.rfc-editor.org/info/rfc7552>.

[RFC7715] Wijnands, IJ., Ed., Raza, K., Atlas, A., Tantsura, J., and Q. Zhao, "Multipoint LDP (mLDP) Node Protection", RFC 7715, DOI 10.17487/RFC7715, January 2016, <http://www.rfc-editor.org/info/rfc7715>.

10.2. Informative References

[I-D.ietf-rtgwg-policy-model] Shaikh, A., Shakir, R., D'Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-ietf-rtgwg-policy-model-01 (work in progress), April 2016.

[I-D.iwijnand-mpls-mldp-multi-topology] Wijnands, I. and K. Raza, "mLDP Extensions for Multi Topology Routing", draft-iwijnand-mpls-mldp-multi-topology-03 (work in progress), June 2013.

[I-D.openconfig-netmod-opstate] Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <http://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, <http://www.rfc-editor.org/info/rfc7307>.

Appendix A. Additional Contributors

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

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

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

Authors' Addresses

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

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

Sowmya Krishnaswamy
Cisco Systems, Inc.
Email: sowkrish@cisco.com

Xufeng Liu
Ericsson
Email: xliu@kuatrotech.com

Jeff Tantsura
Ericsson
Email: jeff.tantsura@ericsson.com

Santosh Esale
Juniper Networks
Email: sesale@juniper.net

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

Loa Andersson
Huawei Technologies
Email: loa@pi.nu

Himanshu Shah
Ciena Corporation
Email: hshah@ciena.com

Matthew Bocci
Alcatel-Lucent
Email: matthew.bocci@alcatel-lucent.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 14, 2016

Kishore Tiruveedhula, Ed.
Juniper Networks
Uwe Joorde
Deutsche Telekom
Arvind Venkateswaran
Cisco Systems

October 12, 2015

Definitions of Managed Objects for the LDP Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths
draft-tiruveedhula-mpls-mldp-mib-05

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols. In particular it defines objects for managing multicast LDP point-to-multipoint (P2MP) and multipoint-to-multipoint (MP2MP) Label Switched Paths. The MIB module defined in this document is extension of LDP MIB defined in RFC3815 which supports only for LDP point-to-point LSPs.

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 April 14, 2016.

Copyright Notice

Copyright (c) 2015 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

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols. In particular it defines objects for managing multicast LDP point-to-multipoint (P2MP) and multipoint-to-multipoint (MP2MP) Label Switched Paths. The MIB module defined in this document is extension of LDP MIB defined in RFC3815 which supports only for LDP point-to-point LSPs.

The RFC3815 describes only unicast Managed objects for the Label distribution protocol. The RFC6388 describes LDP protocol extensions for the point to multipoint and multipoint to multipoint LSPs. The RFC 6826 describes multicast LDP inband signalling for P2MP and MP2MP LSPs.

This document defines a MIB module for managing and controlling mLDP P2MP and MP2MP LSPs. It builds on the objects and tables defined in [RFC3815] for mLDP MIB.

2. The Internet-Standard Management Framework

[[CREF1: The title and text for this section has been copied from the official boilerplate, and should not be modified unless the official boilerplate text from the OPS Area web site has changed. See RFC4818 section 3.1 for a discussion of the boilerplate section.]]

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

3. Conventions

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 BCP 14, RFC 2119 [RFC2119].

4. Overview

This document focusses on the management of following multicast LDP (mLDP) features, which were defined after unicast LDP [RFC5036].

RFC6388: Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths.

RFC6826: Multipoint LDP In-Band Signaling for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths.

RFC7060: Using LDP Multipoint Extensions on Targeted LDP Sessions.

[I-D.ietf-rtgwg-mofrr] Multicast only Fast Re-Route.

[I-D.ietf-mpls-mldp-node-protection] mLDP Node Protection.

For all the above features, the mLDP MIB needs to include the following information:

- Session Capability (P2MP, MP2MP) information: configured capability, negotiated capability.

- mLDP FECs: include opaque information (Generic LSP Identifier, source and group address) and MoFRR enable.

- Primary and backup upstream session when mLDP MoFRR enabled.

- Active and inactive upstream session for make before break.

- mLDP Traffic stats per mLDP Fec: The traffic stats for mLDP fec.

- mLDP Traffic stats per per Interface: The mLDP traffic stats per Interface.

- Traps when mLDP Fec LSP up, down.

5. Future Considerations

Any new opaque TLVs added for any other mLDP features, the opaque value object in the mplsMldpFecTable need to be enhanced accordingly.

6. Structure of the MIB Module

This section describes the structure of the mLDP MIB. In this MIB MPLS-MLDP-STD-MIB, scalar objects, table objects and notifications are defined. Following section describes in details about each object.

6.1. Summary of mLDP Scalar Objects

New scalar objects mplsMldpP2mpCapable and mplsMldpMp2mpCapable are defined to provide the mLDP capabilities of P2MP, MP2MP support.

New scalar objects mplsMldpMbbCapable and mplsMldpMbbTime are defined to provide MBB capability information.

New scalar object mplsMldpNumFecs which will give the total number of mLDP FECs setup on the LSR.

Another New scalar object mplsMldpNumFecsActive, which will give the total number of active mLDP FECs.

New scalar objects mplsMldpPlrCapable, mplsMldpMptCapable, mplsMldpProtLsrCapable and mplsMldpNodeProtCapable are defined to provide mLDP node protection capabilities.

6.2. Summary of mLDP Table Objects

mplsLdpPeerCapabilityTable to include peer capability information.

mplsMldpSessionStatsTable : This table contains the number of mLDP FECs received and advertised to particular LDP session.

mplsMldpFecTable: This table is similar to point to point mplsLdpFecTable and will have mLDP specific Fec information.

mplsMldpFecBranchStatsTable : This table contains the traffic statistics for the given mLDP FECs on particular interface.

mplsMldpFecUpstreamSessTable : Includes the upstream session info for the particular mLDP Fec and also includes the primary or backup upstream session, that may be used for mLDP MoFRR.

mplsMldpInterfaceStatsTable : This table contains the traffic statistics for all mLDP related FECs.

7. mLDP Scalar Objects

There are ten scalars, listed below are defined for this MIB module.

7.1. mplsMldpP2mpCapable

The mplsMldpP2mpCapable scalar object denotes whether the LSR is capable of supporting multicast LDP with Point-to-Multipoint capability.

7.2. mplsMldpMp2mpCapable

The mplsMldpMp2mpCapable scalar object denotes whether the LSR is capable of supporting multicast LDP with Multipoint-to-Multipoint LSPs.

7.3. mplsMldpMbbCapable

The mplsMldpMbbCapable scalar object denotes whether the LSR is capable of supporting multicast LDP with MBB (make before break) feature mentioned in the section 8 of RFC 6388 .

7.4. mplsMldpMbbTime

The mplsMldpMbbTime scalar object denotes MBB time for which LSR is waiting for MBB Ack from upstream node. This timer helps LSR to prevent waiting indefinitely for the MBB Notification from upstream node.

7.5. mplsMldpNumFecs

The mplsMldpNumFecs provides a read-only counter of the number of mLDP FECs setup on this LSR.

7.6. mplsMldpNumFecsActive

The mplsMldpNumFecsActive provides a read-only counter of the number of mLDP FECs Active on this LSR.

7.7. mplsMldpPlrCapable

The mplsMldpPlrCapable scalar object denotes whether the LSR is capable of supporting PLR capability as specified in the section 5.1 of [I-D.ietf-mpls-mldp-node-protection].

7.8. mplsMldpMptCapable

The mplsMldpMptCapable scalar object denotes whether the LSR is capable of supporting MPT capability as specified in the section 5.2 of [I-D.ietf-mpls-mldp-node-protection].

7.9. mplsMldpProtLsrCapable

The mplsMldpProtLsrCapable scalar object denotes whether the LSR is capable of supporting the "Protected LSR" capability as specified in the section 5.3 of [I-D.ietf-mpls-mldp-node-protection].

7.10. mplsMldpNodeProtCapable

The mplsMldpNodeProtCapable scalar object denotes whether the LSR is capable of supporting the "Node Protection" capability as specified in the section 5.4 of [I-D.ietf-mpls-mldp-node-protection].

8. mLDP Table Objects

8.1. LDP Peer Capability Table mplsLdpPeerCapabilityTable

The new table mplsLdpPeerCapabilityTable is read-only table, which contains learned capability information from LDP peer. This table augments the mplsLdpPeerTable, which is defined in RFC 3815.

8.2. mLDP Session Stats Table: mplsMldpSessionStatsTable

The mplsMldpSessionStatsTable is a read-only table which contains mLDP statistical information on sessions. This table augments the mplsLdpSessionStatsTable, which is defined in the RFC 3815.

8.3. mLDP Fec Table: mplsMldpFecTable

The mplsMldpFecTable is a table which contains FEC (Forwarding Equivalence Class) information relating to point to multi-point and multipoint to multipoint LDP LSP. Each entry/row represents a single FEC Element. This table is similar LDP LSP FEC Table, mplsLdpLspFecTable, which is defined in the RFC 3815, which associates FECs with the LSPs.

8.4. mLDP Fec Branch Traffic statistics Table: mplsMldpFecBranchStatsTable

This table mplsMldpFecBranchStatsTable gives the information about number of packets and number of bytes sent out on particualr downstream session or on outgoing interface.

8.5. mLDP Fec Upstreaam Session Table: mplsMldpFecUpstreamSessTable

The mplsMldpFecUpstreamSessTable is a read-only table which contains mLDP upstream session information for mLDP Fec. This table is similar to mplsInSegmentLdpLspTable. This table will also have information about primary, backup upstream session, and also indicates whether the label is in MBB request or MBB Ack received state.

8.6. mLDP Interface Traffic statistics Table: mplsMldpInterfaceStatsTable

This table mplsMldpInterfaceStatsTable gives the information about number of mLDP packets and number of mLDP bytes sent and received on particualr interface for all mLDP FECs.

9. The mLDP Notifications

The RFC 3815 defined some of the notifications related to session and P2P Fec. In this MIB, the following notification added to support mLDP features.

The mplsMldpFecUp and mplsMldpFecDown notifications are generated when mLDP FEC changes the state to UP and Down.

The mplsMldpMoFrrStatusChange notification is generated when mLDP MoFRR status switches from primary to backup path and vice versa.

10. Relationship to Other MIB Modules

This section describes relationships between MIB tables defined in this document as part of MPLS-MLDP-STD-MIB, and the tables defined in MPLS-LDP-STD-MIB [RFC3815] and MPLS-LSR-STD-MIB [RFC3813].

The Figure 1 shows the diagrammatic representation of the relationship between MPLS-MLDP-STD-MIB, MPLS-LDP-STD-MIB and MPLS-LSR-STD-MIB. An arrow in the Figure shows that the MIB table pointed from contains a reference to the MIB table pointed to.

10.1. Diagrammatic Representation

```
       mplsMldpFecTable
           |
           |
           v
          mplsLdpLspFecTable
              |      |    |
              |      |    +----------------+
              |      |                     |
              |      +---------------+     v
              |                      |     mplsMldpFecUpstreamSessTable
              +--->mplsXCTable--+    |
                                |    |
                                |    |
                                |    |
                                |  mplsMldpFecBranchStatsTable
                                |        |
                                |        |
                                |        |
       mplsInSegmentTable<------+        |
                                |        |
                                v        |
                     mplsOutSegmentTable<-+
```

Figure 1 : Dependencies Between MIB Tables

Figure 1

10.2. Relationship to the LSR MIB

The LSR MIB [RFC3813] have below tables, which cross connects the incoming label to outgoing label. Below Tables will be used for mLDP also in the similar way as in the point to point LDP LSPs.

mplsXCTable

mplsInSegmentTable

mplsOutSegmentTable

10.3. Relationship to the LDP MIB

The MIB module defined in this document is extension of MPLS-LDP-STD-MIB to support multicast LDP features.

Below optional tables in MPLS-LDP-STD-MIB, will also be used in mLDP for associating the mLDP LSPs to LSR-MIB tables.

mplsLdpLspFecTable

mplsInSegmentLdpLspTable

mplsOutSegmentLdpLspTable

11. Multicast MPLS Label Distribution Protocol MIB Definitions

```
MPLS-MLDP-STD-MIB DEFINITIONS ::= BEGIN
IMPORTS
   MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
   Unsigned32, Counter32, Counter64, TimeTicks
      FROM SNMPv2-SMI                                   -- RFC 2578
   MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
      FROM SNMPv2-CONF                                  -- RFC 2580
   TruthValue, RowStatus, StorageType, TimeStamp
      FROM SNMPv2-TC                                    -- RFC 2579

   InterfaceIndex
        FROM IF-MIB                                    --  [RFC2020]

   mplsStdMIB, MplsLdpIdentifier
      FROM MPLS-TC-STD-MIB                              -- RFC 3811

   MplsIndexType
      FROM MPLS-LSR-STD-MIB                             -- RFC 3813

   IndexInteger, IndexIntegerNextFree
      FROM DIFFSERV-MIB                                 -- RFC 3289

   InetAddress, InetAddressType
      FROM INET-ADDRESS-MIB                             -- RFC 4001

   mplsLdpStdMIB
      FROM MPLS-LDP-STD-MIB                             -- RFC 3815
   ;

 mplsMldpStdMIB MODULE-IDENTITY
      LAST-UPDATED "201510120000Z"  -- October 12, 2015
      ORGANIZATION "Multiprotocol Label Switching (mpls)
                    Working Group"
      CONTACT-INFO

          "     Kishore Tiruveedhula
                Juniper Networks
                Email: kishoret@juniper.net
```

Uwe Joorde
Deutsche Telekom
Email: Uwe.Joorde@telekom.de

Arvind Venkateswaran
Cisco Systems
EMail: arvvenka@cisco.com

Comments about this document should be emailed
directly to the MPLS working group mailing list at
mpls@lists.ietf.org"

```
         DESCRIPTION
           "Copyright (c) 2009 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 in effect on the
            date of publication of this document
            (http://trustee.ietf.org/license-info). Please review these
            documents carefully, as they describe your rights and
            restrictions with respect to this document.

            The initial version of this MIB module was published in
            RFC XXXX. For full legal notices see the RFC itself or see:
            http://www.ietf.org/copyrights/ianamib.html
-- RFC Editor. Please replace XXXX with the RFC number for this
-- document and remove this note.

            This MIB module contains managed object definitions for mLDP LSPS
            defined in Label Distribution Protocol Extensions Point-to-Multipoin
t and
            Multipoint-to-Multipoint Label Switched Paths, RFC 6388, November
            2011."

         REVISION "2015101200Z"  -- October 12, 2015
         DESCRIPTION
           "Initial version issued as part of RFC XXXX."
-- RFC Editor. Please replace XXXX with the RFC number for this
-- document and remove this note.

         ::= { mplsStdMIB YYY }
-- RFC Editor. Please replace YYY with the codepoint issued by IANA
-- and remove this note.

   -- Top level components of this MIB module.


   -- notifications
```

```
   mplsMldpNotifications OBJECT IDENTIFIER ::= { mplsMldpStdMIB 0 }
   -- tables, scalars
   mplsMldpScalars       OBJECT IDENTIFIER ::= { mplsMldpStdMIB 1 }
   mplsMldpObjects       OBJECT IDENTIFIER ::= { mplsMldpStdMIB 2 }

  -- MPLS mLDP LSP scalars.

   mplsMldpP2mpCapable OBJECT-TYPE
         SYNTAX      INTEGER {
                       enable(1),
                       disable(2)
                     }
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
            "This object provides the P2MP capability of the LSR."

         REFERENCE
           "Section 2.1 of [RFC6388]."

      ::= { mplsMldpScalars 1 }


  mplsMldpMp2mpCapable OBJECT-TYPE
         SYNTAX      INTEGER {
                       enable(1),
                       disable(2)
                     }
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
           "This object provides MP2MP capability of the LSR."

         REFERENCE
           "Section 3.1 of [RFC6388]."

      ::= { mplsMldpScalars 2 }


  mplsMldpMbbCapable OBJECT-TYPE
         SYNTAX      INTEGER {
                       enable(1),
                       disable(2)
                     }
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
           "This object provides MBB (make before break) capability of the LSR."
```

```
         REFERENCE
           "Section 8.3 of [RFC6388]."

      ::= { mplsMldpScalars 3 }

     mplsMldpMbbTime OBJECT-TYPE
         SYNTAX      Unsigned32 (1..300)
         UNITS       "seconds"
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
           "The 32-bit unsigned integer value provides the time for waiting MBB
Ack
            from upstream node."

         DEFVAL { 30 }
         ::= { mplsMldpScalars 4 }


   mplsMldpNumFecs OBJECT-TYPE
      SYNTAX        Unsigned32
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION
           "The number of active and passive mLdp Fecs on this device."

      ::= { mplsMldpScalars 5 }

   mplsMldpNumFecsActive OBJECT-TYPE
      SYNTAX        Unsigned32
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION
           "The number of mLdp FECs Active on this device. The mLDP FEC is
           considered active if the mplsMldpFecOperStatus is up(1)."

      ::= { mplsMldpScalars 6 }

   mplsMldpPlrCapable OBJECT-TYPE
         SYNTAX      INTEGER {
                       enable(1),
                       disable(2)
                     }
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
           "This object provides Point of Local Repair (PLR)
            capability of the LSR."
```

```
         REFERENCE
           "Section 5.1 of [I-D.ietf-mpls-mldp-node-protection]."

      ::= { mplsMldpScalars 7 }

   mplsMldpMptCapable OBJECT-TYPE
         SYNTAX      INTEGER {
                       enable(1),
                       disable(2)
                     }
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
           "This object provides  Merge Point (MPT) capability of the LSR."

         REFERENCE
           "Section 5.2 of [I-D.ietf-mpls-mldp-node-protection]."

      ::= { mplsMldpScalars 8 }

   mplsMldProtLsrCapable OBJECT-TYPE
         SYNTAX      INTEGER {
                       enable(1),
                       disable(2)
                     }
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
           "This object provides Protected LSR capability."

         REFERENCE
           "Section 5.3 of [I-D.ietf-mpls-mldp-node-protection]."

      ::= { mplsMldpScalars 9 }

   mplsMldProtNodeProtCapable OBJECT-TYPE
         SYNTAX      INTEGER {
                       enable(1),
                       disable(2)
                     }
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
           "This object provides Node Protection capability of the LSR."

         REFERENCE
           "Section 5.3 of [I-D.ietf-mpls-mldp-node-protection]."
```

```
      ::= { mplsMldpScalars 10 }

   -- End of MPLS mLDP scalars.

   -- MPLS mLDP tables.

     --
     -- The MPLS LDP Peer Capability Table
     --

     mplsLdpPeerCapabilityTable OBJECT-TYPE
         SYNTAX      SEQUENCE OF MplsLdpPeerCapabilityEntry
         MAX-ACCESS  not-accessible
         STATUS      current
         DESCRIPTION
             "This table will have learned information relating to Mldp.
         ::= { mplsMldpObjects 1 }

     mplsLdpPeerCapabilityEntry OBJECT-TYPE
         SYNTAX      MplsLdpPeerCapabilityEntry
         MAX-ACCESS  not-accessible
         STATUS      current
         DESCRIPTION
             "Information about a single Peer which is related
             to a Session.  This table is augmented by
             the mplsLdpSessionTable."
         INDEX       { mplsLdpEntityLdpId,
                       mplsLdpEntityIndex,
                       mplsLdpPeerLdpId }

         ::= { mplsLdpPeerCapabilityTable 1 }

     mplsLdpPeerCapabilityEntry ::= SEQUENCE {
         mplsLdpPeerLdpId                MplsLdpIdentifier,
         mplsLdpPeerCapability           Integer32,
     }

     mplsLdpPeerCapability OBJECT-TYPE
         SYNTAX      BITS {
                        none (0),
                        p2mp (1),
                        mp2mp(2),
                        mbb  (3),
                        upstream-label-assignment  (4),
                        dynamic (5),
                        plr (6),
                        mpt (7),
                        prot-lsr (8),
```

```
                        node-prot (9)
                    }
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
          " This will indicate the LDP capability information about peer.
          The p2mp indicates peer supports P2MP Capability.
          The mp2mp indicates peer supports MP2MP Capability.
          The mbb indicates peer supports MBB Capability.
          The upstream-label-assignment indicates peer supports Upstream label
          assignment Capability.
          The dynamic indicates peer supports dynamic Capability.
          The plr indicates Point of Local Repair Capability.
          The mpt indicates Point of Merge Point Capability.
          The prot-lsr indicates Protected LSR Capability.
          The node-prot indicates Node Protection LSR Capability.
          "

        REFERENCE
        "RFC6388, Section 2.1 for P2MP Capability TLV.
        and the section 3.1 for MP2MP Capability TLV.
        The RFC6388 for MBB Capability TLV.
        RFC5561 Section 9 for Dynamic Capability Announcement TLV.
        RFC6389 Section 3 for Upstream Label Assignment Capability TLV.
        Section 5 of [I-D.ietf-mpls-mldp-node-protection] describes for Point o
f Local Repair (plr)
        capability,  Merge Point (mpt)  capability,
        The Protected LSR (port-lsr) and Node Protection (node-prot) Capability
. "

        ::= { mplsLdpPeerCapability 2 }

    --
    -- The MPLS mLDP Session Statistics Table
    --

    mplsMldpSessionStatsTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF MplsMldpSessionStatsEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table of statistics related to mLDP on Sessions.
            This table AUGMENTS the mplsLdpSessionStatsTable."
        ::= { mplsMldpObjects 2 }

    mplsMldpSessionStatsEntry OBJECT-TYPE
        SYNTAX      MplsMldpSessionStatsEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
```

```
            "An entry in this table represents mLDP statistical
            information on a single session between an LDP
            Entity and LDP Peer."

        AUGMENTS       { mplsLdpSessionStatsEntry }
        ::= { mplsmMldpSessionStatsTable 1 }

    MplsMldpSessionStatsEntry ::= SEQUENCE {
        mplsMldpSessionStatsNumFecsSent           Counter32,
        mplsMldpSessionStatsNumMbbReqSentState    Counter32,
        mplsMldpSessionStatsNumFecsRcvd           Counter32,
        mplsMldpSessionStatsNumMbbReqRcvdState    Counter32,
        mplsMldpSessionStatsNumMbbResetAckByTimer Counter32
    }

    mplsMldpSessionStatsNumFecsSent OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object counts the number of mLDP FECs sent on this
            session. If the FEC is withdrawn, then this number is
            decremented.

            Discontinuities in the value of this counter can occur
            at re-initialization of the management system, and at
            other times as indicated by the value of
            mplsLdpSessionDiscontinuityTime."

        ::= { mplsMldpSessionStatsEntry 1 }

    mplsMldpSessionStatsNumMbbReqSentState OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object counts the number of mLDP FECs sent on this
            session and waiting for MBB Ack. This counter will get incremented
            when MBB req sent for a label on this session and will get
            decremented when the MBB Ack received.

        ::= { mplsMldpSessionStatsEntry 2 }


    mplsMldpSessionStatsNumFecsRcvd OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
```

```
        DESCRIPTION
            "This object counts the number of mLDP FECs received on this
            session. If the FEC is withdrawn from the downstream session,
            then this is decremented.

            Discontinuities in the value of this counter can occur
            at re-initialization of the management system, and at
            other times as indicated by the value of
            mplsLdpSessionDiscontinuityTime."

        ::= { mplsMldpSessionStatsEntry 3 }

    mplsMldpSessionStatsNumMbbReqRcvdState OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object counts the number of mLDP FECs received on this
            session and waiting for sending MBB Ack. This counter will get
            incremented when MBB req is received for a label on this session
            and will get decremented when the MBB Ack sent."

        ::= { mplsMldpSessionStatsEntry 4 }


    mplsMldpSessionStatsNumMbbResetAckByTimer OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object counts the number mLDP FECs for which the MBB Ack is
            reset by MBB timer, in which the LSR is waiting for MBB ack.

        ::= { mplsMldpSessionStatsEntry 5 }

     --
     -- Mpls mLDP FEC Table
     --

     mplsMldpFecTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF MplsFecEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This table represents the FEC
            (Forwarding Equivalence Class)
            Information associated with an mLDP LSP."
```

```
        ::= { mplsMldpObjects 3 }


    mplsMldpFecEntry OBJECT-TYPE
        SYNTAX      MplsMldpFecEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Each row represents a single mLDP FEC Element."
        INDEX       { mplsMldpFecIndex }

        ::= { mplsMldpFecTable 1 }

    MplsMldpFecEntry ::= SEQUENCE {
        mplsMldpFecIndex                IndexInteger,
        mplsMldpFecType                 INTEGER,
        mplsMldpFecRootAddrType         InetAddressType,
        mplsMldpFecRootAddr             InetAddress,
        mplsMldpFecOpaqueType           INTEGER,
        mplsMldpFecOpaqueGenLspId       Unsigned32,
        mplsMldpFecOpaqueTransitSourceOrBidirAddrType   InetAddressType,
        mplsMldpFecOpaqueTransitSourceOrBidirAddr       InetAddress,
        mplsMldpFecOpaqueTransitGroupAddrType           InetAddressType,
        mplsMldpFecOpaqueTransitGroupAddr               InetAddress,
        mplsMldpFecAdminStatus          INTEGER,
        mplsMldpFecOperStatus           INTEGER,
        mplsMldpFecMoFrr                INTEGER,
        mplsMldpFecLsrState             INTEGER,
        mplsMldpFecUpTime               TimeStamp
    }

    mplsMldpFecIndex OBJECT-TYPE
        SYNTAX      IndexInteger
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The index which uniquely identifies this entry."

        ::= { mplsMldpFecEntry 1 }


    mplsMldpFecType  OBJECT-TYPE
        SYNTAX      INTEGER {
                       p2mp(6),
                       mp2mpUpstream(7),
                       mp2mpDownstream(8)
                    }
        MAX-ACCESS  read-only
```

```
        STATUS      current
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The type of the FEC.  If the value of this object
            is 6, then it is P2MP Fec Type, and 7, 8 are correspond to
            MP2MP upstream and downstream type."

        REFERENCE
            "RFC6388, Section 2.2. The P2MP FEC Element and the section 3.3
            for the MP2MP Fec elements."

        ::= { mplsMldpFecEntry 2 }

    mplsMldpFecRootAddrType  OBJECT-TYPE
        SYNTAX      InetAddressType
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The value of this object is the type of the
            Internet address.  The value of this object,
            decides how the value of the mplsMldpFecRootAddr object
            is interpreted."
        REFERENCE
            "RFC6388, Section 2.2. The P2MP FEC Element and the section 3.3
            for the MP2MP Fec elements."

        ::= { mplsMldpFecEntry 3 }

    mplsMldpFecRootAddr     OBJECT-TYPE
        SYNTAX      InetAddress
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The value of this object is interpreted based
            on the value of the mplsMldpFecRootAddrType object.
            This is ingress node address for the mLDP LSP."

        REFERENCE
            "RFC6388, Section 2.2. The P2MP FEC Element and the section 3.3
            for the MP2MP Fec elements."

        ::= { mplsMldpFecEntry 4 }


    mplsMldpFecOpaqueType  OBJECT-TYPE
        SYNTAX      INTEGER {
                       genericLspId(1),
```

```
                        transitIpv4Source(3),
                        transitIpv6Source(4),
                        transitIpv4Bidir(5),
                        transitIpv6Bidir(6)
                     }
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
           "This is opaque type of the mLDP FEC. The value of this object is
           shown below.

             1 - The Generic LSP Identifier
             3 - Transit IPv4 Source TLV
             4 - Transit IPv6 Source TLV
             5 - Transit IPv4 Bidir TLV
             6 - Transit IPv6 Bidir TLV.
             "
         ::= { mplsMldpFecEntry 5 }

      mplsMldpFecOpaqueGenLspId OBJECT-TYPE
         SYNTAX      Unsigned32
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
             "The 32-bit unsigned integer value which is to represent Generic
             LSP ID. This value is only valid if the mplsMldpFecOpaqueType is
             genericLspId(1), otherwise 0 must be returned."

         REFERENCE
             "RFC6388, Section 2.3.1."

         ::= { mplsMldpFecEntry 6 }

     mplsMldpFecOpaqueTransitSourceOrBidirAddrType        OBJECT-TYPE
         SYNTAX      InetAddressType
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
             "The value of this object is the type of the
             Internet address.  The value of this object,
             decides how the value of the  mplsMldpFecOpaqueTransitSourceOrBidir
Addr
             object is interpreted."
         REFERENCE
             "RFC6826, Section 3.1."

         ::= { mplsMldpFecEntry 7 }

     mplsMldpFecOpaqueTransitSourceOrBidirAddr OBJECT-TYPE
```

```
        SYNTAX      InetAddress
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The value of this object is interpreted based
            on the value of the mplsMldpFecOpaqueTransitSourceOrBidirAddrType
             object.  This is source node address for the mLDP inband LSP."

        REFERENCE
            "RFC6826, Section 3.1."

        ::= { mplsMldpFecEntry 8 }

    mplsMldpFecOpaqueTransitGroupAddrType        OBJECT-TYPE
        SYNTAX      InetAddressType
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The value of this object is the type of the
            Internet address.  The value of this object,
            decides how the value of the  mplsMldpFecOpaqueTransitGroupAddr
            object is interpreted."

        REFERENCE
            "RFC6826, Section 3.2."

        ::= { mplsMldpFecEntry 9 }

    mplsMldpFecOpaqueTransitGroupAddr OBJECT-TYPE
        SYNTAX      InetAddress
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The value of this object is interpreted based
            on the value of the mplsMldpFecOpaqueTransitGroupAddrType
             object.  This is group node address for the mLDP inband LSP."

        REFERENCE
            "RFC6826, Section 3.2."

        ::= { mplsMldpFecEntry 10 }


   mplsMldpFecAdminStatus OBJECT-TYPE
      SYNTAX       INTEGER {
                      up(1),        -- ready to pass data
                      down(2)       -- out of service
                    }
```

```
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION
           "Indicates the admin status of this mLDP FEC."

      DEFVAL { up }

      ::= { mplsMldpFecEntry 11 }

   mplsMldpFecOperStatus OBJECT-TYPE
      SYNTAX        INTEGER {
                      up(1),             -- ready to pass data
                      down(2)            -- out of service
                    }
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION
           "Indicates the actual operational status of this mLDP Fec."

      ::= { mplsMldpFecEntry 12 }

     mplsMldpFecMoFrr OBJECT-TYPE
         SYNTAX      INTEGER {
                       enable(1),
                       disable(2)
                     }
         MAX-ACCESS  read-only
         STATUS      current
         DESCRIPTION
            "This object provides whether MoFRR enabled for this mLDP FEC.
            on this mLDP FEC. As mentioned in the section 3.2 of [I-D.ietf-rtgwg
-mofrr],
            When this is enabled, then mLDP may select two upstream sessions,
            one is primary and other one is backup. The backup traffic is
            discarded when the primary upstream session is UP. When the
            primary upstream session goes down, the traffic from the backup
            upstream session will be forwarded to downstream.
             "

      ::= { mplsMldpFecEntry 13 }

   mplsMldpFecLsrState OBJECT-TYPE
      SYNTAX        INTEGER {
                       egress(1),
                       bud(2),
                       transit(3),
                       ingress(4)
                    }
      MAX-ACCESS    read-only
```

```
      STATUS        current
      DESCRIPTION
           "Indicates the role of FEC either egress, bud, transit or ingress"

      ::= { mplsMldpFecEntry 14 }

   mplsMldpFecUpTime OBJECT-TYPE
      SYNTAX        TimeStamp
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION
          "This values shows Fec UP time. This is time since mplsMldpFecOperStat
us is UP."

      ::= { mplsMldpFecEntry 15 }

   -- MPLS mLDP LSP Branch Traffic Stats Table.

   mplsMldpFecBranchStatsTable  OBJECT-TYPE
      SYNTAX        SEQUENCE OF MplsMldpFecBranchStatsEntry
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION
           "This table provides mLDP Fec branch MPLS Traffic Stats
            information."

      ::= { mplsMldpObjects 4 }

   mplsMldpFecBranchStatsEntry OBJECT-TYPE
      SYNTAX        MplsMldpFecBranchStatsEntry
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION
           "An entry in this table is created by the LSR for each
            downstream branch (out-segment) from this LSR for this mLDP
            LSP. Each downstream session may represent a single out-segment.

            Each entry in the table is indexed by the four identifiers
            of the mLDP LSP, and the out-segment that identifies the
            outgoing branch."

         INDEX       { mplsLdpEntityLdpId,
                       mplsLdpEntityIndex,
                       mplsLdpPeerLdpId,
                       mplsMldpFecBranchFecIndex,
                       mplsMldpFecBranchOutSegIndex
                     }

      ::= { mplsMldpFecBranchStatsTable 1 }
```

```
MplsMldpFecBranchStatsEntry ::= SEQUENCE {
      mplsMldpFecBranchFecIndex          MplsIndexType,
      mplsMldpFecBranchOutSegIndex       MplsIndexType,
      mplsMldpFecBranchStatsPackets      Counter64,
      mplsMldpFecBranchStatsBytes        Counter64,
      mplsMldpFecBranchStatsDiscontinuityTime TimeStamp
}

mplsMldpFecBranchFecIndex          OBJECT-TYPE
   SYNTAX        MplsIndexType
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
          "This index identifies the mLDP FEC entry in the
           mplsMldpFecTable.  This is same as mplsMldpFecIndex."

      ::= { mplsMldpFecBranchStatsEntry 1 }


mplsMldpFecBranchOutSegIndex          OBJECT-TYPE
   SYNTAX        MplsIndexType
   MAX-ACCESS    not-accessible
   STATUS        current
   DESCRIPTION
        "This object identifies an outgoing branch from this mLDP LSP
         Its value is unique within the context of the mLDP LSP.

          This contains the same value as the mplsOutSegmentIndex in the
          MPLS-LSR-STD-MIBs mplsOutSegmentTable."

      ::= { mplsMldpFecBranchStatsEntry 2 }

mplsMldpFecBranchStatsPackets OBJECT-TYPE
   SYNTAX        Counter64
   MAX-ACCESS    read-only
   STATUS        current
   DESCRIPTION
        "This object represent the 64-bit value, which gives the number
         of packets forwarded by the mLDP LSP onto this branch.
         This object should be read in conjunction with
         mplsMldpFecBranchStatsDiscontinuityTime."

   ::= { mplsMldpFecBranchStatsEntry 3 }

mplsMldpFecBranchStatsBytes OBJECT-TYPE
   SYNTAX        Counter64
   MAX-ACCESS    read-only
   STATUS        current
```

```
      DESCRIPTION
           "This object represent the 64-bit value, which gives the number
            of bytes forwarded by the mLDP LSP onto this branch.
            This object should be read in conjunction with
            mplsMldpFecBranchStatsDiscontinuityTime."

      ::= { mplsMldpFecBranchStatsEntry 4 }

   mplsMldpFecBranchStatsDiscontinuityTime OBJECT-TYPE
      SYNTAX      TimeStamp
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
           "The value of sysUpTime on the most recent occasion at which
            any one or more of this rows Counter32 or Counter64 objects
            experienced a discontinuity. If no such discontinuity has
            occurred since the last re-initialization of the local
            management subsystem, then this object contains a zero
            value."
      ::= { mplsMldpFecBranchStatsEntry 5 }

   -- End of mplsMldpFecBranchStatsTable


   -- MPLS mLDP LSP Upstream Session Table.

   mplsMldpFecUpstreamSessTable  OBJECT-TYPE
      SYNTAX        SEQUENCE OF MplsMldpFecUpstreamSessEntry
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION
           "This table provides mLDP Fec upstream Session information."

      ::= { mplsMldpObjects 5 }

   mplsMldpFecUpstreamSessEntry OBJECT-TYPE
      SYNTAX        MplsMldpFecUpstreamSessEntry
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION
           "An entry in this table is created by the LSR for each
            upstream session (in-segment) from this LSR for this mLDP
            LSP. Each upstream session may represent a single in-segment.

            Each entry in the table is indexed by the four identifiers
            of the mLDP LSP, and the in-segment that identifies the
            incoming traffic."
```

```
        INDEX       { mplsLdpEntityLdpId,
                      mplsLdpEntityIndex,
                      mplsLdpPeerLdpId,
                      mplsMldpFecUpstreamSessFecIndex,
                      mplsMldpFecUpstreamSessInSegIndex
                    }

      ::= { mplsMldpFecUpstreamSessTable 1 }

   MplsmLdpFecUpstreamSessEntry ::= SEQUENCE {
        mplsMldpFecUpstreamSessFecIndex     MplsIndexType,
        mplsMldpFecUpstreamSessInSegIndex   MplsIndexType,
        mplsMldpFecUpstreamSessPrimary      INTEGER,
        mplsMldpFecUpstreamSessActive       INTEGER,
        mplsMldpFecUpstreamSessPackets      Counter64,
        mplsMldpFecUpstreamSessBytes        Counter64,
        mplsMldpFecUpstreamSessDiscontinuityTime TimeStamp
   }

   mplsMldpFecUpstreamSessFecIndex          OBJECT-TYPE
      SYNTAX        MplsIndexType
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION
             "This index identifies the mLDP FEC entry in the
              mplsMldpFecTable."

        ::= { mplsMldpFecUpstreamSessEntry 1 }


   mplsMldpFecUpstreamSessInSegIndex          OBJECT-TYPE
      SYNTAX        MplsIndexType
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION
           "This object identifies an upstream session from this mLDP LSP
            Its value is unique within the context of the mLDP LSP.

             This contains the same value as the mplsInSegmentIndex in the
             MPLS-LSR-STD-MIBs mplsInSegmentTable."

        ::= { mplsMldpFecUpstreamSessEntry 2 }

    mplsMldpFecUpstreamSessPrimary  OBJECT-TYPE
        SYNTAX      INTEGER {
                       primary(1),
                       backup(2)
                    }
```

```
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This indicated wether the received traffic from upstream is
            primary or backup. This is valid only if the MoFRR
            (mplsMldpFecMoFrr) is enabled on this FEC."

        ::= { mplsMldpFecUpstreamSessEntry 3 }

    mplsMldpFecUpstreamSessActive  OBJECT-TYPE
        SYNTAX      INTEGER {
                       active(1),
                       inactive(2)
                    }
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
        "This indicates whether the upstream session is active, means the
        LSR programmed the forwarding engine to receive the traffic from
        this upstream session. This will be Inactive if the LSR is wating
        for MBB Ack."

        ::= { mplsMldpFecUpstreamSessEntry 4 }

   mplsMldpFecUpstreamSessPackets      OBJECT-TYPE
      SYNTAX        Counter64
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION
           "This object represent the 64-bit value, which gives the number
            of packets received by the mLDP LSP from this upstream
            session.  This object should be read in conjunction with
            mplsMldpFecUpstreamSessDiscontinuityTime."

      ::= { mplsMldpFecUpstreamSessEntry 5 }

   mplsMldpFecUpstreamSessBytes      OBJECT-TYPE
      SYNTAX        Counter64
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION
           "This object represent the 64-bit value, which gives the number
            of bytes received by the mLDP LSP from this upstream
            session.  This object should be read in conjunction with
            mplsMldpFecUpstreamSessDiscontinuityTime."

      ::= { mplsMldpFecUpstreamSessEntry 6 }
```

```
   mplsMldpFecUpstreamSessDiscontinuityTime OBJECT-TYPE
      SYNTAX      TimeStamp
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
           "The value of sysUpTime on the most recent occasion at which
            any one or more of this rows Counter32 or Counter64 objects
            experienced a discontinuity. If no such discontinuity has
            occurred since the last re-initialization of the local
            management subsystem, then this object contains a zero
            value."
      ::= { mplsMldpFecUpstreamSessEntry 7 }

   -- End of mplsMldpFecBranchStatsTable


   -- MPLS mLDP Interface Traffic Stats Table.

   mplsMldpInterfaceStatsTable  OBJECT-TYPE
      SYNTAX        SEQUENCE OF MplsMldpInterfaceStatsEntry
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION
           "This table provides mLDP  Traffic Stats on specified interface."

      ::= { mplsMldpObjects 6 }

   mplsMldpInterfaceStatsEntry OBJECT-TYPE
      SYNTAX        MplsMldpInterfaceStatsEntry
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION
           "An entry in this table is created by the LSR for each
            downstream branch (out-segment) from this LSR for this mLDP
            LSP. Each downstream session may represent a single out-segment.

            Each entry in the table is indexed by the four identifiers
            of the mLDP LSP, and the out-segment that identifies the
            outgoing branch."

         INDEX       { mplsMldpInterfaceIndex
                     }

      ::= { mplsMldpInterfaceStatsTable 1 }

   MplsMldpInterfaceStatsEntry ::= SEQUENCE {
         mplsMldpInterfaceIndex                InterfaceIndex,
         mplsMldpInterfaceStatsSentPackets     Counter64,
```

```
        mplsMldpInterfaceStatsSentBytes        Counter64,
        mplsMldpInterfaceStatsRecvPackets      Counter64,
        mplsMldpInterfaceStatsRecvBytes        Counter64
   }

   mplsMldpInterfaceIndex          OBJECT-TYPE
      SYNTAX        InterfaceIndex
      MAX-ACCESS    not-accessible
      STATUS        current
      DESCRIPTION
             "This index identifies the specific interface. "

        ::= { mplsMldpInterfaceStatsEntry 1 }

   mplsMldpInterfaceStatsSentPackets OBJECT-TYPE
      SYNTAX        Counter64
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION
            "This is 64 bit value, which gives the number of packets
            forwarded by all mLDP LSPs onto this interface."

      ::= { mplsMldpInterfaceStatsEntry 2 }

   mplsMldpInterfaceStatsSentBytes OBJECT-TYPE
      SYNTAX        Counter64
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION
            "This is 64 bit value, which gives the number of bytes
            forwarded by all mLDP LSPs onto this interface."

      ::= { mplsMldpInterfaceStatsEntry 3 }


  mplsMldpInterfaceRecvPackets OBJECT-TYPE
      SYNTAX        Counter64
      MAX-ACCESS    read-only
      STATUS        current
      DESCRIPTION
            "This is 64 bit value, which gives the number of packets
            received by all mLDP LSPs from this interface."

      ::= { mplsMldpInterfaceStatsEntry 4 }

   mplsMldpInterfaceStatsRecvBytes OBJECT-TYPE
      SYNTAX        Counter64
      MAX-ACCESS    read-only
```

```
      STATUS        current
      DESCRIPTION
            "This is 64 bit value, which gives the number of bytes
            received by all mLDP LSPs from this interface."

      ::= { mplsMldpInterfaceStatsEntry 5 }

   -- End of mplsMldpInterfaceStatsTable

   -- Notifications.

   mplsMldpFecUp NOTIFICATION-TYPE
      OBJECTS     {
         mplsMldpFecAdminStatus,
         mplsMldpFecOperStatus
      }
      STATUS      current
      DESCRIPTION
           "This notification is generated when a mplsMldpFecOperStatus
            object changes from down to up."

      ::= { mplsMldpNotifications 1 }

   mplsMldpFecDown NOTIFICATION-TYPE
      OBJECTS     {
         mplsMldpFecAdminStatus,
         mplsMldpFecOperStatus
      }
      STATUS      current
      DESCRIPTION
           "This notification is generated when a mplsMldpFecOperStatus
            object changes from up to down."

      ::= { mplsMldpNotifications 2 }


   mplsMldpMoFrrStatusChange NOTIFICATION-TYPE
      OBJECTS     {
         mplsMldpFecUpstreamSessPrimary,
      }
      STATUS      current
      DESCRIPTION
           "This notification is generated when a mplsMldpFecUpstreamSessPrimary
            object changes from primary to backup and vice versa."

            ::= { mplsMldpNotifications 3 }

   -- End of notifications.
```

12. Security Considerations

This MIB module is useful for the configuration of certain objects and monitoring of mLDP LSPs.

There are no management objects defined in this MIB module that have a MAX-ACCESS clause of read-write and/or read-create. So, if this MIB module is implemented correctly, then there is no risk that an intruder can alter or create any management objects of this MIB module via direct SNMP SET operations.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

- o mplsMldpFecTable
- o mplsLdpPeerCapabilityTable
- o mplsMldpSessionStatsTable
- o mplsMldpFecBranchStatsTable
- o mplsMldpFecUpstreamSessTable
- o mplsMldpInterfaceStatsTable
- o mplsMldpNumFecs
- o mplsMldpNumFecsActive
- o mplsMldpMbbTime

Above listed tables and objects show information about the mLDP LSPs, its route through the network, and its traffic statistics. Knowledge of this information could be used to compromise the network, or simply to breach confidentiality. If an Administrator does not want to reveal this information, these tables and objects should be considered sensitive/vulnerable.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), there is no control as to who on the secure network is allowed to

access and GET/SET (read/change/create/delete) the objects in this MIB module.

Implementations SHOULD provide the security features described by the SNMPv3 framework (see [RFC3410]), and implementations claiming compliance to the SNMPv3 standard MUST include full support for authentication and privacy via the User-based Security Model (USM) [RFC3414] with the AES cipher algorithm [RFC3826]. Implementations MAY also provide support for the Transport Security Model (TSM) [RFC5591] in combination with a secure transport such as SSH [RFC5592] or TLS/DTLS [RFC6353].

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

13. IANA Considerations

This is new MPLS MIB module, contained in this document and IANA is requested to assign an oid (mplsStdMIB YYY) under the mplsStdMIB subtree to the MPLS-MLDP-STD-MIB module specified in this document.

14. Acknowledgments

The authors wish to thank Santosh Esale, Alia Atlas and Martin Ehlers for doing the detailed review. Thanks to Adrian Farrel and Raveendra Torvi for their input to this work and for many helpful suggestions.

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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, DOI 10.17487/RFC2578, April 1999, <http://www.rfc-editor.org/info/rfc2578>.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, DOI 10.17487/RFC2579, April 1999, <http://www.rfc-editor.org/info/rfc2579>.

[RFC2580] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Conformance Statements for SMIv2", STD 58, RFC 2580, DOI 10.17487/RFC2580, April 1999, <http://www.rfc-editor.org/info/rfc2580>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <http://www.rfc-editor.org/info/rfc3031>.

[RFC3811] Nadeau, T., Ed. and J. Cucchiara, Ed., "Definitions of Textual Conventions (TCs) for Multiprotocol Label Switching (MPLS) Management", RFC 3811, DOI 10.17487/RFC3811, June 2004, <http://www.rfc-editor.org/info/rfc3811>.

[RFC3813] Srinivasan, C., Viswanathan, A., and T. Nadeau, "Multiprotocol Label Switching (MPLS) Label Switching Router (LSR) Management Information Base (MIB)", RFC 3813, DOI 10.17487/RFC3813, June 2004, <http://www.rfc-editor.org/info/rfc3813>.

[RFC3815] Cucchiara, J., Sjostrand, H., and J. Luciani, "Definitions of Managed Objects for the Multiprotocol Label Switching (MPLS), Label Distribution Protocol (LDP)", RFC 3815, DOI 10.17487/RFC3815, June 2004, <http://www.rfc-editor.org/info/rfc3815>.

[RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <http://www.rfc-editor.org/info/rfc5036>.

[RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL. Le Roux, "LDP Capabilities", RFC 5561, DOI 10.17487/RFC5561, July 2009, <http://www.rfc-editor.org/info/rfc5561>.

[RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <http://www.rfc-editor.org/info/rfc6388>.

[RFC6826] Wijnands, IJ., Ed., Eckert, T., Leymann, N., and M. Napierala, "Multipoint LDP In-Band Signaling for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6826, DOI 10.17487/RFC6826, January 2013, <http://www.rfc-editor.org/info/rfc6826>.

[RFC7060] Napierala, M., Rosen, E., and IJ. Wijnands, "Using LDP Multipoint Extensions on Targeted LDP Sessions", RFC 7060, DOI 10.17487/RFC7060, November 2013, <http://www.rfc-editor.org/info/rfc7060>.

[RFC6389] Aggarwal, R. and JL. Le Roux, "MPLS Upstream Label Assignment for LDP", RFC 6389, DOI 10.17487/RFC6389, November 2011, <http://www.rfc-editor.org/info/rfc6389>.

15.2. Informative References

[RFC2223] Postel, J. and J. Reynolds, "Instructions to RFC Authors", RFC 2223, DOI 10.17487/RFC2223, October 1997, <http://www.rfc-editor.org/info/rfc2223>.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, DOI 10.17487/RFC3410, December 2002, <http://www.rfc-editor.org/info/rfc3410>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[RFC4181] Heard, C., Ed., "Guidelines for Authors and Reviewers of MIB Documents", BCP 111, RFC 4181, DOI 10.17487/RFC4181, September 2005, <http://www.rfc-editor.org/info/rfc4181>.

[RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, DOI 10.17487/RFC4001, February 2005, <http://www.rfc-editor.org/info/rfc4001>.

[I-D.ietf-rtgwg-mofrr] Karan, A., Filsfils, C., Wijnands, I., and B. Decraene, "Multicast only Fast Re-Route", draft-ietf-rtgwg-mofrr-08 (work in progress), May 2015.

[I-D.ietf-mpls-mldp-node-protection]
Wijnands, I., Raza, K., Atlas, A., Tantsura, J., and Q. Zhao, "mLDP Node Protection", draft-ietf-mpls-mldp-node-protection-08 (work in progress), September 2015.

Appendix A. Change Log

Appendix B. Open Issues

Authors' Addresses

Kishore Tiruveedhula (editor)
Juniper Networks
10 Technology Park Drive
Westford MA 01886
USA

Phone: +1 9785898861
EMail: kishoret@juniper.net

Uwe Joorde
Deutsche Telekom
Dahlweg 100
Munster 48153
Germany

EMail: Uwe.Joorde@telekom.de

Arvind Venkateswaran
Cisco Systems
510 McCarthy Blvd
Milpitas CA 95035
USA

EMail: arvvenka@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 13, 2019

L. Zheng
G. Zheng
Huawei Technologies
G. Mirsky
ZTE Corp.
R. Rahman
F. Iqbal
Cisco Systems

January 9, 2019

YANG Data Model for LSP-Ping
draft-zheng-mpls-lsp-ping-yang-cfg-10

Abstract

When an LSP fails to deliver user traffic, the failure cannot always be detected by the MPLS control plane. RFC 8029 defines a mechanism that would enable users to detect such failure and to isolate faults. YANG, defined in RFC 6020 and RFC 7950, is a data modeling language used to specify the contents of a conceptual data stores that allows networked devices to be managed using NETCONF, as specified in RFC 6241. This document defines a YANG data model that can be used to configure and manage LSP-Ping.

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 July 13, 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

When an LSP fails to deliver user traffic, the failure cannot always be detected by the MPLS control plane. [RFC8029] defines a mechanism that would enable users to detect such failure and to isolate faults. YANG, defined in [RFC6020] and [RFC7950], is a data modeling language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. This document defines a YANG data model that can be used to configure and manage LSP-Ping [RFC8029].

The rest of this document is organized as follows. Section 2 presents the scope of this document. Section 3 provides the design of the LSP-Ping configuration data model in details by containers. Section 4 presents the complete data hierarchy of LSP-Ping YANG model. Section 5 discusses the interaction between LSP-Ping data model and other MPLS tools data models. Section 6 specifies the YANG module and section 7 lists examples which conform to the YANG module specified in this document. Finally, security considerations are discussed in Section 8.

This version of the LSP Ping data model conforms to the Network Management Datastore Architecture (NMDA) [RFC8342].

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.

1.2. Support of Long Running Command with NETCONF

LSP Ping is one of the examples of what can be described as "long-running operation". Unlike most of the configuration operations that result in single response execution of an LSP Ping triggers multiple responses from a node under control. The question of implementing the long-running operation in NETCONF is still open and possible solutions being discussed:

1. Consecutive Remote Processing Calls (RPC) to poll for results.

2. Model presented in [RFC4560].

3. The one outlined in [I-D.mahesh-netconf-persistent].

The problem of long-running operation as well can be considered as a case of controlling and obtaining results from a Measurement Agent (MA) as defined in [RFC7594].

2. Scope

The fundamental mechanism of LSP-Ping is defined in [RFC8029]. Extensions of LSP-Ping has been developed over the years. There are extensions for performing LSP ping, for example, over P2MP MPLS LSPs [RFC6425] or for Segment Routing IGP Prefix and Adjacency SIDs with an MPLS data plane [RFC8287]. These extensions will be considered in a later update of this document.

3. Design of the Data Model

This YANG data model is defined to be used to configure and manage LSP-Ping and it provides the following features:

1. The configuration of control information of an LSP-Ping test.

2. The configuration of schedule parameters of an LSP-Ping test.

3. Display of result information of an LSP-Ping test.

The top-level container lsp-pings holds the configuration of the control information, schedule parameters and result information for multiple instances of LSP-Ping test.

3.1. The Configuration of Control Information

Container lsp-pings:lsp-ping:control-parameters defines the configuration parameters which control an LSP-Ping test. Examples are the target-fec-type/target-fec of the echo request packet and the reply mode of the echo reply packet. Values of some parameters may be auto-assigned by the system, but in several cases, there is a requirement for configuration of these parameters. Examples of such parameters are source address and outgoing interface.

The data hierarchy for control information configuration is presented below:

```
module: ietf-lsp-ping
   +--rw lsp-pings
      +--rw lsp-ping* [lsp-ping-name]
         +--rw lsp-ping-name          string
         +--rw control-parameters
         |  +--rw target-fec-type?       target-fec-type
         |  +--rw (target-fec)?
         |  |  +--:(ip-prefix)
         |  |  |  +--rw ip-address?            inet:ip-address
         |  |  +--:(bgp)
         |  |  |  +--rw bgp?                   inet:ip-address
         |  |  +--:(rsvp)
         |  |  |  +--rw tunnel-interface?      string
         |  |  +--:(vpn)
         |  |  |  +--rw vrf-name?              uint32
         |  |  |  +--rw vpn-ip-address?        inet:ip-address
         |  |  +--:(pw)
         |  |  |  +--rw vcid?                  uint32
         |  |  +--:(vpls)
         |  |     +--rw vsi-name?              string
         |  +--rw traffic-class?         uint8
         |  +--rw reply-mode?            reply-mode
         |  +--rw timeout?               uint32
         |  +--rw timeout-units?         units
         |  +--rw interval?              uint32
         |  +--rw interval-units?        units
         |  +--rw probe-count?           uint32
         |  +--rw data-size?             uint32
         |  +--rw data-fill?             string
         |  +--rw description?           string
         |  +--rw source-address?        inet:ip-address
         |  +--rw ttl?                   uint8
         |  +--rw (outbound)?
         |     +--:(interface)
         |     |  +--rw interface-name?        string
         |     +--:(nexthop)
         |        +--rw nexthop?               inet:ip-address
```

3.2. The Configuration of Schedule Parameters

Container lsp-pings:lsp-ping:scheduling-parameters defines the schedule parameters of an LSP-Ping test, which describes when to start and when to end the test. Four start modes and three end modes are defined respectively. To be noted that, the configuration of "interval" and "probe-count" parameter defined in container lsp-pings:lsp-ping:control-parameters could also determine when the test ends implicitly. All these three parameters are optional.If the user

does not configure either "interval" or "probe-count" parameter, then the default values will be used by the system. If the user configures "end-test", then the actual end time of the LSP-Ping test is the smaller one between the configuration value of "end-test" and the time implicitly determined by the configuration value of "interval"/"probe-count".

The data hierarchy for schedule information configuration is presented below:

```
module: ietf-lsp-ping
   +--rw lsp-pings
      +--rw lsp-ping* [lsp-ping-name]
         +--rw lsp-ping-name          string
         +--rw control-parameters
         ...
         +--rw scheduling-parameters
         |  +--rw (start-test)?
         |  |  +--:(now)
         |  |  |  +--rw start-test-now?           empty
         |  |  +--:(at)
         |  |  |  +--rw start-test-at?            yang:date-and-time
         |  |  +--:(delay)
         |  |  |  +--rw start-test-delay?         uint32
         |  |  |  +--rw start-test-delay-units?   units
         |  |  +--:(daily)
         |  |     +--rw start-test-daily?         yang:date-and-time
         |  +--rw (end-test)?
         |     +--:(at)
         |     |  +--rw end-test-at?              yang:date-and-time
         |     +--:(delay)
         |     |  +--rw end-test-delay?           uint32
         |     |  +--rw end-test-delay-units?     units
         |     +--:(lifetime)
         |        +--rw end-test-lifetime?        uint32
         |        +--rw lifetime-units?           units
```

3.3. Display of Result Information

Container lsp-pings:lsp-ping:result-info shows the result of the current LSP-Ping test. Both the statistical result e.g. min-rtt, max-rtt, and per test probe result e.g. return code, return subcode, are shown.

The data hierarchy for display of result information is presented below:

```
module: ietf-lsp-ping
   +--rw lsp-pings
      +--rw lsp-ping* [lsp-ping-name]
         +--rw lsp-ping-name          string
         +--rw control-parameters
         ...
         +--rw scheduling-parameters
         ...
         +--ro result-info
            +--ro operational-status?    operational-status
            +--ro source-address?        inet:ip-address
            +--ro target-fec-type?       target-fec-type
            +--ro (target-fec)?
            |  +--:(ip-prefix)
            |  |  +--ro ip-address?            inet:ip-address
            |  +--:(bgp)
            |  |  +--ro bgp?                   inet:ip-address
            |  +--:(rsvp)
            |  |  +--ro tunnel-interface?      string
            |  +--:(vpn)
            |  |  +--ro vrf-name?              uint32
            |  |  +--ro vpn-ip-address?        inet:ip-address
            |  +--:(pw)
            |  |  +--ro vcid?                  uint32
            |  +--:(vpls)
            |     +--ro vsi-name?              string
            +--ro min-rtt?               uint32
            +--ro max-rtt?               uint32
            +--ro average-rtt?           uint32
            +--ro probe-responses?       uint32
            +--ro sent-probes?           uint32
            +--ro sum-of-squares?        uint32
            +--ro last-good-probe?       yang:date-and-time
            +--ro probe-results
               +--ro probe-result* [probe-index]
                  +--ro probe-index        uint32
                  +--ro return-code?       uint8
                  +--ro return-sub-code?   uint8
                  +--ro rtt?               uint32
                  +--ro result-type?       result-type
```

4. Data Hierarchy

The complete data hierarchy of LSP-Ping YANG model is presented below.

```
module: ietf-lsp-ping
```

```
      +--rw lsp-pings
         +--rw lsp-ping* [lsp-ping-name]
            +--rw lsp-ping-name          string
            +--rw control-parameters
            |  +--rw target-fec-type?       target-fec-type
            |  +--rw (target-fec)?
            |  |  +--:(ip-prefix)
            |  |  |  +--rw ip-address?            inet:ip-address
            |  |  +--:(bgp)
            |  |  |  +--rw bgp?                   inet:ip-address
            |  |  +--:(rsvp)
            |  |  |  +--rw tunnel-interface?      string
            |  |  +--:(vpn)
            |  |  |  +--rw vrf-name?              uint32
            |  |  |  +--rw vpn-ip-address?        inet:ip-address
            |  |  +--:(pw)
            |  |  |  +--rw vcid?                  uint32
            |  |  +--:(vpls)
            |  |     +--rw vsi-name?              string
            |  +--rw traffic-class?         uint8
            |  +--rw reply-mode?            reply-mode
            |  +--rw timeout?               uint32
            |  +--rw timeout-units?         units
            |  +--rw interval?              uint32
            |  +--rw interval-units?        units
            |  +--rw probe-count?           uint32
            |  +--rw data-size?             uint32
            |  +--rw data-fill?             string
            |  +--rw description?           string
            |  +--rw source-address?        inet:ip-address
            |  +--rw ttl?                   uint8
            |  +--rw (outbound)?
            |     +--:(interface)
            |     |  +--rw interface-name?        string
            |     +--:(nexthop)
            |        +--rw nexthop?               inet:ip-address
            +--rw scheduling-parameters
            |  +--rw (start-test)?
            |  |  +--:(now)
            |  |  |  +--rw start-test-now?           empty
            |  |  +--:(at)
            |  |  |  +--rw start-test-at?            yang:date-and-time
            |  |  +--:(delay)
            |  |  |  +--rw start-test-delay?         uint32
            |  |  |  +--rw start-test-delay-units?   units
            |  |  +--:(daily)
            |  |     +--rw start-test-daily?         yang:date-and-time
            |  +--rw (end-test)?
```

```
             |      +--:(at)
             |      |  +--rw end-test-at?              yang:date-and-time
             |      +--:(delay)
             |      |  +--rw end-test-delay?           uint32
             |      |  +--rw end-test-delay-units?     units
             |      +--:(lifetime)
             |         +--rw end-test-lifetime?        uint32
             |         +--rw lifetime-units?           units
            +--ro result-info
               +--ro operational-status?    operational-status
               +--ro source-address?        inet:ip-address
               +--ro target-fec-type?       target-fec-type
               +--ro (target-fec)?
               |  +--:(ip-prefix)
               |  |  +--ro ip-address?            inet:ip-address
               |  +--:(bgp)
               |  |  +--ro bgp?                   inet:ip-address
               |  +--:(rsvp)
               |  |  +--ro tunnel-interface?      string
               |  +--:(vpn)
               |  |  +--ro vrf-name?              uint32
               |  |  +--ro vpn-ip-address?        inet:ip-address
               |  +--:(pw)
               |  |  +--ro vcid?                  uint32
               |  +--:(vpls)
               |     +--ro vsi-name?              string
               +--ro min-rtt?               uint32
               +--ro max-rtt?               uint32
               +--ro average-rtt?           uint32
               +--ro probe-responses?       uint32
               +--ro sent-probes?           uint32
               +--ro sum-of-squares?        uint32
               +--ro last-good-probe?       yang:date-and-time
               +--ro probe-results
                  +--ro probe-result* [probe-index]
                     +--ro probe-index        uint32
                     +--ro return-code?       uint8
                     +--ro return-sub-code?   uint8
                     +--ro rtt?               uint32
                     +--ro result-type?       result-type
```

5. Interaction with other MPLS OAM Tools Models

TBA

6. LSP-Ping YANG Module

```
<CODE BEGINS> file "ietf-lsp-ping@2018-11-29.yang"
module ietf-lsp-ping {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-lsp-ping";
  //namespace need to be assigned by IANA
  prefix "lsp-ping";

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

  organization "IETF Multiprotocol Label Switching Working Group";

    contact
    "WG Web: http://tools.ietf.org/wg/mpls/
     WG List: mpls@ietf.org

     Editor: Greg Mirsky
          gregimirsky@gmail.com
     Editor: Lianshu Zheng
          vero.zheng@huawei.com
     Editor: Guangying Zheng
          zhengguangying@huawei.com
     Editor: Reshad Rahman
          rrahman@cisco.com
     Editor: Faisal Iqbal
          faiqbal@cisco.com";

    description
    "This YANG module specifies a vendor-independent model
     for the LSP Ping.

     This YANG data model is defined to be used to configure and manage
     LSP-Ping and it provides the following features:
     1.  The configuration of control information of an LSP-Ping test.
     2.  The configuration of schedule parameters of an LSP-Ping test.
     3.  Display of result information of an LSP-Ping test.

        Copyright (c) 2018 IETF Trust and the persons identified as
        the document authors.  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.";

    reference "draft-zheng-mpls-lsp-ping-yang-cfg";

    revision "2018-11-29" {
      description
        "10 version, refine the target fec type,
         as per RFC8029 and update Security Considerations section.";
    reference "draft-zheng-mpls-lsp-ping-yang-cfg";
  }

  typedef target-fec-type {
    type enumeration {
      enum ip-prefix {
        value "0";
        description "IPv4/IPv6 prefix";
      }
      enum bgp {
        value "1";
        description "BGP IPv4/IPv6 prefix";
      }
      enum rsvp {
        value "2";
        description "Tunnel interface";
      }
      enum vpn {
        value "3";
        description "VPN IPv4/IPv6 prefix";
      }
      enum pw {
        value "4";
        description "FEC 128 pseudowire IPv4/IPv6";
      }
      enum vpls {
        value "5";
        description "FEC 129 pseudowire IPv4/IPv6";
      }
    }
    description "Target FEC type, as defined in RFC 8029";
  }
```

```
  typedef reply-mode {
    type enumeration {
      enum do-not-reply {
        value "1";
        description "Do not reply";
      }
      enum reply-via-udp {
        value "2";
        description "Reply via an IPv4/IPv6 UDP packet";
      }
      enum reply-via-udp-router-alert {
        value "3";
        description
          "Reply via an IPv4/IPv6 UDP packet with Router Alert";
      }
      enum reply-via-control-channel {
        value "4";
        description
          "Reply via application level control channel";
      }
    }
    description "Reply mode";
  }

  typedef units  {
    type enumeration {
      enum seconds {
        description "Seconds";
      }
      enum milliseconds {
        description "Milliseconds";
      }
      enum microseconds {
        description "Microseconds";
      }
      enum nanoseconds {
        description "Nanoseconds";
      }
    }
    description "Time units";
  }

  typedef operational-status {
    type enumeration {
      enum enabled {
        value "1";
        description "The Test is active";
      }
```

```
      enum disabled {
        value "2";
        description "The test has stopped";
      }
      enum completed {
        value "3";
        description "The test is completed";
      }
    }
    description "Operational state of an LSP Ping test";
  }

  typedef result-type {
    type enumeration {
      enum success {
        value "1";
        description "The test probe is successful";
      }
      enum fail {
        value "2";
        description "The test probe has failed";
      }
      enum timeout {
        value "3";
        description "The time of the test probe has expired";
      }
    }
    description "Result of each LSP Ping test probe";
  }

  container lsp-pings {
    description "Multi-instance of the LSP Ping test";
    list lsp-ping {
      key "lsp-ping-name";
      description "LSP Ping test";
      leaf lsp-ping-name {
        type string {
            length "1..31";
        }
        mandatory "true";
        description "LSP Ping test name";
      }
      container control-parameters {
        description "Control information of the LSP Ping test";
        leaf target-fec-type {
          type target-fec-type;
          description "Specifies the address type of the Target FEC";
        }
```

```
        choice target-fec {
          case ip-prefix {
            leaf ip-address {
              type inet:ip-address;
              description "IPv4/IPv6 Prefix";
            }
          }
          case bgp {
            leaf bgp {
              type inet:ip-address;
              description "BGP IPv4/IPv6 Prefix";
            }
          }
          case rsvp {
            leaf tunnel-interface {
              type string;
              description "Tunnel interface";
            }
          }
          case vpn {
            leaf vrf-name {
              type uint32;
              description "Layer3 VPN Name";
            }
            leaf vpn-ip-address {
              type inet:ip-address;
              description "Layer3 VPN IPv4 Prefix";
            }
          }
          case pw {
            leaf vcid {
              type uint32;
              description "VC ID";
            }
          }
          case vpls {
            leaf vsi-name {
              type string;
              description "VPLS VSI";
            }
          }
          description "Specifies the type of the Target FEC";
        }
        leaf traffic-class {
          type uint8;
          description "Specifies the Traffic Class";
        }
        leaf reply-mode {
```

```
          type reply-mode;
          description "Specifies the Reply Mode";
        }
        leaf timeout {
          type uint32;
          description
            "Specifies the time-out value for a LSP Ping operation.";
        }
        leaf timeout-units {
          type units;
          description "Time-out units";
        }
        leaf interval {
          type uint32;
          default 1;
          description
            "Specifies the interval between transmissions
             of LSP Ping echo request packets (probes)
             as part of the LSP Ping test.";
        }
        leaf interval-units {
          type units;
          default seconds;
          description "Interval units";
        }
        leaf probe-count {
          type uint32;
          default 5;
          description
            "Specifies the number of probes sent in the LSP Ping test.";
        }
        leaf data-size {
          type uint32;
          description
            "Specifies the size of the data portion to
             be transmitted in an LSP Ping operation, in octets.";
        }
        leaf data-fill {
          type string{
              length "0..1564";
          }
          description
            "Used together with the corresponding
             data-size value to determine how to fill the data
             portion of a probe packet.";
        }
        leaf description {
          type string{
```

```
              length "1..31";
          }
          description "A descriptive name of the LSP Ping test";
        }
        leaf source-address {
          type inet:ip-address;
          description "Specifies the source address";
        }
        leaf ttl {
          type uint8;
          default 255;
          description "Time to live";
        }
        choice outbound {
          case interface {
            leaf interface-name{
              type string{
                length "1..255";
              }
              description "Specifies the outgoing interface";
            }
          }
          case nexthop{
            leaf nexthop {
              type inet:ip-address;
              description "Specifies the nexthop";
            }
          }
          description "Specifies the out interface or nexthop";
        }
      }

      container scheduling-parameters {
        description "LSP Ping test schedule parameter";
        choice start-test{
          case now {
            leaf start-test-now {
              type empty;
              description "Start test now";
            }
          }
          case at {
            leaf start-test-at {
              type yang:date-and-time;
              description "Start test at a specific time";
            }
          }
          case delay {
```

```
            leaf start-test-delay {
              type uint32;
              description "Start after a specific delay";
            }
            leaf start-test-delay-units {
              type units;
              default seconds;
              description "Delay units";
            }
          }
          case daily {
            leaf start-test-daily {
              type yang:date-and-time;
              description "Start test daily";
            }
          }
          description
            "Specifies when the test begins to start,
             include 4 schedule method: start now(1), start at(2),
             start delay(3), start daily(4).";
        }

        choice end-test{
          case at {
            leaf end-test-at{
              type yang:date-and-time;
              description "End test at a specific time";
            }
          }
          case delay {
            leaf end-test-delay {
              type uint32;
              description "End after a specific delay";
            }
            leaf end-test-delay-units {
              type units;
              default seconds;
              description "Delay units";
            }
          }
          case lifetime {
            leaf end-test-lifetime {
              type uint32;
              description "Set the test lifetime";
            }
            leaf lifetime-units {
              type units;
              default seconds;
```

```
              description "Lifetime units";
            }
          }
          description
            "Specifies when the test ends, include 3
             schedule method: end at(1), end delay(2),
             end lifetime(3).";
        }
      }

      container result-info {
        config "false";
        description "LSP Ping test result information";
        leaf operational-status {
          type operational-status;
          description "Operational state of a LSP Ping test";
        }
        leaf source-address {
          type inet:ip-address;
          description "The source address of the test";
        }
        leaf target-fec-type {
          type target-fec-type;
          description "The Target FEC address type";
        }
        choice target-fec {
          case ip-prefix {
            leaf ip-address {
              type inet:ip-address;
              description "IPv4/IPv6 Prefix";
            }
          }
          case bgp {
            leaf bgp {
              type inet:ip-address;
              description "BGP IPv4/IPv6 Prefix";
            }
          }
          case rsvp {
            leaf tunnel-interface {
              type string;
              description "Tunnel interface";
            }
          }
          case vpn {
            leaf vrf-name {
              type uint32;
              description "Layer3 VPN Name";
```

```
            }
            leaf vpn-ip-address {
              type inet:ip-address;
              description "Layer3 VPN IPv4 Prefix";
            }
          }
          case pw {
            leaf vcid {
              type uint32;
              description "VC ID";
            }
          }
          case vpls {
            leaf vsi-name {
              type string;
              description "VPLS VSI";
            }
          }
          description "The Target FEC address";
        }
        leaf min-rtt {
          type uint32;
          description
            "The minimum LSP Ping round-trip-time (RTT)
             received measured in usec.";
        }
        leaf max-rtt {
          type uint32;
          description
            "The maximum LSP Ping round-trip-time (RTT)
             received measured in usec.";
        }
        leaf average-rtt {
          type uint32;
          description
            "The current average LSP Ping round-trip-time
             (RTT) measured in usec.";
        }
        leaf probe-responses {
          type uint32;
          description
            "Number of responses received for the
             corresponding LSP Ping test.";
        }
        leaf sent-probes {
          type uint32;
          description
            "Number of probes sent for the
```

```
             corresponding LSP Ping test.";
        }
        leaf sum-of-squares {
          type uint32;
          description
            "The sum of the squares of RTT,
             calculated as the sum of the squared
             differences between each RTT and the overall
             mean RTT, for all replies received.";
        }
        leaf last-good-probe {
          type yang:date-and-time;
          description
            "Date and time when the last response
             was received for a probe.";
        }

        container probe-results {
          description "Result info of test probes";
          list probe-result {
            key "probe-index";
            description "Result info of each test probe";
            leaf probe-index {
              type uint32;
              config false;
              description "Probe index";
            }
            leaf return-code {
              type uint8;
              config false;
              description "The Return Code set in the echo reply";
            }
            leaf return-sub-code {
               type uint8;
               config false;
               description
                  "The Return Sub-code set in the echo reply.";
            }
            leaf rtt {
              type uint32;
              config false;
              description "The round-trip-time (RTT) received";
            }
            leaf result-type {
              type result-type;
              config false;
              description "The probe result type";
            }
```

```
          }
        }
      }
    }
  }
}
<CODE ENDS>
```

7. Examples

The following examples show the netconf RPC communication between client and server for one LSP-Ping test case.

7.1. Configuration of Control Information

Configure the control-parameters for sample-test-case.

Request from netconf client:

```
<rpc
  message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lsp-ping">
        <lsp-ping>
          <lsp-ping-name>sample-test-case</lsp-ping-name>
          <control-parameters>
            <target-fec-type>ip-prefix</target-fec-type>
            <ip-prefix>2001:db8::1:100/64</ip-prefix>
            <reply-mode>reply-via-udp</reply-mode>
            <timeout>1</timeout>
            <timeout-units>seconds</timeout-units>
            <interval>1</interval>
            <interval-units>seconds</interval-units>
            <probe-count>6</probe-count>
            <admin-status>enabled</admin-status>
            <data-size>64</data-size>
            <data-fill>this is a lsp ping test</data-fill>
            <source-address>2001:db8::4</source-address>
            <ttl>56</ttl>
          </control-parameters>
        </lsp-ping>
      </lsp-pings>
    </config>
  </edit-config>
</rpc>
```

Reply from netconf server:

```
<rpc-reply
  message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

7.2. The Configuration of Schedule Parameters

Set the scheduling-parameters for sample-test-case to start the test.

```
Request from netconf client:
<rpc
  message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lsp-ping">
        <lsp-ping>
          <lsp-ping-name>sample-test-case</lsp-ping-name>
          <scheduling-parameters>
            <start-test-now/>
          </scheduling-parameters>
        </lsp-ping>
      </lsp-pings>
    </config>
  </edit-config>
</rpc>

Reply from netconf server:
<rpc-reply
  message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

7.3. Display of Result Information

Get the result-info of sample-test-case.

```
Request from netconf client:
<rpc
  message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lsp-ping">
       <lsp-ping>
         <lsp-ping-name>sample-test-case</lsp-ping-name>
           <result-info/>
       </lsp-ping>
      </lsp-pings>
    </filter>
    </get>
</rpc>

Reply from netconf server:
<rpc-reply
  message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
<data>
  <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lsp-ping">
    <lsp-ping>
      <lsp-ping-name>sample-test-case</lsp-ping-name>
      <result-info>
        <operational-status>completed</operational-status>
        <source-address>2001:db8::4</source-address>
        <target-fec-type>ip-prefix</target-fec-type>
        <ip-prefix>2001:db8::1:100/64</ip-prefix>
        <min-rtt>10</min-rtt>
        <max-rtt>56</max-rtt>
        <average-rtt>36</average-rtt>
        <probe-responses>6</probe-responses>
        <sent-probes>6</sent-probes>
        <sum-of-squares>8882</sum-of-squares>
        <last-good-probe>2015-07-01T10:36:56<last-good-probe>
        <probe-results>
          <probe-result>
            <probe-index>0</probe-index>
            <return-code>0</return-code>
            <return-sub-code>3</return-sub-code>
            <rtt>10</rtt>
            <result-type>success</result-type>
          </probe-result>
          <probe-result>
            <probe-index>1</probe-index>
            <return-code>0</return-code>
            <return-sub-code>3</return-sub-code>
            <rtt>56</rtt>
            <result-type>success</result-type>
          </probe-result>
          <probe-result>
            <probe-index>2</probe-index>
            <return-code>0</return-code>
            <return-sub-code>3</return-sub-code>
            <rtt>35</rtt>
            <result-type>success</result-type>
          </probe-result>
          <probe-result>
            <probe-index>3</probe-index>
            <return-code>0</return-code>
            <return-sub-code>3</return-sub-code>
            <rtt>38</rtt>
            <result-type>success</result-type>
          </probe-result>
          <probe-result>
            <probe-index>4</probe-index>
            <return-code>0</return-code>
```

```
                <return-sub-code>3</return-sub-code>
                <rtt>36</rtt>
                <result-type>success</result-type>
              </probe-result>
              <probe-result>
                <probe-index>5</probe-index>
                <return-code>0</return-code>
                <return-sub-code>3</return-sub-code>
                <rtt>41</rtt>
                <result-type>success</result-type>
              </probe-result>
            </probe-results>
          </result-info>
        </lsp-ping>
      </lsp-pings>
    </data>
  </rpc-reply>
```

8. Security Considerations

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

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 an adverse effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

TBD

Unauthorized access to any data node of these subtrees can adversely affect the routing subsystem of both the local device and the network. This may lead to corruption of the measurement that may result in false corrective action, e.g., false negative or false positive. That could be, for example, prolonged and undetected

deterioration of the quality of service or actions to improve the quality unwarranted by the real network conditions.

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:

TBD

Unauthorized access to any data node of these subtrees can disclose the operational state information of VRRP on this device.

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. These are the operations and their sensitivity/vulnerability:

TBD

The LSP ping YANG module inherits all security consideration of [RFC8029].

9. IANA Considerations

The IANA is requested to as assign a new namespace URI from the IETF XML registry.

URI:TBA

Contributors

Yanfeng Zhang

Huawei Technologies

zhangyanfeng@huawei.com

Sam Aldrin

Google

aldrin.ietf@gmail.com

Acknowledgments

TBD

12. References

12.1. 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>.

[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>.

[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>.

[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>.

12.2. Informative References

[I-D.mahesh-netconf-persistent] Jethanandani, M., "NETCONF and persistent responses", draft-mahesh-netconf-persistent-00 (work in progress), October 2014.

[RFC4560] Quittek, J., Ed. and K. White, Ed., "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", RFC 4560, DOI 10.17487/RFC4560, June 2006, <https://www.rfc-editor.org/info/rfc4560>.

[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>.

[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>.

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <https://www.rfc-editor.org/info/rfc7594>.

[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>.

[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>.

[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>.

[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>.

Authors' Addresses

Lianshu Zheng
Huawei Technologies
China

Email: vero.zheng@huawei.com

Guangying Zheng
Huawei Technologies
China

Email: zhengguangying@huawei.com

Greg Mirsky
ZTE Corp.
USA

Email: gregimirsky@gmail.com

Reshad Rahman
Cisco Systems
Canada

Email: rrahman@cisco.com

Faisal Iqbal
Cisco Systems

Email: faiqbal@cisco.com