Abstract

[Editor’s note - there was a comment that synonymous was not the right term because synonymous implied a greater degree of interchangeability than is actually the case (there is only one way interchangeability). I have looked for other terms, and so far I have only come up with enhanced and multi-purpose, but they are not quite right either. I plan to continue with the term unless anyone has a better idea.]

This document describes a method of providing flow identification information when making RFC6374 performance measurements. This allows RFC6374 measurements to be made on multi-point to point LSPs and allows the measurement of flows within an MPLS construct using RFC6374.

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.
1. Introduction

[I-D.bryant-mpls-flow-ident] describes the requirement for introducing flow identities when using RFC6374 [RFC6374] packet Loss Measurements (LM). In summary RFC6374 uses the LM packet as the packet accounting demarcation point. Unfortunately this gives rise to a number of problems that may lead to significant packet accounting errors in certain situations. For example:
1. Where a flow is subjected to Equal Cost Multi-Path (ECMP) treatment packets can arrive out of order with respect to the LM packet.

2. Where a flow is subjected to ECMP treatment, packets can arrive at different hardware interfaces, thus requiring reception of an LM packet on one interface to trigger a packet accounting action on a different interface which may not be co-located with it. This is a difficult technical problem to address with the required degree of accuracy.

3. Even where there is no ECMP (for example on RSVP-TE, MPLS-TP LSPs and FWs) local processing may be distributed over a number of processor cores, leading to synchronization problems.

4. Link aggregation techniques may also lead to synchronization issues.

5. Some forwarder implementations have a long pipeline between processing a packet and incrementing the associated counter again leading to synchronization difficulties.

An approach to mitigating these synchronization issue is described in [I-D.tempia-ippm-p3m] and [I-D.chen-ippm-coloring-based-ipfpm-framework] in which packets are batched by the sender and each batch is marked in some way such that adjacent batches can be easily recognized by the receiver.

An additional problem arises where the LSP is a multi-point to point LSP, since MPLS does not include a source address in the packet. Network management operations require the measurement of packet loss between a source and destination. It is thus necessary to introduce some source specific information into the packet to identify packet batches from a specific source.

This document describes a method of accomplishing this by using a technique called Synonymous Flow Labels (SFL) (see (Section 3)) in which labels which mimic the behaviour of other labels provide the packet batch identifiers and enable the per batch packet accounting.

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].
3. Synonymous Flow Labels

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

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

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

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

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

1. Applications label present
2. Single label stack

4.1. Applications Label Present

Figure 1 shows the case in which both an LSP label and an application label is present in the MPLS label stack. Uninstrumented traffic runs over the "normal" stack, and instrumented flows run over the SFL stack with the SFL used to indicate the packet batch.

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

"Normal" Label Stack         Label Stack with SFL
```

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

At the egress LSR the LSP label is popped (if present). Then the SFL is processed in exactly the same way as the corresponding application label would have been processed. Where the SFL is being used to support RFC6374 packet loss measurements, as an additional operation, the total number of packets received with this particular SFL is recorded.

Where the number of labels used by a single application is large, and the increase in the number of allocated labels needed to support the
SFL actions consequently becomes too large to be viable, it may be necessary to introduce an additional label in the stack to act as an aggregate instruction. This situation will be considered in a future version of this document.

4.1.1. Setting TTL and the Traffic Class Bits

To be provided in a future version of this draft.

4.2. Single Label Stack

Figure 2 shows the case in which only an LSP label is present in the MPLS label stack. Uninstrumented traffic runs over the "normal" stack and instrumented flows run over the SFL stack with the SFL used to indicate the packet batch. However in this case it is necessary for the ingress LSR to first push the SFL and then to push the LSP label.

```
+-----------------+                      +-----------------+
|                 |                      |                 |
|     LSP         |                      |     LSP        |
|     Label       |                      |     Label      |
+-----------------+                      +-----------------+  <= Synonymous with Explicit NULL

|                 |                      |                 |
|      Payload    |                      |      Payload    |
+-----------------+                      +-----------------+  <= Bottom of stack

"Normal" Label Stack                      Label Stack with SFL
```

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

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

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

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

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

4.2.1. Setting TTL and the Traffic Class Bits

To be provided in a future version of this draft.

4.3. Aggregation of SFL Actions

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

This third SFL case requires further thought by the authors and this section will be updated in a future version of this draft to reflect those thoughts.

5. Equal Cost Multipath Considerations

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

1. The operator can elect to always run with the SFL in place in the MPLS label stack.
2. The operator can elect to use [RFC6790] Entropy Labels which, in a network that fully supports this type of ECMP, results in the ECMP decision being independent of the value of the other labels in the label stack.

6. RFC6374 Packet Loss Measurement with SFL

The packet format of an RFC6374 Query message using SFLs is shown in Figure 4.

![Diagram of RFC6374 Query Packet with SFL]

The MPLS label stack is exactly the same as that used for the user data service packets being instrumented (see Section 4). The RFC6374 measurement message consists of the three components, the RFC6374 fixed header as specified in [RFC6374] carried over the ACH channel.
type specified the type of measurement being made (currently: loss, delay or loss and delay) as specified in RFC6374.

Two optional TLVs MAY also be carried if needed. The first is the SFL TLV specified in Section 6.1. This is used to provide the implementation with a reminder of the SFL that was used to carry the RFC6374 message. This is needed because a number of MPLS implementations do not provide the MPLS label stack to the MPLS OAM handler. This TLV is required if RFC6374 messages are sent over UDP (draft-bryant-mpls-RFC6374-over-udp). This TLV MUST be included unless, by some method outside the scope of this document, it is known that this information is not needed by the RFC6374 Responder.

The second set of information that may be needed is the return information that allows the responder send the RFC6374 response to the Querier. This is not needed if the response is requested in-band and the MPLS construct being measured is a point to point LSP, but otherwise MUST be carried. The return address TLV is defined in RFC6378 and the optional UDP Return Object is defined in [I-D.ietf-mpls-rfc6374-udp-return-path].

6.1. RFC6374 SFL TLV

[Editor’s Note we need to review the following in the light of further thoughts on the associated signaling protocol(s). I am fairly confident that we need all the fields other than SFL Batch and SFL Index. The Index is useful in order to map between the label and information associated with the FEC. The batch is part of the lifetime management process]

The required RFC6374 SFL TLV is shown in Figure 5. This contains the SFL that was carried in the label stack, the FEC that was used to allocate the SFL and the index into the batch of SLs that were allocated for the FEC that corresponds to this SFL.
<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>MBZ</th>
<th>SFL Batch</th>
<th>SFL Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFL</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5: SFL TLV**

Where:

- **Type**: Type is set to Synonymous Flow Label (SFL-TLV).
- **Length**: The length of the TLV as specified in [RFC6374].
- **MBZ**: MUST be sent as zero and ignored on receive.
- **SFL Batch**: The SFL batch that this SFL was allocated as part of (see draft-bryant-mpls-sfl-control).
- **SPL Index**: The index into the list of SFLs that were assigned against the FEC that corresponds to the SFL.
- **SFL**: The SFL used to deliver this packet. This is an MPLS label which is a component of a label stack entry as defined in Section 2.1 of [RFC3032].
- **Reserved**: MUST be sent as zero and ignored on receive.
- **FEC**: The Forwarding Equivalence Class that was used to request this SFL. This is encoded as per Section 3.4.1 of

This information is needed to allow for operation with hardware that discards the MPLS label stack before passing the remainder of the stack to the OAM handler. By providing both the SFL and the FEC plus index into the array of allocated SFLs a number of implementation types are supported.
7. The Application of SFL to other PM Types

SFL can be used to enable other types of PM in addition to loss. Delay, Delay Variation and Throughput may be calculated based on measurement results collected through Loss and Delay Measurement test sessions. Further details will be provided in a future version of this draft.

8. 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.

9. Security Considerations

The issue noted in Section 8 is a security consideration. There are no other new security issues associated with the MPLS dataplane. Any control protocol used to request SFLs will need to ensure the legitimacy of the request.

10. IANA Considerations

IANA is request to allocate a new TLV from the 0-127 range on the MPLS Loss/Delay Measurement TLV Object Registry:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>Synonymous Flow Label</td>
<td>This</td>
</tr>
</tbody>
</table>

A value of 4 is recommended.

11. Acknowledgements

TBD

12. References
12.1.  Normative References

[I-D.ietf-mpls-rfc6374-udp-return-path]


12.2.  Informative References

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

[I-D.chen-ippm-coloring-based-ipfpm-framework]

[I-D.tempia-ippm-p3m]


Authors’ Addresses

Stewart Bryant
Cisco Systems

Email: stbryant@cisco.com
Refresh Interval Independent FRR Facility Protection
draft-chandra-mpls-ri-rsvp-frr-04

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 November 7, 2016.

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
Abstract

RSVP-TE relies on periodic refresh of RSVP messages to synchronize and maintain the LSP related states along the reserved path. In the absence of refresh messages, the LSP related states are automatically deleted. Reliance on periodic refreshes and refresh timeouts are problematic from the scalability point of view. The number of RSVP-TE LSPs that a router needs to maintain has been growing in service provider networks and the implementations should be capable of handling increase in LSP scale.

RFC 2961 specifies mechanisms to eliminate the reliance on periodic refresh and refresh timeout of RSVP messages, and enables a router to increase the message refresh interval to values much larger than the default 30 seconds defined in RFC 2205. However, the protocol extensions defined in RFC 4090 for supporting fast reroute (FRR) using bypass tunnels implicitly rely on short refresh timeouts to cleanup stale states.

In order to eliminate the reliance on refresh timeouts, the routers should unambiguously determine when a particular LSP state should be deleted. Coupling LSP state with the corresponding RSVP-TE signaling adjacencies as recommended in RSVP-TE Scaling Recommendations (draft-ietf-teas-rsvp-te-scaling-rec) will apply in scenarios other than RFC 4090 FRR using bypass tunnels. In scenarios involving RFC 4090 FRR using bypass tunnels, additional explicit tear down messages are necessary. Refresh-interval Independent RSVP FRR (RI-RSVP-FRR) extensions specified in this document consists of procedures to enable LSP state cleanup that are essential in scenarios not covered by procedures defined in RSVP-TE Scaling Recommendations.

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].
1. Introduction

RSVP-TE Fast Reroute [RFC4090] defines two local repair techniques to reroute label switched path (LSP) traffic over pre-established backup tunnel. Facility backup method allows one or more LSPs traversing a connected link or node to be protected using a bypass tunnel. The many-to-one nature of local repair technique is attractive from scalability point of view. This document enumerates facility backup procedures in RFC 4090 that rely on refresh timeout and hence make facility backup method refresh-interval dependent. The RSVP-TE extensions defined in this document will enhance the facility backup protection mechanism by making the corresponding procedures refresh-interval independent.

1.1. 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 RSVP Refresh Overhead Reduction Extensions [RFC2961].

The problems listed above adversely affect RSVP control plane scalability and RSVP-TE [RFC3209] inherited 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. For more details on eliminating dependency on refresh timeout for stale state cleanup, refer to "Refresh Interval Independent RSVP" section in [TE-SCALE-REC].

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.

2. Terminology

The reader is assumed to be familiar with the terminology in [RFC2205], [RFC3209], [RFC4090] and [RFC4558].

Phop node: Previous-hop router along the label switched path

PPhop node: Previous-Previous-hop router along the LSP

LP-MP node: Merge Point router at the tail of Link-protecting bypass tunnel

NP-MP node: Merger Point router at the tail of Node-protecting bypass tunnel

TED: Traffic Engineering Database

Conditional PathTear: PathTear message containing a suggestion to a receiving downstream router to retain Path state if the receiving router is NP-MP

Remote PathTear: PathTear message sent from Point of Local Repair (PLR) to MP to delete state on MP if PLR had not reliably sent backup Path state before

3. Problem Description
In the topology 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. As each router should consider itself as a Merge Point, C will time out the state only after waiting for an additional duration equal to refresh timeout.

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.

- Utilize MP determination mechanism specified in [SUMMARY-FRR] that enables the PLR to signal availability of local protection to MP. In addition, introduce PLR and MP procedures to establish Node-ID hello session between the PLR and the MP to detect router failures and to determine capability. See section 4.1 for more details. This part of the solution re-uses some of the extensions defined in [SUMMARY-FRR] and [TE-SCALE-REC], and the subsequent sub-sections will list the extensions in these drafts that are utilized in this document.

- 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 Handshake between PLR and MP

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 additional procedures.

- 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 detect MP node failure as well as determine the capability of the MP node. If the MP sets I-bit in CAPABILITY object [TE-SCALE-REC] carried in Hello message corresponding to NodeID based Hello session, then the PLR SHOULD conclude that the MP supports refresh-interval independent FRR procedures defined in this document.

- If the bypass LSP comes up, then the PLR SHOULD include Bypass Summary FRR Association object and triggers PATH to be sent. If Bypass Summary FRR Association object is included in PATH message, then the encoding rules specified in [SUMMARY-FRR] MUST be followed.
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. The default hello interval for this NodeID hello session SHOULD be set to the default specified in [TE-SCALE-REC].

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. MP Behavior

When the NNhop or Nhop node receives the triggered PATH with a "matching" Bypass Summary FRR Association object, the node should consider itself as the MP for the PLR IP address "corresponding" to the Bypass Summary FRR Association object. The matching and ordering rules of Bypass Summary FRR Association specified in [SUMMARY-FRR] SHOULD be followed by implementations supporting this document.

In addition to the above procedures, the node SHOULD check the presence of remote signaling adjacency with PLR (this check is needed to detect network being partitioned). If a matching Bypass Summary FRR Association object is found in PATH and the RSVP-TE signaling adjacency is present, the node concludes that the PLR will undertake refresh-interval independent FRR procedures specified in this document. 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 a matching Bypass Summary FRR Association object is included by PPhop node, then it is NP-MP. If a matching Bypass Summary FRR Association object is included by Phop node, it concludes it is LF-MP.

4.1.4. "Remote" state on MP

Once a router concludes it is MP for a PLR running refresh-interval independent FRR procedures, 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 RSVP_HOP object. The RSVP_HOP object in "remote" Path state contains the address that the PLR uses to send NodeID hello messages to MP.

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

- MP later receives a PATH with no matching Bypass Summary FRR Association object corresponding to the PLR RRO, 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 from 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.

Immediate node failures are detected from the state of NodeID hello sessions established with immediate neighbors. [TE-SCALE-REC] recommends each router to establish NodeID hello sessions with all its immediate neighbors. PLR or MF node failure is detected from the
state of remote signaling adjacency established according to Section 4.1.2 of this document.

4.2.1. Non-MP Behavior

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

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.

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.

4.2.3. NP-MP Behavior

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.

When a router that is NP-MP does not detect Phop link or node failure, but receives Conditional PathTear from the Phop node, then 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.

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, C SHOULD remove SUMMARY_FRR_BYPASS_ASSOCIATION sub-object corresponding to B from the RRO 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 SUMMARY_FRR_BYPASS_ASSOCIATION sub-object corresponding to B.

4.2.4. Behavior of a Router that is both 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.

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 had previously included Bypass Summary FRR Association object in PATH. If the object had been included previously by Phop, then the node processing Conditional PathTear from Phop SHOULD remove the corresponding object 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

To be assigned

C-type
M bit

If M-bit is set to 1, then the PathTear message SHOULD be processed based on the condition if the receiver router is a Merge Point or not.

If M-bit is set to 0, then the PathTear message SHOULD be processed as normal PathTear message.

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. The TTL in "remote" PathTear message SHOULD be set to 255.

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 RSVP_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 "Refresh interval Independent FRR" or RI-RSVP-FRR referred below in this section refers to the changes that have been proposed in previous sections. Any implementation that does not support them has been termed as "non-RI-RSVP-FRR implementation". The extensions proposed in [SUMMARY-FRR] are applicable to implementations that do not support RI-RSVP-FRR. On the other hand, changes proposed relating to LSP state cleanup namely Conditional and remote PathTear require support from one-hop and two-hop neighboring 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 Refresh interval Independent FRR

An implementation supporting the extensions specified in previous sections (called RI-RSVP-FRR hereafter) SHOULD set the flag "Refresh interval Independent RSVP" or RI-RSVP in CAPABILITY object in Hello messages. The RI-RSVP flag is specified in [TE-SCALE-REC].

- 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 RI-RSVP-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 RI-RSVP-FRR extensions.

- If node protection is requested for an LSP and if (a) PPhop node has not included a matching Bypass Summary FRR Association object in PATH or (b) PPhop node has not initiated remote node Hello messages, then the node SHOULD conclude that PLR does not support RI-RSVP-FRR extensions. The details are described in the "Procedures for backward compatibility" section below.

Any node that sets the I-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 non-RI-RSVP-FRR implementations.

4.5.2.1. Lack of support on Downstream Node

- If the Nhop does not support the RI-RSVP-FRR extensions, 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 the RI-RSVP-FRR extensions, 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 the RI-RSVP-FRR extensions, 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 the RI-RSVP-FRR extensions, 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 RI-RSVP-FRR extensions, 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 RI-RSVP-FRR extensions, 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 procedures specified in Section 4.2 of this document.

4.5.2.3. Incremental Deployment

The backward compatibility procedures described in the previous subsections imply that a router supporting the RI-RSVP-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.

- RI-RSVP-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.

- RI-RSVP-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.

- RI-RSVP-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.

- RI-RSVP-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 an implementation supporting the RI-RSVP-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 security considerations pertaining to [RFC2205], [RFC3209] and [RFC5920] remain relevant.

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. Implementations SHOULD also
provide the option to specify a limit on the number of Node-ID based Hello sessions that can be established on a router supporting the extensions defined in this document.

6. IANA Considerations

6.1. New Object - CONDITIONS

RSVP Change Guidelines [RFC3936] 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 from 128-183 range called the "CONDITIONS" object is defined in Section 4.3 of this document. The Class-Number from 128-183 range will be allocated by IANA.

7. Normative References


8. Informative References


9. Acknowledgments

We are very grateful to Yakov Rekhter for his contributions to the development of the idea and thorough review of content of the draft. Thanks to Raveendra Torvi and Yimin Shen for their comments and inputs.

10. Contributors

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

Ebben Aries
Juniper Networks
11. Authors’ Addresses

Chandra Ramachandran
Juniper Networks
Email: csekar@juniper.net

Ina Minei
Google, Inc
inaminei@google.com

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

Tarek Saad
Cisco Systems Inc.
Email: tsaad@cisco.com
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 .................................................. 3
2. Requirements for MPLS-TP Ring Protection .................... 4
   2.1. Recovery of Multiple Failures ........................... 4
   2.2. Smooth Upgrade from Linear Protection to Ring Protection 4
   2.3. Configuration Complexity ............................... 5
3. Terminology and Notation .................................... 5
4. Shared Ring Protection Architecture .......................... 5
   4.1. Ring Tunnel ............................................. 5
   4.1.1. Establishment of Ring Tunnel .......................... 6
   4.1.2. Label Assignment and Distribution .................... 8
   4.1.3. Forwarding Operation ................................. 8
   4.2. Failure Detection ....................................... 9
   4.3. Ring Protection ......................................... 10
   4.3.1. Wrapping ............................................. 10
   4.3.2. Short Wrapping ....................................... 12
   4.3.3. Steering ............................................. 14
   4.4. Interconnected Ring Protection .......................... 17
   4.4.1. Interconnected Ring Topology .......................... 17
   4.4.2. Interconnected Ring Protection Mechanisms ............ 18
   4.4.3. Ring Tunnels in Interconnected Rings ................. 19
   4.4.4. Interconnected Ring Switching Procedure .............. 21
   4.4.5. Interconnected Ring Detection Mechanism ............... 22
5. Ring Protection Coordination Protocol ........................ 23
   5.1. RPS Protocol ........................................... 24
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.
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:

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

   ┌─┐
   │ F │---------------------│ A │ +-- LSP1
   └─┘
        +---+#####[RaP_D(E)]####+---+
        | E |
        +---+                          +---+
        #/*                        */#       
        
        ┌─┐
        │ E │---------------------│ B │
        └─┘                          +---+  
        #/*                        */#       
        
        ┌─┐
        │ E |
        +---+  
        #/
        /#       
        
        ┌─┐
        │ E |
        +---+  
        #/
        /#       
        
        ┌─┐
        │ E |
        +---+  
        #/
        /#       
        
        ┌─┐
        │ E |
        +---+  
        #/
        /#       

        +---+#####[RaP_D(C)]####+---+  
        | D |---------------------| C |
        +---+#####[RaP_D(D)]####+---+  

-----physical links      ***** RxW_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:

\[ \text{[LSP1]}(\text{original data traffic carried by LSP1}) \rightarrow \text{[Rcw}_D(B)|\text{LSP1]}(\text{NodeA}) \rightarrow \text{[Rcw}_D(C)|\text{LSP1]}(\text{NodeB}) \rightarrow \text{[Rcw}_D(D)|\text{LSP1]}(\text{NodeC}) \rightarrow \text{[LSP1]}(\text{data traffic carried by LSP1}). \]

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:
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:

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

\[
\text{Figure 5. Wrapping for link failure}
\]

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.

\[ +---+#####[RaP_D(F)]#####+-+-+ LSP1 \\
| F |---------------------| A | +-- LSP1 \\
+---+*****[RcW_D(A)]*****+++ +/
\]

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

\[ [RaP_D(D)]#/ [RcW_D(C)]*/[RaP_D(B)] \\
\#/  \\
+/-----++[RcW_D(D)]++++++ ++
LSP1 +-- | D |---------------------| C | \\
+/-----[RaP_D(C)]++++++ ++
\]

-----physical links xaaaaa Failure Node
*****RaP_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 |
++++          ++++          
    */
[RaP_D(D)]#
[RcW_D(C)]*RaP_D(B)
    *
++++[RcW_D(D)]*****+-+
LSPL  +++| 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 failure 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.

![Diagram]

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
update 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.

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) through
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.
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 10. Steering operation and protection switching (2)
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.:

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

Figure 13. Ring tunnels for the interconnected rings
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 detect 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.
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 identify 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 behavior 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.
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.

- 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.

- 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 1 0 |0 0 0 0 |0 0 0 0 0 0 0 0 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Dest Node ID  | Src Node ID   |   Request     |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Figure 17. G-ACh RPS Packet Format
```

The following fields MUST be provided:

- **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.

- **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.

- **RPS request code**: A code consisting of eight bits as specified below:
<table>
<thead>
<tr>
<th>Bits</th>
<th>Condition, State or external Request</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 1 1 1</td>
<td>Lockout of Protection (LP)</td>
<td>highest</td>
</tr>
<tr>
<td>0 0 0 0 0 1 1 0</td>
<td>Forced Switch (FS)</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 1</td>
<td>Signal Fail (SF)</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 0</td>
<td>Manual Switch (MS)</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 1</td>
<td>Wait-To-Restore (WTR)</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 1 0 0 1</td>
<td>Exercise (EXER)</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td>Reverse Request (RR)</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td>No Request (NR)</td>
<td>lowest</td>
</tr>
</tbody>
</table>

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:

- LP and LP
- FS and FS
- SF and SF
- 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:

- 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
- An externally initiated command for this node
- 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:

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

- A different RPS request with a higher priority than WTR is received
- Another failure is detected
- 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:

- An equal priority, a higher priority, or an allowed coexisting externally initiated command
5.2. RPS State Machine

5.2.1. Initial States

- The detection of an equal priority, a higher priority, or an allowed coexisting automatic initiated command
- The receipt of an equal, a higher priority, or an allowed coexisting RPS request destined to this node
### 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.

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Signaled RPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Idle</td>
</tr>
<tr>
<td></td>
<td>Working: no switch</td>
</tr>
<tr>
<td></td>
<td>Protection: no switch</td>
</tr>
<tr>
<td>B</td>
<td>Pass-through</td>
</tr>
<tr>
<td></td>
<td>Working: no switch</td>
</tr>
<tr>
<td></td>
<td>Protection: pass through</td>
</tr>
<tr>
<td>C</td>
<td>Switching - LP</td>
</tr>
<tr>
<td></td>
<td>Working: no switch</td>
</tr>
<tr>
<td></td>
<td>Protection: no switch</td>
</tr>
<tr>
<td>D</td>
<td>Idle - LW</td>
</tr>
<tr>
<td></td>
<td>Working: no switch</td>
</tr>
<tr>
<td></td>
<td>Protection: no switch</td>
</tr>
<tr>
<td>E</td>
<td>Switching - FS</td>
</tr>
<tr>
<td></td>
<td>Working: switched</td>
</tr>
<tr>
<td></td>
<td>Protection: switched</td>
</tr>
<tr>
<td>F</td>
<td>Switching - SF</td>
</tr>
<tr>
<td></td>
<td>Working: switched</td>
</tr>
<tr>
<td></td>
<td>Protection: switched</td>
</tr>
<tr>
<td>G</td>
<td>Switching - MS</td>
</tr>
<tr>
<td></td>
<td>Working: switched</td>
</tr>
<tr>
<td></td>
<td>Protection: switched</td>
</tr>
<tr>
<td>H</td>
<td>Switching - WTR</td>
</tr>
<tr>
<td></td>
<td>Working: switched</td>
</tr>
<tr>
<td></td>
<td>Protection: switched</td>
</tr>
<tr>
<td>I</td>
<td>Switching - EXER</td>
</tr>
<tr>
<td></td>
<td>Working: no switch</td>
</tr>
<tr>
<td></td>
<td>Protection: no switch</td>
</tr>
<tr>
<td>Initial state</td>
<td>New request</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>A (Idle)</td>
<td>LP</td>
</tr>
<tr>
<td></td>
<td>LW</td>
</tr>
<tr>
<td></td>
<td>FS</td>
</tr>
<tr>
<td></td>
<td>SF</td>
</tr>
<tr>
<td>Recover from SF</td>
<td>N/A</td>
</tr>
<tr>
<td>MS</td>
<td>N/A</td>
</tr>
<tr>
<td>WTR expires</td>
<td>N/A</td>
</tr>
<tr>
<td>EXER</td>
<td>I (Switching - EXER)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>B (Pass-trough)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>LW</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td>Recover from SF</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>WTR expires</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>EXER</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>

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

---

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

Cheng, et al. Expires February 12, 2016 [Page 34]
<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>G (Switching - MS)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td>LW</td>
<td></td>
<td>D (Idle - LW) - if on the same span</td>
</tr>
<tr>
<td>FS</td>
<td></td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td>SF</td>
<td></td>
<td>N/A - if on the samespan</td>
</tr>
<tr>
<td>Recover from SF</td>
<td></td>
<td>F (Switching - SF) - if on another span</td>
</tr>
<tr>
<td>MS</td>
<td></td>
<td>N/A - if on the same span</td>
</tr>
<tr>
<td>Clear</td>
<td></td>
<td>G (Switching - MS) - if on another span release the switches but signal MS</td>
</tr>
<tr>
<td>WTR expires</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>EXER</td>
<td></td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>H (Switching - WTR)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td>LW</td>
<td></td>
<td>D (Idle - W)</td>
</tr>
<tr>
<td>FS</td>
<td></td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td>SF</td>
<td></td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td>Recover from SF</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>MS</td>
<td></td>
<td>G (Switching - MS)</td>
</tr>
<tr>
<td>Clear</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>WTR expires</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>EXER</td>
<td></td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I (Switching - EXER)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td>LW</td>
<td></td>
<td>D (Idle - W)</td>
</tr>
<tr>
<td>FS</td>
<td></td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td>SF</td>
<td></td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td>Recover from SF</td>
<td></td>
<td>N/A</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (Idle)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>G (Switching - MS)</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A - cannot happen when there is LP or MS request in the ring</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>N/A - cannot happen when there is LP, FS, SF or MS request in the ring</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>N/A - if received from other side</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>A (Idle)</td>
</tr>
</tbody>
</table>

both sides

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (Switching - LP)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>N/A - cannot happen when there is LP request in the ring</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>N/A - cannot happen when there is LP request in the ring</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>N/A - cannot happen when there is LP request in the ring</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>N/A - cannot happen when there is LP request in the ring</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>D (Idle - LW)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>G (Switching - MS)</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>I (Switching - EXER)</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>D (Idle - LW)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>E (Switching - FS)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>N/A - cannot happen when there is FS request in the ring</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>N/A - cannot happen when there is FS request in the ring</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>F (Switching - SF)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>N/A - cannot happen when there is SF request in the ring</td>
</tr>
<tr>
<td>Initial state</td>
<td>New request</td>
<td>New state</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>G (Switching - MS)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>G (Switching - MS) - release the switches but signal MS</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>N/A - cannot happen when there is MS request in the ring</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>G (Switching - MS)</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>H (Switching - WTR)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>G (Switching - MS)</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>H (Switching - WTR)</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>N/A - cannot happen when there is WTR request in the ring</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>H (Switching - WTR)</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I (Switching - EXER)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>G (Switching - MS)</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>I (Switching - EXER)</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>I (Switching - EXER)</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (Idle)</td>
<td>LP</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>B (Pass-trough)</td>
<td>LP</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>N/A - cannot happen when there is LP request in the ring</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>N/A - cannot happen when there is LP request in the ring</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>N/A - cannot happen when there is LP, FS or SF request in the ring</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A - cannot happen when there is LP, FS, SF or MS request in the ring</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>N/A - cannot happen when there is LP, FS, SF, MS or WTR request in the ring</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>B (Pass-trough)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (Switching - LP)</td>
<td>LP</td>
<td>C (Switching - LP)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>N/A - cannot happen when there is LP request in the ring</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>N/A - cannot happen when there is LP request in the ring</td>
</tr>
</tbody>
</table>

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 request in the ring
EXER N/A - cannot happen when there
       is LP request in the ring
RR  N/A
NR  N/A

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>D (Idle - LW)</td>
<td>LP</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>E (Switching - FS)</td>
<td>LP</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>E (Switching - FS)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>N/A - cannot happen when there is FS request in the ring</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A - cannot happen when there is FS request in the ring</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>N/A - cannot happen when there is FS request in the ring</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>F (Switching - SF)</td>
<td>LP</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>F (Switching - SF)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>N/A - cannot happen when there is SF request in the ring</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A - cannot happen when there is SF request in the ring</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>N/A - cannot happen when there is SF request in the ring</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
<tr>
<td>Initial state</td>
<td>New request</td>
<td>New state</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>G (Switching - MS)</td>
<td>LP</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>G (Switching - MS) - release</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A - cannot happen when there</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>N/A - cannot happen when there</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>H (Switching - WTR)</td>
<td>LP</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>N/A - cannot happen when there</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial state</th>
<th>New request</th>
<th>New state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I (Switching - EXER)</td>
<td>LP</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>SF</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>MS</td>
<td>B (Pass-trough)</td>
</tr>
<tr>
<td></td>
<td>WTR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>EXER</td>
<td>I (Switching - EXER)</td>
</tr>
<tr>
<td></td>
<td>RR</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>NR</td>
<td>N/A</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>Ring Protection Switching Protocol (RPS)</td>
<td>this document</td>
</tr>
</tbody>
</table>

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:

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

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


10.2. Informative References


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
Abstract

RFC 5492 defines capabilities advertisement for the BGP peer. In addition, it is useful to know the capabilities of the BGP Next-Hop, in particular for forwarding plane features. RFC 5492 is not applicable because the BGP peer may be different from the BGP Next-Hop, in particular when BGP Route Reflection is used. This document defines a mechanism to advertise such BGP Next Hop Capabilities.

This document defines a new BGP non-transitive attribute to carry Next-Hop Capabilities. This attribute is deleted when the BGP Next Hop is changed.

This document also defines a Next-Hop capability to advertise the ability to handle the Entropy Label defined in RFC 6790.

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."
1. Introduction

[RFC5492] defines capabilities advertisement for the BGP peer. It is also useful to know the capabilities of the BGP Next-Hop, in particular for forwarding plane features. RFC 5492 is not applicable because the BGP peer may be different from the BGP Next-Hop, in particular when BGP Route Reflection is used. This document defines a mechanism to advertise such BGP Next Hop Capabilities.
This document defines a new BGP non-transitive attribute to carry Next-Hop Capabilities. This attribute is deleted when the BGP Next Hop is changed.

This document also defines a first application to advertise the capability to handle the Entropy Label defined in [RFC6790]. Note that RFC 6790 had originally defined a BGP attribute for this but it has been latter deprecated in [RFC7447]

2. BGP Next-Hop Capabilities Attribute

The BGP Next-Hop Capabilities Attribute is an optional, non-transitive BGP Attribute, of value TBD1. The attribute consists of a set of Next-Hop Capabilities.

Inclusion of a Next-Hop Capability "X" in a BGP UPDATE message, indicates that the BGP Next-Hop, encoded in either the NEXT_HOP attribute defined in [RFC4271] or the Network Address of Next Hop field of the MP_REACH_NLRI attribute defined in [RFC4760], supports the capability "X" for the NLRI advertised in this BGP UPDATE. This document does not make distinction between these two Next-Hop fields and uses the term 'BGP Next-Hop' to refer to whichever one is used in a given BGP UPDATE message.

A Next-Hop Capability is a triple (Capability Code, Capability Length, Capability Value) aka a TLV:

A Next-Hop Capability.

+-------------------------------+
<table>
<thead>
<tr>
<th>Capability Code (1 octet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capability Length (1 octet)</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
<tr>
<td>Capability Value (variable)</td>
</tr>
</tbody>
</table>
+-------------------------------+

Capability Code: a one-octet unsigned binary integer which indicates the type of "Next-Hop Capability" advertised and unambiguously identifies an individual capability.

Capability Length: a one-octet unsigned binary integer which indicates the length, in octets, of the Capability Value field. A length of 0 indicates that no Capability Value Field is present.
Capability Value: a variable-length field from 0 to 255 octets. It is interpreted according to the value of the Capability Code.

BGP speakers SHOULD NOT include more than one instance of a Next-Hop capability with the same Capability Code, Capability Length, and Capability Value. Note, however, that processing of multiple instances of such capability does not require special handling, as additional instances do not change the meaning of the announced capability; thus, a BGP speaker MUST be prepared to accept such multiple instances.

BGP speakers MAY include more than one instance of a capability (as identified by the Capability Code) with non-zero Capability Length field, but with different Capability Value and either the same or different Capability Length. Processing of these capability instances is specific to the Capability Code and MUST be described in the document introducing the new capability.

3. BGP Next-Hop Capabilities Attribute Operation

The BGP Next-Hop Capabilities attribute being non-transitive, as per [RFC4271], a BGP speaker which does not understand it will quietly ignore it and not pass it along to other BGP peers.

A BGP speaker that understands the BGP Next-Hop Capabilities Attribute and does not change the BGP Next-Hop, SHOULD NOT change the BGP Next-Hop Capabilities Attribute and SHOULD pass the attribute unchanged along to other BGP peers.

A BGP speaker that understands the BGP Next-Hop Capabilities Attribute and changes the BGP Next-Hop, MUST remove the received BGP Next-Hop Capabilities before propagating the BGP UPDATE to other BGP peers. It MAY attach a new BGP Next-Hop Capabilities attribute describing the capabilities of the new BGP Next-Hop.

4. BGP Next-Hop Capability Code Operation

A BGP speaker receiving a BGP Next-Hop Capability Code that it supports may behave as defined in the document defining this Capability Code. A BGP speaker receiving a BGP Next-Hop Capability Code that it does not support MUST ignore this BGP Next-Hop Capability Code. In particular, this MUST NOT be handled as an error. In both cases, the BGP speaker MUST examine the remaining BGP Next-Hop Capability Code that may be present in the BGP Next-Hop Capabilities Attribute.

The BGP Next-Hop Capability Code MUST reflect the capability of the router indicated in the BGP Next-Hop. If a BGP speaker sets the BGP
Next-Hop to an address of a different router (e.g. R), it MUST NOT advertise BGP Next-Hop Capabilities not supported by this router R.

The presence of a Next-Hop Capability SHOULD NOT influence route selection or route preference of an route, unless tunneling is used to reach the BGP Next-Hop or the selected route has been learnt from EBGP (i.e. the Next-Hop is in a different AS). Indeed, it is in general impossible for a node to know that all BGP routers of the Autonomous System (AS) will understand a given Next-Hop Capability; and having different routers, within an AS, use a different preference for a route, may result in forwarding loops if tunnelling is not used to reach the BGP Next-Hop.

An implementation MAY allow, by configuration, removing this attribute when advertising the routes over eBGP.

5. BGP Next-Hop Attribute Error Handling

A BGP Next-Hop Capabilities Attribute is considered malformed if the length of the Attribute is not equal to the sum of all (BGP Hop Capability Length +2) of each capability carried in this attribute. Note that "2" is the length of the fields "Type" and "Length" of each BGP Next Hop Capability.

A BGP UPDATE message with a malformed BGP Next-Hop Capabilities Attribute SHALL be handled using the approach of "attribute discard" defined in [I-D.ietf-idr-error-handling].

Unknown Next-Hop Capabilities Codes MUST be silently ignored.

A document that specifies a new Next-Hop Capability SHOULD provide specifics regarding what constitutes an error for that Next-Hop Capability.

If a Next-Hop Capability is malformed, this Next-Hop Capability Type MUST be ignored. Others Next-Hop Capabilities MUST be processed as usual.

6. Entropy Label Next-Hop Capability

The Entropy Label Next-Hop Capability has type code 1 and a length of 0 or 1 octet.

The inclusion of the "Entropy Label" Next-Hop Capability indicates that the BGP Next-Hop can be sent packets, for all routes indicated in the NRLI, with a MPLS entropy labels (ELI, EL) added immediately after the label stack advertised with the NLRI.
On the receiving side, suppose BGP speaker S has determined that packet P is to be forwarded according to BGP route R, where R is a route of one of the labeled address families. And suppose that L is the label stack embedded in the NLRI of route R. Then to forward packet P according to route R, S either replaces P’s top label with L, or else pushes L onto the MPLS label stack. If the EL-Capability is advertised in the BGP UPDATE advertising this route R, S knows that it may safely place the ELI and an EL on the label stack immediately beneath L.

A BGP speaker S that sends an UPDATE with the BGP Next-Hop NH MAY include the Entropy Label Next-Hop Capability only if, for all the NLRI in the BGP UPDATE, either of the following is true:

- Egress case: NH is the egress of the LSP advertised with the NLRI and can lookup and its capable of handling the ELI.
- Transit LSR case: NH is a transit LSR for the LSP advertised with the NLRI (i.e. swap one of the label advertised in the NLRI) and next downstream BGP Next-Hop(s) has(have) advertised the Entropy Label Next-Hop Capability (or a similar capability signalled by protocol X if the route is redistributed, by NH, from X to BGP).

6.1. Readable Label Depth

When stacked LSPs are used and the ingress nests LSP inside this BGP signaled LSP, it would be useful for ingress LSRs to know how many additional labels the downstream LSR may read when load-balancing based on the Entropy Label. In other words, how many labels the ingress LER may push, in addition to the BGP label(s) advertised in the Network Layer Reachability Information (NLRI) field, before pushing an entropy label that will be seen by all downstream LSRs.

This maximum number of additional labels is called the Readable Label Depth (RLD) of the LSP(s). It is related, yet different, to the RLD of an node which is defined in [I-D.ietf-mpls-spring-entropy-label]

The RLD of the LSP(s) advertised in the NLRI, may be advertised in the value field of the Entropy Label Next-Hop Capability. This value field is optional. If present, the value field is a one-octet unsigned binary integer which indicates the maximum Readable Label Depth (RLD) of the LSP(s) advertised in the NLRI. In other words, this is the maximum number of additional MPLS labels that may be pushed by the ingress, in addition to the label(s) of the NRLI advertised in the BGP UPDATE, before pushing the ELI, EL labels, if it wish that all downstream LSR be capable of performing load-balancing based on the entropy label.
S SHOULD advertise a RLD of:

o its own local RLD minus the number of labels advertised in the NLRI, if S is the egress of the LSP(s) advertised in the NLRI;

o the minimum of:
  * its own node RLD minus the number of labels advertised in the NLRI;
  * the RLD of the LSP from itself to BGP NEXT_HOP of its received route minus the number of labels received in the NLRI (if any);
  * the RLD in the received BGP route (if any).

The first term represents the limitation of the new BGP NEXT_HOP (S), the second term the contribution from the new (sent) BGP NEXT_HOP (S) toward the old (received) BGP NEXT_HOP (S’), the third term represent the contribution from the old BGP NEXT_HOP (S’) toward the egress.

255 is a reserved value.

Note that the local RLD is meant as a node value. If a router has multiple line cards with different capabilities, the router SHOULD advertise the smallest one. However, a router MAY choose to only consider the line cards that may be used by the BGP routers receiving the ELC. e.g. if the ELC is advertised over an EBGP session with peer S’, a router MAY consider only the line cards connected to peer S’.

6.2. Entropy Label Next-Hop Capability error handling

If the Entropy Label Next-Hop Capability is present more than once, it MUST be considered as received once with a length of 0.

If the Entropy Label Next-Hop Capability is received with a length other than 0 or 1, it is not considered malformed, but its semantics are exactly the same as if it had a length of 0. This is to allow for graceful future extension.

7. IANA Considerations

7.1. Next-Hop Capabilities Attribute

IANA is requested to allocate a new Path Attribute, called "Next-Hop Capabilities", type Code TBD1, from the "BGP Path Attributes" registry.
7.2. Next-Hop Capability registry

The IANA is requested to create and maintain a registry entitled "Next-Hop Capabilities".

The registration policies [RFC5226] for this registry are:

1-63   IETF Review
64-127   First Come First Served
128-250   Standards Action
251-254   Experimental Use
255   Reserved

IANA is requested to make the following initial assignments:

Registry Name: Next-Hop Capability.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>This document</td>
</tr>
<tr>
<td>1</td>
<td>Entropy Label</td>
<td>This document</td>
</tr>
<tr>
<td>2-250</td>
<td>Unassigned</td>
<td></td>
</tr>
<tr>
<td>251-255</td>
<td>Experimental</td>
<td>This document</td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td>This document</td>
</tr>
</tbody>
</table>

8. Security Considerations

This document does not introduce new security vulnerabilities in BGP. Specifically, an operator who is relying on the information carried in BGP must have a transitive trust relationship back to the source of the information. Specifying the mechanism(s) to provide such a relationship is beyond the scope of this document. Please refer to the Security Considerations section of [RFC4271] for security mechanisms applicable to BGP.

9. Acknowledgement

The Entropy Label Next-Hop Capability defined in this document is based on the ELC BGP attribute defined in section 5.2 of [RFC6790].

The authors wish to thank John Scudder for the discussions on this topics and Eric Rosen for his review of this document.
10. References

10.1. Normative References


10.2. Informative References


Authors’ Addresses

Bruno Decraene
Orange

Email: bruno.decraene@orange.com
Fast Reroute for Node Protection in LDP-based LSPs

draft-esale-mpls-ldp-node-frr-05

Abstract

This document describes procedures to support node protection for unicast Label Switched Paths (LSPs) established by Label Distribution Protocol (LDP). In order to protect a node N, the Point of Local Repair (PLR) of N must discover the Merge Points (MPs) of node N such that traffic can be redirected to them in case of node N failure. Redirecting the traffic around the failed node N depends on existing point-to-point LSPs originated from the PLR to the MPs while bypassing the protected node N. The procedures described in this document are topology independent in a sense that they provide node protection in any topology so long as there is a alternate path in the network that avoids the protected node.

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
1. Introduction

This document describes procedures to support node protection for unicast Label Switched Paths (LSPs) established by Label Distribution Protocol (LDP) [RFC5036]. In order to protect a node N, the Point of Local Repair (PLR) of N must discover the Merge Points (MPs) of node N such that traffic can be redirected to them in case of node N.
failure. Redirecting the traffic around the failed node N depends on existing explicit path Point-to-Point (P2P) LSPs originated from the PLR LSR to the MPs while bypassing node N. The procedures to setup these P2P LSPs are outside the scope of this document, but one option is to use RSVP-TE based techniques [RFC3209] to accomplish it. Finally, sending traffic from the PLR to the MPs requires the PLR to obtain FEC-label bindings from the MPs. The procedures described in this document relies on Targeted LDP (tLDP) session [RFC5036] for the PLR to obtain such FEC-Label bindings.

The procedure described in this document assumes the use of platform-wide label space. The procedures for node protection described in this document fall into the category of local protection. The procedures described in this document apply to LDP LSPs bound to either an IPv4 or IPv6 Prefix FEC element. The procedures described in this document are topology independent in a sense that they provide node protection in any topology so long as there is a alternate path in the network that avoids the protected node. Thus these procedures provide topology independent fast reroute.
1.1 Abbreviations

PLR: Point of Local Repair - the LSR that redirects the traffic to one or more Merge Point LSRs.

MP: Merge Point. Any LSR on the LDP-signaled (multi-point to point) LSP, provided that the path from that LSR to the egress of that LSP is not affected by the failure of the protected node.

tLDP: A targeted LDP session is an LDP session between non-directly connected LSRs, established using the LDP extended discovery mechanism.

FEC: Forwarding equivalence class.

IGP: Interior Gateway Protocol.

BR: Border Router.

3. Merge Point (MP) Discovery

For a given LSP that traverses the PLR, the protected node N, and a particular neighbor of the protected node, we’ll refer to this neighbor as the "next next-hop". Note that from the PLR’s perspective the protected node N is the next hop for the FEC associated with that LSP. Likewise, from the protected node’s perspective the next next-hop is the next hop for that FEC. If for a given <LSP, PLR, N> triplet the next next-hop is in the same routing subdomain (area) as the PLR, then that next next-hop acts as the MP for that triplet. For a given LSP traversing a PLR and the node protected by the PLR, the PLR discovers its next next-hops (MPs) that are in the same routing subdomain (IGP area) as the PLR from IGP shortest path first (SPF) calculations. The discovery of next next-hop, depending on an implementation, may not involve any additional SPF, above and beyond what will be needed by either ISIS or OSPF anyway, as the next next-hop, just like the next-hop, is a by-product of SPF computation.

Also, the PLR may discover all possible MPs from either its traffic engineering database or link state database. Some implementations MAY need appropriate configuration to populate the traffic engineering database. The traffic engineering database is populated by routing protocols such as ISIS and OSPF or configured statically.

If for a given <LSP, PLR, N> triplet the node protected by the PLR is a Border Router (BR), then the PLR and the next next-hop may end up in different routing subdomain. This could happen when an LSP...
traversing the PLR and the protected node does not terminate in the same routing subdomain as the PLR. In this situation the PLR may not be able to determine the next next-hop from shortest path first (SPF) calculations, and thus may not be able to use the next next-hop as the MP. In this scenario the PLR uses an "alternative" BR as the MP, where an alternative BR is defined as follows. For a given LSP that traverses the PLR and the (protected) BR, an alternative BR is defined as any BR that advertises into PLR’s own routing subdomain reachability to the FEC associated with the LSP. Note that even if a PLR protects an BR, for some of the LSPs traversing the PLR and the BR, the next next-hops may be in the same routing subdomain as the PLR, in which case these next next-hops act as MPs for these LSPs. Note that even if the protected node is not an BR, if an LSP traversing the PLR and the protected node does not terminate in the same routing subdomain as the PLR, then for this LSP the PLR MAY use an alternative BR (as defined earlier), rather than the next next-hop as the MP. When there are several candidate BRs for alternative BR, the LSR MUST select one BR. The algorithm used for the alternative BR selection is a local matter but one option is to select the BR per FEC based on shortest path from PLR to the BR.

4. Constructing Bypass LSPs

As mentioned before, redirecting traffic around the failed node N depends on existing explicit path Point-to-Point (P2P) LSPs originated from the PLR to the MPs while bypassing node N. Let’s refer to these LSPs as "bypass LSPs". While the procedures to signal these bypass LSPs are outside the scope of this document, this document assumes use of RSVP-TE LSPs [RFC3209] to accomplish it. Once a PLR that protects a given node N discovers the set of MPs associated with itself and the protected node, at the minimum the PLR MUST (automatically) establish bypass LSPs to all these MPs. The bypass LSPs MUST be established prior to the failure of the protected node.

One could observe that if the protected node is not an BR and the PLR does not use alternative BR(s) as MP(s), then the set of all the IGP neighbors of the protected node forms a superset of the MPs. Thus it would be sufficient for the PLR to establish bypass LSPs with all the IGP neighbors of the protected node, even though some of these neighbors may not be MPs for any of the LSPs traversing the PLR and the protected node.

The bypass LSPs MUST avoid traversing the protected node, which means that the bypass LSPs are explicitly routed LSPs. Of course, using
RSVP-TE to establish bypass LSPs allows these LSPs to be explicitly routed. As a given router may act as an MP for more than one LSP traversing the PLR, the protected node, and the MP, the same bypass LSP will be used to protect all those LSPs.

5. Obtaining Label Mapping from MP

As mentioned before, sending traffic from the PLR to the MPs requires the PLR to obtain FEC-label bindings from the MPs. The solution described in this document relies on Targeted LDP (tLDP) session [RFC5036] for the PLR to obtain such mappings. Specifically, for a given PLR and the node protected by this PLR, at the minimum the PLR MUST (automatically) establish tLDP with all the MPs associated with this PLR and the protected node. These tLDP sessions MUST be established prior to the failure of the protected node. One could observe that if the protected node is not a BR and the PLR does not use alternative BR(s) as MP(s), then the set of all the IGP neighbors of the protected node forms a superset of the MPs. Thus it will be sufficient for the PLR to (automatically) establish tLDP session with all the IGP neighbors of the protected node - except the PLR - that are in the same area as the PLR, even though some of these neighbors may not be MPs for any of the LSPs traversing the PLR and the protected node.

At the minimum for a given tLDP peer the PLR MUST obtain FEC-label mapping for the FEC(s) for which the peer acts as an MP. The PLR MUST obtain this mapping before the failure of the protected node. To obtain this mapping for only these FECs and no other FECs that the peer may maintain, the PLR SHOULD rely on the LDP Downstream on Demand (DoD) procedures [RFC5036]. Otherwise, without relying on the DoD procedures, the PLR may end up receiving from a given tLDP peer FEC-label mappings for all the FECs maintained by the peer, even if the peer does not act as an MP for some of these FECs. If the LDP DoD procedures are not used, then for the purpose of the procedures specified in this draft the only label mappings that SHOULD be exchanged are for the Prefix FEC elements whose PreLen value is either 32 (IPv4), or 128 (IPv6); label mappings for the Prefix FEC elements with any other PreLen value SHOULD NOT be exchanged.

When a PLR has one or more BRs acting as MPs, the PLR MAY use the procedures specified in [draft-ietf-mpls-app-aware-tldp] to limit the set of FEC-label mappings received from non-BR MPs to only the mappings for the FECs associated with the LSPs that terminate in the PLR's own routing subdomain (area).

6. Forwarding Considerations

When a PLR detects failure of the protected node then rather than
swapping an incoming label with a label that the PLR received from the protected node, the PLR swaps the incoming label with the label that the PLR receives from the MP, and then pushes the label associated with the bypass LSP to that MP.

To minimize micro-loop during the IGP global convergence PLR may continue to use the bypass LSP during network convergence by adding small delay before switching to a new path.

7. Synergy with node protection in mLDP

Both the bypass LSPs and tLDP sessions described in this document could also be used for the purpose of mLDP node protection, as described in [draft-ietf-mpls-mldp-node-protection].

8. Security Considerations

The same security considerations apply as those for the base LDP specification, as described in [RFC5036].

9. IANA Considerations

This document introduces no new IANA Considerations.

10. Acknowledgements

We are indebted to Yakov Rekhter for many discussions on this topic. We like to thank Hannes Gredler, Aman Kapoor, Minto Jeyananth, Eric Rosen, Vladimir Blazhkun and Loa Andersson for through review of this document.

11. Normative References


12. Informative References

Esale, et al. Expires September 14, 2017

Authors’ Addresses

Santosh Esale
Juniper Networks
EMail: sesale@juniper.net

Raveendra Torvi
Juniper Networks
EMail: rtorvi@juniper.net

Luyuan Fang
Microsoft
Email: lufang@microsoft.com

Luay Jalil
Verizon
Email: luay.jalil@verizon.com
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.
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 . . . . . . . . . . . . . . . . . . . . . . . . .  4
  1.1. Terminology . . . . . . . . . . . . . . . . . . . . . . . .  6
  1.2. DC and DCI Reference Model . . . . . . . . . . . . . . .  8
2. Requirements . . . . . . . . . . . . . . . . . . . . . . . . . 10
  2.1. MPLS-Based HSDN Design Requirements . . . . . . . . . . 10
  2.2. Hardware Requirements . . . . . . . . . . . . . . . . . . 11
3. HSDN Architecture - Forwarding Plane . . . . . . . . . . . . 11
  3.1. Hierarchical Underlay Partitioning . . . . . . . . . . . . 12
  3.2. Underlay Partition Border Nodes . . . . . . . . . . . . 14
    3.2.1. UPBN and UPBG Naming Convention . . . . . . . . . . . 17
    3.2.2. HSDN Label Stack . . . . . . . . . . . . . . . . . . 17
    3.2.3. HSDN Design Example . . . . . . . . . . . . . . . . . 18
  3.3. MPLS-Based HSDN Forwarding . . . . . . . . . . . . . . . 20
    3.3.1 Non-TE Traffic . . . . . . . . . . . . . . . . . . . . 21
3.3.2 TE Traffic ............................................. 23
4. Scalability Analysis ........................................ 24
  4.1. LFIB Sizing - ECMP ..................................... 24
  4.2. LFIB Sizing - TE ........................................ 25
5. HSDN Label Stack Assignment Scheme .......................... 26
6. HSDN Architecture - Control Plane ............................ 28
  6.1. The SDN Approach ........................................ 28
  6.2. HSDN Distributed Control Plane ............................ 29
7. Security Considerations ...................................... 29
8. IANA Considerations ......................................... 29
9. Acknowledgments ............................................. 30
10. Contributors ................................................ 30
11. References .................................................. 31
  11.1 Normative References .................................... 31
  11.2 Informative References ................................... 31
Authors’ Addresses ............................................. 32
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].

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP</td>
<td>Border Gateway Protocol</td>
</tr>
<tr>
<td>BGP-LU</td>
<td>Border Gateway Protocol Labeled Unicast</td>
</tr>
<tr>
<td>DC</td>
<td>Data Center</td>
</tr>
</tbody>
</table>

Fang et al. Expires <July 26, 2016>
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.

- 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.

- 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      |
|             |
+-------------+

+-------------+ +-------------+
|             | |             |
|     DC      | |     DC      |
|             | |             |
+-------------+ +-------------+

DCI/WAN

Figure 1. DCI/WAN interconnecting multiple DCs.

Figure 2 below illustrates the typical structure of a Clos-based DC fabric.
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 (LNs). 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 leaves) 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
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 certain 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.
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 Level 2, 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.

- 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.

- 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.

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 UFBG 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].
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 ----->|                     |
  LS                        |     MS                           |
```

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,
11. References

11.1 Normative References


11.2 Informative References


[HSDNSOSR15] L. Fang et al., "Hierarchical SDN for the Hyper-Scale,

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
Abstract

This document defines MPLS label forwarding operation with no label swapping as a new MPLS label operation extension to the existing basic forwarding operation of label push, pop, and swap.

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 and License 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
1. Introduction

MPLS forwarding operation as defined in [RFC3031] has three basic actions for the labels at the network nodes: push, swap, and pop. This document describes an additional operation action: label forwarding, referred to as NO SWAP. Currently, using the same label as both incoming and outgoing label is typically achieved by "swapping" the incoming label with an identical outgoing label. In order to improve processing efficiency and memory usage reduction, a simple label forwarding operation with no swap is desirable.

1.1. Background

When MPLS Architecture [RFC3031] was defined, the three types of label operation were sufficient. Label swap operation is performed at a Label Switched Router (LSR) which is not an MPLS edge node, while label push and pop can be performed at an MPLS edge node for label imposition and deposition. Penultimate hop popping can also be performed at the penultimate hop for improved efficiency when appropriate. Since the labels are assigned independently in distributed fashion in a non-traffic engineered basic MPLS networks, it is not possible nor necessary to coordinate the label assignment. Therefore, the label swapping function is sufficient and effective for LSR.

With the increased interests and large scale development of Software-Defined Networking (SDN), central controller assigned MPLS label become one of the option for MPLS based forwarding. A coordinated label assignment can be performed by central controller. Use a single label to traverse multiple hops along the Label Switched Path (LSP) become desirable, therefore the needs to extend the label operation with one more action type - forwarding (without swapping).

The performance and memory efficiency can be increased by performing simple forwarding function than swapping the labels with the identical identifier. This is an optimization requirement, it does not change the fundamentals of MPLS architecture and label encoding as defined by [RFC3031] [RFC3032].

1.2. Use Cases

Hierarchical SDN (HSDN), [I-D.fang-mpls-hsdn-for-hsdc], [HSDNSOSR15] is an architectural solution that achieves hyper scale for Cloud networks using very small forwarding tables in the network nodes. HSDN introduces a new paradigm for the forwarding and control planes - all viable paths in the network are pre-established in the forwarding tables and the labels identify entire paths rather than...
simply destinations. These properties of HSDN dramatically simplify establishing tunnels, and thus enable optimal handling of both ECMP and any-to-any end-to-end traffic engineering, which in turn yields extremely high network utilization with small buffers in the switches. The pre-established tunnels make HSDN the ideal underlay infrastructure to enable seamless and lossless VM and VNF overlay mobility, and achieve excellent elasticity. HSDN brings important simplifications in the control plane and in the architecture of the SDN controller.

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.

Label in HSDN is designed to identify the path and destination, the intermediate nodes (non partition border nodes) simply forms the label forwarding with the same incoming and outgoing label. Swapping the label with the same ID can work, but simple label forwarding without swapping performed by hardware can be more efficient as optimization.

In addition to HSDN, there are other source routed label forwarding solutions may benefit from the label forwarding operation as well.

2. 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].

- Incoming Label Map (ILM): It maps each incoming label to a set of NHLFEs. It is used when forwarding packets that arrive as labeled packets.
- Label forward: The extended label forwarding operation without label swap, consisting of looking up an incoming label to determine the outgoing port, and other data handling information.
- Label forwarding: A simple forwarding paradigm allowing streamlined forwarding of data by using labels to identify classes of data packets which are treated indistinguishably when forwarding without label swapping.
Label swap: The basic existing forwarding operation consisting of looking up an incoming label to determine the outgoing label encapsulation, port, and other data handling information.

Label swapping: A forwarding paradigm allowing streamlined forwarding of data by using labels to identify classes of data packets which are treated indistinguishably when forwarding.

Label Switched Path (LSP): The path through one or more LSRs at one level of the hierarchy followed by a packets in a particular forwarding equivalence class (FEC).

Label Switching Router (LSR): An MPLS node which is capable of forwarding native L3 packets.

MPLS edge node: An MPLS node that connects an MPLS domain with a node which is outside of the domain, either because it does not run MPLS, and/or because it is in a different domain. Note that if an LSR has a neighboring host which is not running MPLS, that that LSR is an MPLS edge node.

NHLFE: Next Hop Label Forwarding Entry

Software-Defined Networking (SDN): an architecture that decouples the network control and forwarding functions to enable the network control to be directly programmable and the underlying infrastructure to be abstracted for applications and network services.

3. Label Forwarding

Label forwarding is the use of the following procedures to forward a packet.

Same as in [RFC3031], in order to forward a labeled packet, a LSR examines the label at the top of the label stack. It uses the ILM to map this label to an NHLFE. Using the information in the NHLFE, it determines where to forward the packet, and performs an operation on the packet’s label stack.

Unlike in label swapping, label forwarding does not remove the incoming label and encodes the new label stack into the packet as in label swapping, it forwards the packet with the same label stack as the incoming stack, to the outgoing interface. Other processing may be involved in selecting the outgoing interface, for example, load balancing through IP deader hashing or use of Entropy label [RFC6790].
4. Security Considerations

The MPLS label forwarding operation specified herein does not raise any security issues that are not already present in either the MPLS architecture [RFC3031] or in MPLS label encoding [RFC3032].

In addition, general MPLS and GMPLS considerations and MPLS security defense techniques are documented in [RFC5920].

5. IANA Considerations

None.

6. References

6.1 Normative References


6.2 Informative References


Authors’ Addresses

Luyuan Fang
Microsoft
5600 148th Ave NE
Redmond, WA 98052
Email: lufang@microsoft.com

Fabio Chiussi
Cisco Systems
500 108th Avenue N.E., Suite 500
Bellevue, WA 98004
Email: fchiussi@cisco.com

Barak Gafni
Mellanox
6 Habarzel St.
Tel Aviv, Israel
Email: gbarak@mellanox.com
Abstract

This document describes a way to apply opportunistic security between adjacent nodes on an MPLS Label Switched Path (LSP) or between end points of an LSP. It explains how keys may be agreed to enable encryption, and how key identifiers are exchanged in encrypted MPLS packets. Finally, this document describes the applicability of this approach to opportunistic security in MPLS networks with an indication of the level of improved security as well as the continued vulnerabilities.

This document does not describe security for MPLS control plane protocols.

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."

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 is.
described in the Simplified BSD License.

Notation

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 ................................................... 3
  1.1. Experimental Status ....................................... 4
  1.2. Existing Security Tools for MPLS Data .................... 5
  2.1. Why Do We Need Opportunistic Security? ................... 5
  2.2. Opportunistic Security at 10,000ft ........................ 7
  2.3. What about a Man-in-the-Middle? .......................... 8
2.4. OS in MPLS Overview ........................................ 9
3. MPLS Packet Encryption ........................................ 11
  3.1. MPLS Encryption Label ..................................... 14
  3.2. Control Word ............................................... 14
  3.3. Considerations for ECMP .................................. 15
  3.4. Backward Compatibility ................................... 16
  3.5. MTU Considerations ........................................ 17
  3.6. Recursive Encryption ...................................... 17
4. Key Exchange For Opportunistic Security in MPLS ............... 17
  4.1. Initiating MPLS-OS ........................................ 18
  4.2. MPLS G-ACh Advertisement Protocol for Key Exchange .... 19
  4.3. Key Exchange Protocol ..................................... 19
  4.4. Indicating the Return Path ................................ 24
  4.5. Protecting the Key Exchange Protocol Messages .......... 25
5. Applicability of MPLS Opportunistic Security ................... 25
  5.1. Tunnel MPLS Packets ....................................... 27
  5.2. Penultimate Hop Popping ................................... 28
6. Security Considerations ....................................... 29
  6.1. Security Improvements .................................... 29
  6.2. Applicability ............................................. 29
  6.3. Continued Vulnerabilities ................................ 29
  6.4. New Security Considerations ............................... 29
7. Manageability Considerations .................................. 30
  7.1. MITM Detection ............................................ 30
8. IANA Considerations .......................................... 30
  8.1. GAP Key Exchange TLV ..................................... 30
  8.2. Key Derivation Functions and Symmetric Algorithms ...... 31
9. Acknowledgements ............................................. 31
10. References .................................................. 31
  10.1. Normative References .................................... 31
  10.2. Informative References .................................. 32
Authors’ Addresses ............................................... 34
1. Introduction

MPLS is an established data plane protocol in the Internet. It is found in the majority of core service provider networks and most end-
to-end traffic in the Internet will be carried over MPLS at some point in its path. The MPLS data plane is defined by [RFC3031] and [RFC3032].

Data security (e.g., confidentiality) in MPLS has previously relied on just two features:

- Physical isolation of MPLS networks has been used to ensure that interception of MPLS traffic was not possible.
- Higher-layer protocol security (such as IPsec [RFC4302], [RFC4303]) has been used whenever a particular flow has determined that security was desirable.

These features have a number of significant vulnerabilities:

- Networks are increasingly easily compromised physically such that "taps" may be inserted in links between routers [RFC7258].
- Routers may be compromised either in their entirety or through the management/control plane (or misconfiguration). This may result in packets being diverted to transit inspection points on their way to their destination.
- The increased support for point-to-multipoint (P2MP) MPLS means that routers can easily be configured (or misconfigured) to make a copy of data and to send it to an additional destination.
- End-to-end payload security may be hard to manage and operate and is not turned on by default by many users. While this form of security is desirable, the network should also improve the security of data transfer that it offers.

The concept of Opportunistic Security (OS) is introduced in [RFC7435]. This document describes an OS design pattern for the MPLS data plane. It shows what part of an MPLS packet may be encrypted and provides a way to indicate that the packet is encrypted as well as to carry a key identifier with each packet.

MPLS opportunistic security can be achieved between adjacent Label Switching Routers (LSRs) on an MPLS Label Switched Path (LSP), and also between end points of an LSP.

This document also provides a mechanism for keys to be exchanged to
facilitate encryption. Finally, this document describes the applicability of OS in MPLS networks with an indication of the level of improved security as well as the continued vulnerabilities.

This document does not describe security for MPLS control plane protocols.

Please note that a discussion of the applicability of MPLS opportunistic security is provided in Section 5.

1.1. Experimental Status

This document is presented as experimental. Before advancing this work on the IETF's Standards Track, it is important to get experience of the practicality of the mechanisms described. In particular whether it is practical to achieve these mechanisms in existing hardware, and whether the imposition of additional MPLS labels is acceptable in the MPLS data plane. Additionally, the consequences of the reduced MTU caused by inserting the additional MPLS label and control word as well as the fact that the encrypted packet will be larger than the unencrypted packet need to be investigated.

It is currently believed that MPLS OS can be deployed progressively without the need to negotiate capabilities outside the key exchange mechanisms described here. This means that no specific walled garden needs to be described in this document.

Experimentation and further investigation of the security aspects of these mechanisms are encouraged especially with regard to mitigation of man-in-the-middle attacks. Consideration of the impact of MPLS OS on MPLS Operations, Administration, and Management (OAM) and other MPLS management techniques also needs further exploration.

The key functions of MPLS OS described in Section 2.4 are based on an initial set of choices that may be adequate for MPLS OS. However, security knowledge is evolving and it may be advisable to "upgrade" for example to Elliptic Curve Diffie-Hellman (ECDH) [RFC6239], using NIST curves or new curves (such as 25519). Furthermore, alternative key derivation functions could be chosen, or symmetric cipher mode could be used. Note that changing to a symmetric cipher that is faster in software, but less likely to be available in hardware would not be a good change.

Section 2.4 also describes the frequency with which keys should be changed. The values described here should be subject to more research and experimentation since key change is fundamental to the actual security of the encryption.
Section 4.3.3 defines the input parameters to the key derivation function and includes the LSP identifier. This identifier is only needed if the scope of the key is per LSP. This document is written on that assumption because of the need to rotate the key after a certain number of packets have been transmitted. However, this could be the subject of some investigation since dropping the LSP identifier would simplify the TLV and the computation. It would also address the issue of identifying the LSP in the case of LDP.

Section 4.3.3 also specifies that the alt is not used. Further investigation is needed to see whether this input parameter would add value.

Note that this experiment uses a special-purpose MPLS label. Since this document is experimental it makes use of an extended special-purpose label from the experimental range. If this work is moved to be published on the standards track, it will be possible to achieve the same function using a simple special-purpose label rather than an extended special-purpose label.

1.2. Existing Security Tools for MPLS Data

This section is a placeholder for text that needs to be added to describe existing security tools for securing MPLS Data. The text needs to describe the use of IPsec used for the payload of MPLS LSPs, and should also cover the use of link layer security (such as MACsec).

>>> TBD

2. Principles of Opportunistic Security

This section provides an overview of opportunistic security in the context of MPLS. Readers are advised to familiarize themselves with some of the attack vectors discussed in [RFC7258] and with the more general description of opportunistic security as described in [RFC7435]. The text here is intended for the consumption of MPLS experts who may not have a background in security: it is, therefore, tutorial and simplistic in nature.

2.1. Why Do We Need Opportunistic Security?

To introduce this discussion we start from a basic view of how encryption is typically used in IETF protocols.

Say we have two protocol entities, Alice and Bob, and they would like some message "M" sent from Alice to Bob to have confidentiality. Alice needs to send M encrypted with algorithm "E" under some
symmetric (e.g., AES) key, "k". Thus Alice wants to send Bob "E(k,M)", but for Bob to be able to understand (i.e., decrypt) the message Alice and Bob both need to agree on the key that will be used: this is called their shared secret.

In many IETF protocols, such as the common usage of Transport Layer Security (TLS) S/MIME Cryptographic Message Syntax (CMS) or Pretty Good Privacy (PGP), Alice simply invents a random key "k" and then encrypts that under Bob's public key "Pub-b" and sends Bob both E(Pub-b,k) and E(k,M). (There are lots of other details and other options for how this can be handled, but we ignore those for now.) In such cases, before Alice can send "E(k,M)", she needs to acquire Bob's public key and she needs to be certain that it really is Bob's public key and not Charlie's. That knowledge requires some long-term key management, which is often done using a Public Key Infrastructure (PKI) so that Alice actually stores the public key (Pub-ca) of a Certification Authority (CA), and Bob gets his public key (Pub-b) "certified" by the CA, which means the CA creates a digitally signed data structure "Cert(Pub-ca,Pub-b)". The crucial thing is that Alice, Bob, and a CA need to co-ordinate before Alice and Bob can agree on a key "k", and that process imposes a key-management burden.

Doing such key management is clearly quite possible, since TLS and IPsec and other well-deployed technologies depend on it. But, in the case of HTTP/TLS on the public web, we see that only roughly 30% of web sites actually take on this burden, even though the software required is ubiquitous and, at least for 2nd level DNS domains in .com for example, there are CAs who offer free domain-validated certificates. While some of the 70% who don’t set up certificates might not actually want confidentiality, there are certainly some who would and arguably many that would benefit from confidentiality, if it just happened out of the box, without an administrator having to do anything. And there are also arguably many other protocols where the same is true.

An alternative to the PKI is manual configuration of keys at Alice and Bob. Manual configuration is used in a large number of cases in deployments, however it has a set of issues that make it problematic. These issues include:
- the scale of configuration that is needed for a full set of Security Associations (SAs) between all communicating parties
- the likelihood of configuration errors
- the security vulnerabilities associated with manual keying and unsecured exchange of keys.

Opportunistic Security (OS) is a protocol design pattern to achieve encryption between Alice and Bob without requiring key-management through CAs and without relying on manual configuration of keys.
2.2. Opportunistic Security at 10,000ft

Instead of the "key transport" mechanisms described in Section 2.1, OS aims to use "key agreement". With key management, Alice invents "k" and safely transports it to Bob encrypted with Bob’s public key as "E(Pub-b,k)". With key agreement, both Alice and Bob contribute to calculating "k" as follows.

Assume that Alice and Bob are using some protocol where they can exchange a few messages in order to agree on the key "k" to use. With a Diffie-Hellman key agreement ("D-H") both Alice and Bob have public and private values, where the private value can be randomly generated, perhaps even once per message "M". They swap the public values, and can then, thanks to the "magic" of Diffie-Hellman, derive a key "k" that nobody else can know.

In this way Alice sends Bob "Pub-a" and Bob sends Alice "Pub-b" and at that point both of them can safely calculate a shared secret "k" from those values. And after that Alice can send Bob "E(k,M)".

From here on, we change the terminology slightly and refer to Alice as the initiator, with private key "i" and Bob as the recipient, with private key "r" so that our notation is closer to that used in IPsec’s Internet Key Exchange Protocol (IKE) on which we model our use of OS.

D-H works as follows: Let "p" be well-known large prime number that we use for all modular arithmetic (meaning that "a\^b" is actually "(a\^b) mod p"), and let "g" be another well-known value (called a generator for the group determined by "p"). Also let Alice and Bob’s private values be "i" and "r" respectively. Now, if Alice sends Bob "g\^i" as her public value, and Bob similarly sends Alice "g\^r" then both of them can easily calculate "g\^(i*r)" or "g\^ir" but nobody else can, since calculating "x" when only given "g\^x" is a computationally hard problem for any "x". Once both Alice and Bob have the value "g\^ir" in hand, they can easily derive a value "k" from that using any of a number of well-known key derivation functions (KDFs) such that k = f(g\^ir) for a KDF "f".

As you can see from the above, Alice and Bob do not need to pre-arrange anything other than "g", "p" and "f", and those can be public information that is used by everyone everywhere (or at least by all participants in a particular deployment). Yet, Alice and Bob have managed to derive a common and private value for a key "k" that they can use to encrypt (and decrypt) "M".

This method of using the OS pattern provides strong confidentiality and can be built into any protocol that allows Alice and Bob to
occasionally exchange public values.

There are also additional advantages to key agreement when compared to key transport. The most important of those is that with key agreement we can easily ensure that \( k \) has a property called Perfect Forward Secrecy (PFS). That means that an attacker has to separately attack each key \( k \). In contrast, if we use the key transport approach, then an attacker who somehow accesses Bob’s private key "Priv-b" can record lots of traffic and later go back and decrypt all the "E(Pub-b,k)" values that all Alices have ever sent to Bob. With key agreement as described, since both Alice and Bob contribute to the value \( k \), and since Alice and Bob will typically periodically generate new private values \( i \) and \( r \) (perhaps even for every single \( M \)), compromise of one party is far less catastrophic, and an attacker who gets access to one private value gets far less benefit.

2.3. What about a Man-in-the-Middle?

OS as described so far is vulnerable to Man-in-the-Middle (MITM) attacks. The problem is that Alice does not know that it was really Bob’s public value that she received; it could have been Charlie’s public value sent by Charlie. And Charlie could also send Bob his public value pretending to be Alice. Now Charlie can share a key with Alice and a key with Bob so that Charlie can sit between Alice and Bob decrypting what he gets from Alice and then re-encrypting it to send to Bob. Neither Alice nor Bob can tell that Charlie is present as a "Man-in-the-Middle" and both Alice and Bob think they are safely exchanging encrypted messages.

A MITM attack like that is bad and making a protocol proof against such attacks comes at the cost of the key-management burden described in Section 2.1. Most IETF protocols to date require that such MITM attacks not be feasible.

However, despite its potential vulnerability to MITM attacks, OS still has value. This value arises because of the difficulty of inserting a MITM actor, and the cost of processing for the MITM in the case of a very large number of relationships. In particular, where the choice is between no encryption (as has been the case for MPLS to date) and OS, it is clear that using OS offers better (although not the best) security.

Consider the case where an attacker taps a link on the path between Alice and Bob. In this case, the attacker can capture every packet between the two parties, and if there is no encryption, can read every message. Furthermore, consider that the attacker could tap a fiber in the core of the network and so capture every packet between
a large number of Alices and their corresponding Bobs. In these
cases, Charlie can operate as a "passive MITM" since all he has to do
is watch the packets.

With OS in use, Charlie is forced to be an "active MITM". That is he
must engage in the D-H exchange between each pair of Alices and Bobs,
and he must must decrypt and encrypt each packet he wants to inspect.
This imposes a higher cost and is especially burdensome if he is
attempting to do it in parallel for lots of Alice/Bob pairs using
lots of different keys and communication sessions.

Furthermore, when D-H is in use for OS, management tools can be used
to detect the presence of Charlie as a MITM. This is because
Charlie has to agree one key "kA" with Alice, and a different key
"kB" with Bob. As far as we know, Charlie cannot arrange that kA
equals kB because both sides contribute to the key value in the D-H
key agreement. That means that if Alice and Bob can check with each
other what value of "k" they are using and the values do not match,
then they know that Charlie is present. What is more, Alice and Bob
can make this check on the value of "k" for any of the "E(k,M)" they
ever exchanged.

Thus, in the case of a fiber tap where many Alice/Bob pairs are
being monitored, it only takes one Alice and Bob to detect the MITM
attack for all Alice/Bob pairs to be alerted to the problem. In
such cases the cost of detection for Charlie may be even greater than
the cost of performing the MITM attack.

Hence we conclude that OS can have considerable value when used in
MPLS networks.

2.4. OS in MPLS Overview

The basic requirement for MPLS-OS is that we want to provide a way
for two MPLS nodes to do a key exchange and to derive a session key
from that to use in MPLS packet encryption.

To do that we use a Diffie-Hellman key exchange as outlined in
Section 2.2. We model this on IKE [RFC7296] using essentially the
same parameters. We feed the shared Diffie-Hellman value, which is
g^ir, into a standard KDF that also takes as input an LSP identifier
(LSP ID) together with the sending and receiving LSR IDs - where the
sending LSR is the point of encryption and the receiving LSR is the
point of decryption such that the pair of LSRs define the SA. These
additional inputs are used to ensure that we end up with different
keys on an LSP even if the same g^i and g^r values are re-used,
however it is RECOMMENDED that fresh values of i and r are used each
time [RFC4086]. The KDF to be used here is as defined in [RFC5869].
The D-H values used MUST be of at least 2048-bits. Implementations MUST support the 2048-bit modular exponentiation (MODP) group from Section 3 of [RFC3526] and SHOULD support the larger MODP groups from [RFC3526].

This document also defines the mechanism used to derive an identifier for a key (the key-id) from the shared Diffie-Hellman value, which is also based on the KDF output. The key will be used with a symmetric encryption algorithm, such as AEAD_AES_GCM_128 (the default, following [RFC5116]).

As with any symmetric block cipher, one should not use the same key for too long. The nonce defined for these keys is derived using a 96 bit counter incremented by one for each encrypted packet. It is critical for security that nonce values MUST NOT be re-used with a given key. (This is an inherent issue with how AES-GCM or any counter mode achieves high performance.)

Accordingly, implementations MUST support mechanisms for key change.

To support key change, this document defines a way for two LSRs using a key on an LSP to agree a new key and to switch over to using that key when desired. That means that implementations MUST be able to handle at least two keys (old and new) for a given LSP. Once a new key has been agreed then it should be used for sending packets; once encrypted data packets protected with the new key have been successfully received, the old key SHOULD be discarded. Section 4 describes how two LSRs agree keys: to agree a new key two LSRs simply run the same key agreement exchange, but this time protected with the old session key as described in Section 4.5. This process can, of course, be repeated any number of times for the same LSP. It is RECOMMENDED that the key on an LSP be changed at least once every day or every 10^6 packets whichever is sooner, and MUST change keys before encrypting 2^64 packets. For an LSP running over a fully-busy 100Gbe interface, we might assume that means roughly 160 million packets per second, or roughly 2^44 packets per day. The 2^64 limit therefore means changing keys daily in the busiest cases of some of the largest current links capacities.

In the event of a key agreement exchange or decryption failure, an alarm MUST be raised to the operator. Default (i.e., node-wide) and per-LSP behavior SHOULD be configurable in this case: actions may include reverting to non-encrypted traffic, re-attempting key exchange, or tearing down the LSP. Note that a simple attack on OS is to tamper with key agreement exchange messages or encrypted packets so that OS fails. Such attacks may be intended to cause the LSP to operate without encryption, so an operator should consider this when setting the behavior in this case.
Section 7.1 also discusses a mechanism that allows a pair of LSRs using OS on an LSP to detect that a MITM attack has happened. For this, we simply define a function of the shared secret, which can be logged and later compared. Note that logging a sample of these "witness" values will likely be sufficient to detect pervasive MITM attacks [RFC7258]. As with the key-id, we base this on the same KDF output.

We might want to consider deriving the witness value from a separate invocation of the KDF that does not depend on the LSP-specific inputs. The benefit from that would be that the same MITM-detection infrastructure could be used for many protocols. However, that would require standardizing a generic D-H MITM-detection protocol, or at least formats, in order to be useful. We also need to consider what additional information needs to be logged with the witness value so that comparisons can easily be made at scale but without creating new privacy-invasive meta-data. That last is not much of an issue for MPLS-OS, but could be elsewhere. At present we do not intend to go for the generic MITM-detection approach, but it is worth considering.

An additional discussion of the applicability of MPLS-OS is found in Section 5.

3. MPLS Packet Encryption

MPLS packets are encrypted according to the mechanisms described in this section.

When an MPLS packet is encrypted, this is indicated by the insertion of a new extended special-purpose label [RFC7274] in the label stack. This is referred to as the MPLS Encryption Label (MEL). The format of the MEL is described in Section 3.1.

The MEL MUST have the bottom of stack bit (the S bit) set and MUST be followed by a pseudowire control word [RFC4385]. The format of the control word is described in Section 3.2.

The remainder of the MPLS packet is encrypted and cannot be parsed without decryption. It needs to be understood, therefore, that the phrase "bottom of stack" refers to the parsable label stack (i.e., those label stack entries that have not been encrypted) and does not indicate the full label stack of the unencrypted packet. Figures 1 and 2 should make this point clear.

Implementations MUST support the AEAD_AES_GCM_128 encryption algorithm, as specified in Section 5.1 of [RFC5116], which is the default algorithm as described in Section 4.3 of this document.
Note that it is critical that a new nonce is used for every encryption. The nonce is an implicit packet counter. The initial nonce value is derived from the HMAC-based Key Derivation Function (HKDF) output (see Section 4.3.2) at key agreement time and the counter is incremented by one for each packet encrypted on the sending side and by one for each packet successfully decrypted on the receiver side.

Although the nonce is not transmitted with the packets, a 16-bit counter carried in the control Word indicates the nonce value modulo 65536. This feature allows a receiving node to quickly spot that a packet has been dropped and resynch its own counter in order to be able to continue to decrypt received packets. In the event that the counter cannot be resynchronized or that more than 65536 packet are lost in one batch the receiver will encounter a decryption error. In this case the receiver may report a general decryption error or may attempt to resynchronize by advancing its own counter in units of 65536 according to the modulo value in the received packet. Note that incrementing the counter in order to test for decryption failure does generate a potential DoS if, e.g., an attacker decrements the nonce-mod-65536 value. Implementations that do such tests SHOULD maintain a small maximum window size beyond which they will cease attempting to decrypt. It could be that throwing an error might be the more effective response if the packet loss rates are expected to be low enough.

It should also be noted that the output from encryption will be 16 octets longer than the input.

The bottom of stack bit is set in the MEL to stop implementations continuing to search down the label stack (which is encrypted) and attempting to use the data as though it was a valid label stack. The control word is needed because many implementations that find the bottom of stack expect the next bytes to be a control word or protocol indicator.

The position of the MEL and control word depend on whether hop-by-hop or end-to-end encryption is being applied.

Figure 1 illustrates the format of an example MPLS packet before and after hop-by-hop encryption. The left hand part of the figure shows a normal MPLS packet with a label stack and payload. The bottom label in the stack has the S bit set. The payload is the data carried by the MPLS packet (such as IP) and may be prefixed by a control word.

The right hand part of Figure 1 shows the same packet after it has been encrypted. The top of stack is a label with value 15 that
indicates that an extended special-purpose label follows. Next comes the MEL with the S bit set. The label value of the MEL is from the experimental range 240-255 and is selected according to the scope of the MPLS OS experiment being run. The MEL is followed by a control word. Everything that follows the control word is the entire original MPLS packet encrypted.

```
| Top Label | . | Label 15 |
|-----------+  .  |-----------|
| Label     | . | MEL   S=1 |
|-----------+    . +-----------+
| Label S=1 | . | Ctrl Word |
|-----------+      +-----------+
| Payload   | . | Encrypted |
|-----------| |          |
```

Figure 1: The Use of the MEL for Hop-by-Hop Encryption

Figure 2 illustrates the format of an example MPLS packet before and after end-to-end encryption. The left hand part of the figure shows a normal MPLS packet with a label stack and payload. The bottom label in the stack has the S bit set and the payload may be prefixed by a control word. The right hand part of the figure shows how the top two labels (or however many labels are needed for end-to-end delivery) remain at the top of the label stack. Then follows label 15 to indicate that an extended special-purpose label follows, then

```
<table>
<thead>
<tr>
<th>Top Label</th>
<th></th>
<th>Top Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td></td>
<td>Label</td>
</tr>
<tr>
<td>-----------</td>
<td>--</td>
<td>-----------</td>
</tr>
<tr>
<td>Label</td>
<td></td>
<td>Label 15</td>
</tr>
<tr>
<td>-----------</td>
<td>--</td>
<td>-----------</td>
</tr>
<tr>
<td>Label</td>
<td></td>
<td>MEL   S=1</td>
</tr>
<tr>
<td>-----------</td>
<td>--</td>
<td>-----------</td>
</tr>
<tr>
<td>Label S=1</td>
<td></td>
<td>Ctrl Word</td>
</tr>
<tr>
<td>-----------</td>
<td>--</td>
<td>-----------</td>
</tr>
<tr>
<td>Payload</td>
<td></td>
<td>Encrypted</td>
</tr>
<tr>
<td>-----------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 2: The Use of the MEL for End-to-End Encryption
comes the MEL with S bit set, and a control word. The remainder of
the packet is encrypted and contains the rest of the label stack and
the payload.

3.1. MPLS Encryption Label

The MPLS Encryption Label (MEL) is a normal label stack entry
carrying an extended special-purpose label with a value from the
experimental range 240-255. The format of the label stack entry is
defined in [RFC3032] and shown in Figure 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Label                  | TC  |S|       TTL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3 : Format of the MEL Label Stack Entry

Label: The value of MEL for this experiment

TC: For end-to-end encryption, the value of the TC field SHOULD
be set to the value of the unencrypted label stack entry that
immediately precedes the MEL. In the case of hop-by-hop
encryption, the value of the TC SHOULD be copied from the TC
of the first encrypted label if there is a label stack
present. Otherwise this field SHOULD be set to all zero
(0b000).

S: MUST be set to one.

TTL: SHOULD be set to two to prevent encrypted packets being
accidentally forwarded too far beyond the point of intended
decryption. Note that setting to zero might cause a
receiver to discard the packet when the MEL becomes top of
stack, and setting to one might cause the packet to be sent
to the slow path when the MEL becomes the top of the stack
even though decryption should be a fast-path function.

The sending LSR MAY choose different values for the TTL and TC fields
if it is known that label 15 or the MEL will not be exposed as the
top label at any point along the LSP (for example, by penultimate hop
popping - but see Section 5 for a discussion of MPLS tunnels and
penultimate hop popping).

3.2. Control Word

The control word is inserted after the MEL as described in Section 3.
The S bit set to one in the MEL and the presence of the control word
helps protect against transit nodes that may perform hashing or
inspection of the label stack and payload packet headers when forwarding MPLS packets (for example, to enable ECMP). The control word indicates that the payload is not a protocol that can be meaningfully hashed or inspected.

The format of the control word is defined in [RFC4385] and shown in Figure 4.

```
| 0 0 0 0 | Flags | FRG | Length | Sequence Number |
```

Figure 4: Control Word for Encrypted MPLS

- **Flags:** The Flags field is treated as a four-bit number. It contains the key-id that identifies the algorithm and key as established through configuration or dynamic key exchange as described in Section 4.
- **FRG:** Must be sent as 0, and ignored on receipt. Fragmentation is not used.
- **Length:** MUST be sent as 0, and ignored on receipt.
- **Sequence Number:** This field contains the packet counter (nonce) for the encryption algorithm and key currently in use modulo 65536. It can be used by a receiver to quickly check that the value of the nonce being used for decryption is likely to be correct as described in Section 3.

### 3.3. Considerations for ECMP

As previously stated, the S bit set in the MEL and the presence of the control word prevent implementations from attempting to use the encrypted MPLS packet and its payload to determine a hash value for uses such as ECMP. However, the resultant label stack shown in Figure 2 will probably not provide sufficient entropy for ECMP purposes.

In order to increase the entropy, an implementation that inserts an MEL and MEL MAY also insert an Entropy Label Indicator (ELI) and Entropy Label (EL) as defined in [RFC6790] ELI and EL are positioned in the label stack before the MEL as shown in Figure 5. The setting of the fields in the ELI and EL label stack entries are as described in [RFC6790].

The ELI and EL will normally occur immediately before the label 15 and MEL pair, but they MAY be placed higher up the label stack.
3.4. Backward Compatibility

Keys and encryption algorithms may be configured manually or exchanged dynamically as described in Section 4. These mechanisms provide a preliminary way to protect against a sender encrypting data that the receiver cannot decrypt, however, misconfiguration may lead to a sender using the MEL when the receiver does not support encryption.

When a node finds an unknown label at the top of the label stack it must discard the packet as described in [RFC3031]. Therefore, when a receiver discovers label 15 and does not support extended special-purpose labels it will discard the packet. Similarly when a receiver that supports extended special-purpose labels, but does not support the MEL (i.e., does not support encryption) it will discard the packet. (Note that care must be taken if multiple experiments are being carried out in the same network since a different extended special-purpose label must be used for each experiment.) The net result is that when a sender uses encryption in error, all packets that it sends on the LSP will be discarded by the receiver. Note that in this discussion, "the receiver" may be the next hop if single hop encryption is used, or may be the end of the LSP if end-to-end encryption is used.

Transit nodes that are not actively participating in the encryption will not inspect the MEL except potentially as part of an ECMP hash,
and it should be noted that the use of Special Purpose Labels in hashing is strongly discouraged (see Section 2.4.5.1 of [RFC7325]). This means that transit nodes will not encounter the MEL during normal packet processing and will not discard packets.

3.5. MTU Considerations

Adding label 15, the MEL, and the Control Word as described above will reduce the available data size by 12 octets. Furthermore, as described in Section 3, the output of the encryption algorithm is at least 16 octets longer than the input. Therefore, the use of encryption reduces the available MTU by at least 28 octets. Other encryption algorithms may result in even greater reductions in the available MTU.

When end-to-end encryption is in use this can be considered by the ingress LSR, however, when single-hop encryption is in use the participating LSRs need to advertise this reduction in link MTU so that the packets do not overflow. MPLS packets MUST NOT be fragmented as a result of encryption.

3.6. Recursive Encryption

The use of MEL and control word described in Section 3 may be applied recursively. That is, the payload of an encrypted MPLS packet may, itself be an encrypted MPLS packet. This may be particularly useful in the case where an MPLS VPN has native MPLS traffic.

There are no special considerations except to note that encryption and decryption processing may be burdensome if an LSP and its payload LSP have encryption applied at the same LSR. Additionally, it should be noted that, as described in Section 3.6, each recursive encryption reduces the MTU by 28 octets.

4. Key Exchange For Opportunistic Security in MPLS

For encryption to be useful both ends of an encrypted session must know which algorithm is in use and which key to use. The mechanism described in Section 3 provides a way to indicate an index into a table of algorithms and keys that can be used to decrypt an encrypted MPLS packet.

It is possible that this table has been manually configured or set up using a key exchange protocol such as Internet Key Exchange version 2 (IKEv2) [RFC7296]. However, such a process implies a stable security association between encrypter and decrypter of MPLS packets. While such a stable association is entirely consistent with the concept of OS, OS nonetheless calls for a more dynamic key agreement method.
This section provides a mechanism for adjacent MPLS LSRs, or for a pair of LSRs at opposite ends of an MPLS LSP, to dynamically exchange keys and algorithm identifiers so that encryption may be applied opportunistically.

The mechanism uses message exchanges in the MPLS Generic Associated Channel (G-ACh) [RFC5586] as part of the MPLS Generic Associated Channel (G-ACh) Advertisement Protocol (GAP) [RFC7212]. This channel is in-band with an LSP and may be used to carry messages between neighbors or between the end points of the LSP. A type field within the common message header, the Associated Channel Header (ACH), is used to indicate the type of message carried.

Nodes that receive G-ACh messages and do not understand them, or nodes that understand the G-ACh but do not recognize the ACH message type drop the packets as described in [RFC5586].

Note that this mechanism may benefit from encryption that is already in use on an LSP. Thus key changes using this mechanism can be made using encrypted messages.

4.1. Initiating MPLS-OS

This document assumes that the use of MPLS-OS is initiated by the upstream of a pair of LSRs (either a pair of adjacent LSRs on an LSP, or a pair of LSP end points). That is, the upstream LSR send the first G-ACh that initiates key exchange. The key that is generated after the exchange is then used to encrypt traffic travelling in the direction between initiating and responding LSRs: that is, from upstream to downstream LSR.

In the case of a bidirectional LSP, each direction is treated separately. That is, "upstream" refers to the direction of traffic flow, and not to any signaling that is used to establish the LSP. Thus, it is possible that a bidirectional LSP uses MPLS-OS on none, either one, or both of the directions of traffic flow for the LSP. But it is important to note that the keys used are different in each direction, each being generated and exchanged through a separate instance of the procedures described in this document. Note that the input parameters for key derivation listed in Section 4.3.3 show LSP identifier, initiator LSR identifier, and responder LSR identifier as three of the ordered list of pieces of information used by the key derivation function. In the case of a bidirectional LSP, the LSP identifier will be the same in each direction, and the two LSR identifiers will be the same, but the LSR identifiers will be used in the reverse order at the two end points of the MPLS-OS exchange and this will reduce the chance of the same key being used in each direction.
Note also that in the case of a pair of unidirectional LSPs in reverse directions between a pair of LSRs there should be no relationship between the keys used on each LSP even if there is a tight coupling between the LSPs such as might be the case for associated bidirectional LSPs [RFC7551]. The key derivation function will use different LSP identifiers as well as using the LSR identifiers in a different order.

4.2. MPLS G-ACh Advertisement Protocol for Key Exchange

GAP defines messages exchanged in G-ACh on a common Associated Channel Type code point (0x0059) [RFC7212]. The application for which the messages are exchanged is defined by the Application ID field carried in the Applications Data Block (ADB). MPLS OS capability notification and key exchange uses the GAP Application ID (0x0000) defined by [RFC7212] and a new ADB TLV for MPLS OS.

Implementations that do not support GAP will discard received packets with this Associated Channel Type as described in [RFC5586]. Implementations that support GAP but that do not support key exchange will discard received packets with this ADB TLV as described in [RFC7212]. Either of these discards will result in no dynamic key exchange, but other key definitions are still supported (such as manual configuration) and may be used to construct a table of algorithms and keys that can be used to achieve MPLS encryption using the mechanisms described in Section 3.

4.3. Key Exchange Protocol

4.3.1. Communication Channels

The key exchange protocol described in this document uses a D-H exchange that assumes a bidirectional communication channel. GAP is designed to run over a unidirectional channel and uses normal IP forwarding for return path messages with an optimization to use the return path of a bidirectional LSP. However, LSPs in packet networks are usually unidirectional. That means that, while the key exchange messages can be sent on the LSP in one direction, a channel needs to be established for the return messages.

This document uses a process similar to that defined for MPLS LSP Ping [RFC4379] and [RFC7110], and that described to indicate a return path for MPLS performance measurement in [I-D.ietf-mpls-pm-udp-return]. That is, the forward message is sent on the LSP and includes the identity of the return path communication channel. The return path may be indicated as a UDP communication over IP, as an LSP running in the opposite direction, or as the reverse direction of a bidirectional LSP.
Note that the GAP messages defined in [RFC7212] include a TLV that enables authentication. This feature SHOULD be used if possible, but it is in the nature of opportunistic security that the necessary security association might not exist. In this case the ability to tamper with the instructions that select a return path may provide a mechanism that makes MITM attacks easier. An implementation that initiates key exchange for MPLS Opportunistic Security MUST verify that the response messages are received on the expected return path channel and SHOULD raise an operator alert if the channel is unexpected.

4.3.2. Key Exchange Messages

The format of a GAP message is described in [RFC7212]. When used for key exchange the GAP message includes an ADB with the fields set as follows.

- **Application ID** is set to 0x0000.
- **Element Length** is set to the total length in octets of this ADB including the Application ID and this field.
- **Lifetime** field SHOULD be set to zero and MUST be ignored.

A key exchange ADB MUST include a Key Exchange TLV as shown in Section 4.3.3. The ADB and MAY also include an Authentication TLV as described in [RFC7212] to provide authentication and integrity validation for a GAP message (see Section 4.5). Additionally, the ADB MAY include a Source TLV as described in [RFC7212] and discussed in Section 4.4.

4.3.3. Key Exchange TLV

A session key is to be established between an initiator (Alice) and a recipient (Bob). The D-H public value for Alice is $g^i$ and for Bob, $g^r$. The shared Diffie-Hellman value is $g^{ir}$. $g^{ir}$ is represented as a string of octets in big endian order padded with zeros if necessary to make it the length of the modulus. Both $g^i$ and $g^r$ will be 2048 bits long, if the Diffie-Hellman modulus is 2048 bits long.

The Key Exchange TLV is modelled on that from Section 3.4 of [RFC7296] with the addition of information to identify the LSP and its return path, and is encoded as shown in Figure 6.
Figure 6 - Key Exchange Message TLV

Type is set to TBD1 to indicate that this is a Key Exchange TLV.

The Reserved and Length fields are defined in [RFC7212].

The flag D denotes the direction of the message, ‘0’ indicates a message from initiator (Alice) to recipient. ‘1’ indicates the reverse direction.

The Rsvd bits are reserved. They SHOULD be set to zero and ignored on receipt.

The Return field is used on a message from the initiator to indicate the type of return path to be used for messages from the responder. The Path Identifier field is interpreted in this context. Possible values are as follows:

0  The reverse path of a bidirectional LSP is to be used for the response. Used on a message from an initiator.
1  The reverse path messages are to be sent encapsulated in UDP. Used on a message from an initiator.
2  Any LSP between the recipient and the initiator may be used.
3  Any LSP between the recipient and the initiator that is already using MPLS-OS may be used.
4  The reverse path messages are to be sent on a specific LSP.

All other values are undefined and MUST be processed as an encoding error as described in Section 4.3.4. Similarly, if the value zero is used on a unidirectional LSP then it MUST be handled as an encoding error.
The Path Identifier is interpreted in the context of the Return field. The field only has meaning on messages from the initiator and SHOULD be ignored on responses. If the Return is set to the following values, the Path Identifier has the following meaning:

0 In this case the Path Identifier field has no meaning and SHOULD be ignored.
1 The Path Identifier field contains a UDP port number from the dynamic port range that the initiator will listen on for a response.
2 In this case the Path Identifier field has no meaning and SHOULD be ignored.
3 In this case the Path Identifier field has no meaning and SHOULD be ignored.
4 The Path Identifier field contains an LSP-ID that must be used for reverse path messages.

See Section 4.4 for more discussion of return paths.

The LSP-ID parameter indicates the LSP to which this key exchange applies. On messages from initiator to recipient this field MUST be set to the LSP on which the message flows and any mismatch MUST be treated as an encoding error (Section 4.3.4). On messages from recipient to initiator, this value MUST be copied from the received message and an initiator that cannot match the message and LSP-ID to a message that it previously sent MUST treat the situation as an encoding error.

The Algorithm field is a one octet field that specifies both the KDF to use and the symmetric algorithm to be used for data packet encryption. A registry for values of this field is defined in Section 8.2. The value 0 is used to indicate the default KDF and symmetric encryption mode. An implementation receiving a value for an Algorithm it does not support MUST treat the case as an encoding error as described in Section 4.3.4. All implementations MUST support the default KDF. Note that since implementation of encryption and decryption is likely to be implemented in hardware for reasons of data throughput performance, the introduction of new algorithms may be bound by firmware or even hardware upgrades.

The Diffie-Hellman Group Num is from [RFC3526], so the group number for 2048 MODP is decimal 14. Note that this is a one octet field, but is two octets in the [RFC7296] equivalent. This is not an issue because there are only 30 MODP groups defined at present and new groups are not added frequently.

The D-H public value will contain g^i or g^r depending on the direction (i.e., the setting of the D flag) and is in big endian.
order.

The length of the Diffie-Hellman public value for MODP groups MUST be equal to the length of the prime modulus over which the exponentiation was performed, prepending zero bits to the value if necessary.

Once both sides have derived \( g^{ir} \) they need to feed that and the other inputs described in Section 2.4 into the KDF indicated by the algorithm field. With the default algorithm (value zero), the KDF to be used is HKDF as specified in [RFC5869].

The parameters for the use of HKDF are:

- **Hash**: SHA-256
- **Salt**: Not used
- **Skip**: Do not skip
- **Info**: The concatenation of a fixed string indicating use of MPLS-OS, with the value "MPLS-OS", the first 32 bits of the key exchange message, with the D flag set to 0, plus the LSP ID and the sender and receiver LSR IDs in that order. That is:

  \[
  \text{MPLS-OS} \| 0 \| \text{payloadLen} \| \text{alg} \| \text{group Num} \| \text{LSP-ID} \| i-\text{LSR-ID} \| r-\text{LSR-ID}
  \]

  \[L: \text{The output length in bits is 272.}\]

The fixed string "MPLS-OS" is used as an input here to prevent potential cross-protocol attacks. Those might otherwise be possible if this mechanism were to be copied in other protocols. (If copying this mechanism for any reason, then a different fixed string value should be used.)

LSP-ID is a unique identifier shared between the initiator and receiver (Alice and Bob) that uniquely identifies the LSP.

[[If RSVP-TE is used for signaling, the LSP-ID is known along the LSP and at the two end points. Similarly, the LSP-ID is known if the LSP is manually configured. It is not so clear how the LSP-ID is known for LSPs established using LDP, although possibly we could use the FEC as defined for RFC 4379 and its extensions.]]

i-LSR-ID and r-LSR-ID are the LSR-IDs of the initiator and receiver respectively (Alice and Bob), where an LSR-ID is the 32 bit, globally unique identifier of the LSR as described in [RFC5036] and [RFC4990].
superior security.

The default encryption algorithm, AEAD_AES_GCM_128, specified in Section 3, requires a 128 bit session key.

The 272-bit HKDF output is the catenation of the session key, the key-id, the witness value and the high-order 16 bits of the initial nonce value in that order. That is the session key is the leftmost 128 bits of the HKDF output. The key-id is the next 4 bits, the witness value is the next 124 bits and the last 16 bits are the 16 most significant bits of the initial nonce value. The low order 64 bits of the initial nonce value are set to zero before the first call to the AES-GCM encryption function. The key-id is carried in encrypted packets as described in Section 3.2.

Note that a 4 bit key-id is adequate in a system where, for any one LSP there is one active key and one new or replaced key. There might also be more than one algorithm, and it is possible that new keys need to be pipelined if roll-over is frequent. In the case that a newly-generated key-id is already in use, the key-id value is repeatedly incremented (modulo 16) until an unused value is found. If all 16 values are already in use, the key derivation function should not be executed.

4.3.4. Encoding Errors

Unknown values in received key Exchange TLVs MUST be treated as encoding errors. All messages that constitute encoding errors MUST be silently discarded. That is, such errors MUST NOT cause response messages to be sent since those messages could be used as part of an attack to determine the capabilities of an LSR.

An LSR SHOULD log such errors and notify the operator. However, care is needed even in these actions since they may be externally visible.

4.4. Indicating the Return Path

The key exchange for MPLS-OS requires a two-way exchange of messages. The Return field of the Key Exchange TLV indicates the reverse path to use for key exchange messages relevant to a particular LSP.

Whenever the LSP being secured is bidirectional, the same LSP SHOULD be used for reverse path messages. Otherwise, the initiator selects the communication channel as described in Section 4.3.3.

If UDP is being used and it may be unclear to what address the messages should be sent, the initiator MUST include a Source Address TLV [RFC7212] to provide this information.
Operators should consider the security implications of the return path. The use of an already-secured LSP (Return type 3) may provide
Implementations MUST make the choice of return path request sent by an initiator available as a configuration option. As noted in Section 4.3.1, the fact that the initial GAP messages might not be protected means that there is the potential to tamper with the instructions that select a return path. This could be used as a vector for MITM attacks. To protect against this, an implementation that initiates key exchange for MPLS Opportunistic Security MUST verify that the response messages are received on the expected return path channel and SHOULD raise an operator alert if the channel is unexpected. In these circumstances an implementation MAY be configured to abort establishment of MPLS-OS although, since that in itself is an attack vector, it is RECOMMENDED that implementations continue toward the use of MPLS-OS while notifying the operator.

4.5. Protecting the Key Exchange Protocol Messages

GAP includes an Authentication TLV that can be used to protect GAP messages as described in [RFC7212]. If there is already an SA between the initiator and recipient this TLV SHOULD be used. However, it is probable with MPLS-OS that no such SA exists and the point of the mechanisms described in this document is to exchange keys in that case, therefore, it is quite likely that the Authentication TLV cannot be used on the first GAP exchanges.

As described in Section 2.4, once one key exchange has been successfully completed, further key exchanges should be protected using a previous key. This is simply achieved since key exchange messages are, themselves, carried in MPLS packets on the LSP and may be subject to encryption exactly as any other packet.

Furthermore, once keys have been established, they may also be used in the GAP Authentication TLV.

5. Applicability of MPLS Opportunistic Security

MPLS-OS provides another tool in the security and privacy toolkit. It is not a panacea and does not solve (nor is it intended to solve) all security or privacy problems. In particular, the use of MPLS-OS does not protect user-data end-to-end that might be better secured using encryption at the IP layer or at higher layers.

As noted throughout this document, the intention of OS in MPLS is to allow one LSR to enable encryption between itself and its neighbor, or between itself and the other end of an LSP, in a dynamic and unplanned way. This can have benefits in a number of scenarios where
the network that generates MPLS traffic transmits it over another network (for example, carrier’s carrier, or some deployments of enterprise network). Additionally, the use of MPLS-OS might allow a service provider to offer a secure edge-to-edge service for a variety of applications ranging from VPNs through pseudowires and where the payload traffic might not always be IP. Lastly, in some non-traditional carriers the user data belongs to the operator or is the direct responsibility of the operator (for example, in data centers, or in large-scale private networks).

As with all security mechanisms, there is a trade-off between a number of factors. On one side is the completeness of the security of the user-data, and on the other side is the complexity of configuring and managing the necessary security associations. Furthermore, while mechanisms closer to the end-user than MPLS-OS (for example, TLS and IPsec in tunnel mode) provide better security for user-data by virtue of not transmitting the data across any network hops without it being encrypted, such mechanisms often expose more metadata for inspection by snoopers within the network.

Additionally, while a variety of per-link encryption mechanisms exist and could be used to guard against attacks such as fiber taps, those approaches do not protect against subverted nodes (i.e., routers) on the path since, by definition, per-link encryption does not protect packets once they come off the link. MPLS-OS in the end-to-end LSP mode protects packets on the links and as they cross transit routers.

Nevertheless, it is not the purpose of this document to recommend the use of MPLS-OS to the exclusion of all other encryption techniques. As already mentioned, MPLS-OS is offered as another tool in the tool kit and users as well as network operators are strongly advised to consider using a variety of tools to achieve the level of security and privacy that they desire.

Note that, in order that OS can be used, one end of a peering (neighbor or LSP end) must decide to attempt OS and the other end must support it. This can be determined by the message exchanges described in Section 4.3 since if one peer does not send a key exchange message then encryption will not be used, and if the other peer does not respond then it is unwilling or unable to decrypt messages.

MPLS-OS should be applicable to all forms of MPLS. That is, it should be possible to use it in RSVP-TE systems, in LDP systems, and in MPLS-TP systems (by which we mean those that have manually configured LSPs). Equally, it should work for point-to-point (P2P) and multipoint-to-point (MP2P) uses of MPLS because there is a simple relationship between the sender (encrypter) and the receiver.
(decrypter) in both cases. In the MP2P case, the sender’s identity can be extracted from the key identifier and there are considered to be enough key identifiers to allow an arbitrary number of senders on the LSP. There will, however, be the need for the receiver to hold OE state (keys, packet counters) for each sender which may be a significant amount of data for an MP2P LSP (although no more than if the same LSP were replaced by multiple P2P LSPs). Additionally, it should be noted that not only will each sender on an MP2P LSP have a different key, but each may separately decide whether to encrypt data or not.

At this time it is not certain whether MPLS-OS can be applied to a point-to-multipoint (P2MP) or a multipoint-to-multipoint LSP in its entirety because packet replication cannot handle the necessary key conversions for each receiver. However, MPLS-OS can certainly be applied to individual hops on these LSPs. Further work is needed to determine whether non-branching multi-hop segments of P2MP and MP2P LSPs can also be protected using MPLS-OS.

5.1. Tunnel MPLS Packets

Note that in the case of tunneling of MPLS packets in another technology (such as MPLS-in-UDP [RFC7510]) there are two approaches that are viable:

- The payload of the encapsulation (i.e., the entire MPLS packet) can be encrypted using the mechanisms described in this document without any changes. Any payload identifier in the encapsulation header can remain set to "MPLS" since the encrypted packet is always just an MPLS packet.

- The encryption mechanisms present in the encapsulating technology can be used without any need to use the mechanisms described in this document.

In some cases that processing of one label on the label stack depends on the values contained in the previous label stack entry. For example, in the "Pipe Model" [RFC3270], the Diff-Serv treatment of the packet that is forwarded beyond the end of the tunnel depends on the setting of the TC field in the previous label stack entry. This requires that when a label is popped, the value of the TC field in the label stack entry is cached for use while forwarding. In the case that the next label on the stack is the MEL, decryption of the rest of the packet is required, and this caching would be a little more complicated to implement. This situation is mitigated by setting the TC field of the label stack entry that contains the MEL to the value from the preceding label stack entry as described in Section 3.1.
The "Short Pipe Model" [RFC3270] can be handled using a combination of the above technique and the procedures described in the next section.

5.2. Penultimate Hop Popping

In penultimate hop popping (PHP) a label is removed from the label stack of a packet one hop before the end of the LSP. The packet is forwarded as though it was still carried on the LSP, but the label stack entry for the LSP is removed. Sometimes we say that packet uses the "implicit null label".

When there are additional subsequent labels on the label stack, this has no impact on the use of the mechanisms described in this document. It is possible that after PHP the MEL will become the top label in the stack meaning that the received packet may encounter the MEL as the top label. This has implications for the setting of the TC and TTL fields in the MEL label stack entry as described in Section 3.1.

However, in some cases of PHP the popped label is the bottom of the label stack and the packet after the popped label is some non-MPLS payload protocol (such as IPv6). PHP is used specifically because the receiving interface does not have MPLS capabilities in the forwarding plane. In this situation the packet is identified within the link encapsulation on the final hop by its payload protocol type (such as IPv6). If MPLS-OS is used this situation will change because even when the final label is stripped using PHP there will remain a MEL entry in the label stack. Therefore the packet will need to be identified as "MPLS" in the link encapsulation on the final hop, yet the receiver might not be capable of handling MPLS packets.

This problem can be approached in two ways:

- The penultimate hop may note the presence of the MEL during PHP and attempt to remove the MEL as well. This is unlikely to be successful as the encryption negotiation has been conducted between the end points of the LSP and the penultimate hop is not aware of the keys or algorithms needed for decryption.

  Furthermore, this approach would leave the packet unencrypted on its final hop which may be counter to the intent of the LSP end points.

- The end point of the LSP should recognize that it cannot have both MPLS-OS and PHP. Indeed, in agreeing to the use of MPLS-OS the end point is making a statement about its ability to handle the MEL and
so it can choose:

- to request PHP and allow the penultimate hop to set the payload indicator of the link encapsulation header to "MPLS"; or

- to not request PHP.

6. Security Considerations

6.1. Security Improvements

See Section 2.1.

6.2. Applicability

See Section 5.

6.3. Continued Vulnerabilities

The mechanisms described in this document do not provide protection against certain types of MITM attacks. For example, the key exchange protocol in Section 4.3 will not detect if key exchange messages or their responses are intercepted and discarded such that the initiating peer believes that encryption is not supported. Similarly, those messages may be tampered with such that a receiver cannot determine the correct mapping of table index to algorithm and key when an encrypted packet is received. Furthermore, the MEL in an MPLS packet is not protected and may be overwritten such that a receiver is unable to decrypt the packet.

See Section 7.1 for a discussion of how active MITM attacks can be detected.

6.4. New Security Considerations

If a pair of LSRs do not do the key exchange before sending any data packets on the LSP then those first packets will not be protected by OS and hence will be available to a monitor.

If a MITM can prevent the OS key exchange from completing, e.g. via deleting messages or changing bits in messages, and if the LSRs continue to send data regardless then those data packets will be available to a monitor. That is, in simple terms, a MITM attacker is able to prevent OS from being used through a very simple attack, and the only options for the end points in this situation are to send no data or to send data in the clear. Again, it should be pointed out that this occurrence is not worse than not running OS at all, but has the benefit of being detectable by end points. See Section 2.4 and
Section 7.1 for a description of how alarms should be raised in these circumstances. Furthermore, Section 4.3.1 and Section 4 describe how the return path for key exchange messages might be hijacked to better facilitate MITM attacks and indicates how the initiator of MPLS-OS can detect this and what actions it should take.

Thus, as been previously noted, OS is not a cure for all ills or a prevention against all attacks, but it does offer a way to increase security in some circumstances.

7. Manageability Considerations

As described in Section 2.4 node-wide and per-LSP behavior SHOULD be configurable to describe the action where key agreement exchange or packet decryption fails. In any case, such events MUST trigger alarms to the operator.

7.1. MITM Detection

Section 2.4 introduces the concept of a function of the shared secret that can be compared by two LSRs that are using OS to see whether they are victims of an active MITM attack.

Section 4.3 describes how a witness value is derived for the default KDF, HKDF.

The participating LSRs can simply log this value plus the LSP and LSR IDs from time to time and a management application can compare the values. If they are different for the same LSP ID, then an active MITM attack has taken place.

It needs to be carefully noted that the management channel used to log or otherwise compare the witness values from the two LSRs MUST be secure. It is likely that routers use relatively high security management channels for configuration and other management operations.

8. IANA Considerations

8.1. GAP Key Exchange TLV

IANA maintains a registry called "Generic Associated Channel (G-ACh) Parameters" with a sub-registry called "G-ACh Advertisement Protocol Application Registry" from which new assignments may be made through the "IETF review" allocation policy [RFC5226]. IANA is requested to make a new allocation as follows:
IANA maintains a registry called "Generic Associated Channel (G-ACh) Parameters". IANA is requested to create a new sub-registry called "G-ACh Advertisement Protocol: MPLS Encryption Algorithms Registry" with new values to be assigned through "IETF Review" as defined in [RFC5226].

The available range is 0 - 255.

IANA is requested to record the following information and create an initial entry as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Key Derivation Function</th>
<th>Symmetric Algorithm</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HKDF</td>
<td>AEAD_AES_GCM_128</td>
<td>[This.I-D]</td>
</tr>
<tr>
<td>1-255</td>
<td>Unassigned</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9. Acknowledgements

Many thanks to Alia Atlas for detailed discussion of the implications and mechanisms of MPLS opportunistic security. Thanks also to Ron Bonica for encouraging this work, to Sean Turner and Stewart Bryant for early review, and to Jeff Haas, Eric Rosen, and Ross Callon for discussions. Thanks for MPLS Review Team comments from Mach Chen and Lizhong Jin, and to Charlie Kaufman for review comments.

Additional thanks to Andy Malis and Danny McPherson for advice about the use of the Control Word.

10. References

10.1. Normative References


10.2. Informative References

[I-D.ietf-mpls-pm-udp-return]


[RFC4086]  Eastlake, D., Schiller, J., and S. Crocker, "Randomness

2005.

[RFC4303]  Kent, S., "IP Encapsulating Security Payload (ESP)",

Label Switched (MPLS) Data Plane Failures" RFC 4379,
February 2006.

[RFC4990]  Shiomoto, K., Papneja, R., and R. Rabbat, "Use of
Addresses in Generalized Multiprotocol Label Switching

[RFC5036]  Andersson, L., Minei, I., and B. Thomas, "LDP

(SSH)", RFC 6239, May 2011.

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

Is an Attack", BCP 188, RFC 7258, May 2014.

T. Kivinen, "Internet Key Exchange Protocol Version 2
(IKEv2)", STD 79, RFC 7296, October 2014.

[RFC7325]  C. Villamizar, Ed., "MPLS Forwarding Compliance and
Performance Requirements", RFC 7325, August 2014.

of the Time", RFC 7435, December 2014.

[RFC7510]  X. Xu et al., "Encapsulating MPLS in UDP", RFC 7510, April
2015.

[RFC7551]  Zhang, F., Jing, R., and R. Gandhi, "RSVP-TE Extensions
for Associated Bidirectional Label Switched Paths (LSPs)",
Authors' Addresses

Adrian Farrel
Juniper Networks
EMail: adrian@olddog.co.uk

Stephen Farrell
Trinity College Dublin
Dublin, 2
Ireland
Phone: +353-1-896-2354
Email: stephen.farrell@cs.tcd.ie
Abstract

This specification describes the configuration of proactive MPLS-TP Operations, Administration, and Maintenance (OAM) Functions for a given Label Switched Path (LSP) using a set of TLVs that are carried by the LSP-Ping protocol.

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 26, 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 ................................................. 3
   1.1. Conventions used in this document ..................... 4
       1.1.1. Terminology ..................................... 4
       1.1.2. Requirements Language ............................ 5
2. Theory of Operations ........................................... 5
   2.1. MPLS OAM Configuration Operation Overview ................ 5
       2.1.1. Configuration of BFD Sessions ..................... 5
       2.1.2. Configuration of Performance Monitoring ............. 6
       2.1.3. Configuration of Fault Management Signals .......... 6
   2.2. MPLS OAM Functions TLV .................................. 7
       2.2.1. BFD Configuration Sub-TLV ........................ 9
       2.2.2. Local Discriminator Sub-TLV ...................... 11
       2.2.3. Negotiation Timer Parameters Sub-TLV ............... 11
       2.2.4. BFD Authentication Sub-TLV ....................... 12
       2.2.5. Traffic Class Sub-TLV ............................. 13
       2.2.6. Performance Measurement Sub-TLV ................... 14
       2.2.7. PM Loss Measurement Sub-TLV ...................... 16
       2.2.8. PM Delay Measurement Sub-TLV ...................... 17
       2.2.9. Fault Management Signal Sub-TLV .................... 18
       2.2.10. Source MEP-ID Sub-TLV .......................... 20
3. Summary of MPLS OAM Configuration Errors ..................... 20
4. IANA Considerations .......................................... 22
   4.1. TLV and Sub-TLV Allocation ............................ 22
   4.2. MPLS OAM Function Flags Allocation .................... 23
   4.3. OAM Configuration Errors .............................. 23
5. Security Considerations ...................................... 24
6. Acknowledgements ............................................. 25
7. References .................................................. 25
   7.1. Normative References ................................. 25
   7.2. Informative References ................................ 26
Authors’ Addresses .............................................. 27
1. Introduction

The MPLS Transport Profile (MPLS-TP) describes a profile of MPLS that enables operational models typical in transport networks, while providing additional Operations, Administration, and Maintenance (OAM), survivability and other maintenance functions not currently supported by MPLS. [RFC5860] defines the requirements for the OAM functionality of MPLS-TP.

This document describes the configuration of proactive MPLS-TP OAM Functions for a given Label Switched Path (LSP) using TLVs carried in LSP Ping [RFC4379]. In particular it specifies the mechanisms necessary to establish MPLS-TP OAM entities at the maintenance points for monitoring and performing measurements on an LSP, as well as defining information elements and procedures to configure proactive MPLS-TP OAM functions running between LERs. Initialization and control of on-demand MPLS-TP OAM functions are expected to be carried out by directly accessing network nodes via a management interface; hence configuration and control of on-demand OAM functions are out-of-scope for this document.

The Transport Profile of MPLS must, by definition [RFC5654], be capable of operating without a control plane. Therefore there are several options for configuring MPLS-TP OAM, without a control plane by either using an NMS or LSP Ping, or with a control plane using signaling protocols RSVP Traffic engineering (RSVP-TE) [RFC3209] and/or Targeted LDP [RFC5036].

Proactive MPLS-TP OAM is performed by set of protocols, Bi-directional Forwarding Detection (BFD) [RFC6428] for Continuity Check/Connectivity Verification, the delay measurement protocol (DM) [RFC6374], [RFC6375] for delay and delay variation (jitter) measurements, and the loss measurement (LM) protocol [RFC6374], [RFC6375] for packet loss and throughput measurements. Additionally, there is a number of Fault Management Signals that can be configured [RFC6427].

BFD is a protocol that provides low-overhead, fast detection of failures in the path between two forwarding engines, including the interfaces, data link(s), and, to the extent possible, the forwarding engines themselves. BFD can be used to detect the continuity and mis-connection defects of MPLS-TP point-to-point and might also be extended to support point-to-multipoint label switched paths (LSPs).

The delay and loss measurements protocols [RFC6374] and [RFC6375] use a simple query/response model for performing both uni- and bi-directional measurements that allow the originating node to measure packet loss and delay in forward or forward and reverse directions.
By timestamping and/or writing current packet counters to the measurement packets (four times, Transmit and Receive in both directions), current delays and packet losses can be calculated. By performing successive delay measurements, the delay and/or inter-packet delay variation (jitter) can be calculated. Current throughput can be calculated from the packet loss measurements by dividing the number of packets sent/received with the time it took to perform the measurement, given by the timestamp in LM header. Combined with a packet generator the throughput measurement can be used to measure the maximum capacity of a particular LSP. It should be noted that this document does not specify how to configure on-demand throughput estimates based on saturating the connection as defined in [RFC6371]. Rather, only how to enable the estimation of the current throughput based on loss measurements.

1.1. Conventions used in this document

1.1.1. Terminology

BFD - Bidirectional Forwarding Detection
DM - Delay Measurement
FMS - Fault Management Signal
G-ACh - Generic Associated Channel
LSP - Label Switched Path
LM - Loss Measurement
MEP - Maintenance Entity Group End Point
MPLS - Multi-Protocol Label Switching
MPLS-TP - MPLS Transport Profile
NMS - Network management System
PM - Performance Measurement
RSVP-TE - RSVP Traffic Engineering
TC - Traffic Class
1.1.2. 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 RFC 2119 [RFC2119].

2. Theory of Operations

2.1. MPLS OAM Configuration Operation Overview

The MPLS-TP OAM tool set is described in the [RFC6669].

LSP Ping, or alternatively RSVP-TE [RFC7487], can be used to simply enable the different OAM functions, by setting the corresponding flags in the MPLS OAM Functions TLV (refer to Section 2.2). For a more detailed configuration, one may include sub-TLVs for the different OAM functions in order to specify various parameters in detail.

Typically intermediate nodes simply forward OAM configuration TLVs to the end-node without any processing or modification. At least one exception to this is if the FMS sub-TLV (refer to Section 2.2.9 ) is present. This sub-TLV MUST be examined even by intermediate nodes that support this extension. The sub-TLV MAY be present if a flag is set in the MPLS OAM Functions TLV.

2.1.1. Configuration of BFD Sessions

For this specification, BFD MUST run in either one of the two modes:

- Asynchronous mode, where both sides are in active mode
- Unidirectional mode

In the simplest scenario, LSP Ping [RFC5884], or alternatively RSVP-TE [RFC7487], is used only to bootstrap a BFD session for an LSP, without any timer negotiation.

Timer negotiation can be performed either in subsequent BFD control messages (in this case the operation is similar to LSP Ping based bootstrapping described in [RFC5884]) or directly in the LSP-Ping configuration messages.

When BFD Control packets are transported in the ACH encapsulation, they are not protected by any end-to-end checksum, only lower-layers are providing error detection/correction. A single bit error, e.g. a flipped bit in the BFD State field could cause the receiving end to wrongly conclude that the link is down and in turn trigger protection
switching. To prevent this from happening, the BFD Configuration sub-TLV (refer to Section 2.2.1) has an Integrity flag that when set enables BFD Authentication using Keyed SHA1 with an empty key (all 0s) [RFC5880]. This would make every BFD Control packet carry an SHA1 hash of itself that can be used to detect errors.

If BFD Authentication using a pre-shared key/password is desired (i.e. authentication and not only error detection), the BFD Authentication sub-TLV (refer to Section 2.2.4) MUST be included in the BFD Configuration sub-TLV. The BFD Authentication sub-TLV is used to specify which authentication method that should be used and which pre-shared key/ password that should be used for this particular session. How the key exchange is performed is out of scope of this document.

2.1.2. Configuration of Performance Monitoring

It is possible to configure Performance Monitoring functionalities such as Loss, Delay, Delay/Interpacket Delay variation (jitter), and Throughput as described in [RFC6374].

When configuring Performance Monitoring functionalities, it is possible to choose either the default configuration, by only setting the respective flags in the MPLS OAM functions TLV, or a customized configuration. To customize the configuration, one would set the respective flags in the MPLS OAM functions TLV and include the respective Loss and/or Delay sub-TLVs.

By setting the PM Loss flag in the MPLS OAM Functions TLV and including the PM Loss sub-TLV (refer to Section 2.2.7) one can configure the measurement interval and loss threshold values for triggering protection.

Delay measurements are configured by setting the PM Delay flag in the MPLS OAM Functions TLV and including the PM Delay sub-TLV (refer to Section 2.2.8) one can configure the measurement interval and the delay threshold values for triggering protection.

2.1.3. Configuration of Fault Management Signals

To configure Fault Management Signals (FMS) and their refresh time, the FMS flag in the MPLS OAM Functions TLV MUST be set and the FMS sub-TLV MUST be included. When configuring FMS, an implementation can enable the default configuration by setting the FMS flag in the OAM Function Flags sub-TLV. In order to modify the default configuration, the MPLS OAM FMS sub-TLV MUST be included.
If an intermediate point is meant to originate fault management signal messages, this means that such an intermediate point is associated with a Server MEP through a co-located MPLS-TP client/server adaptation function, and the Fault Management subscription flag in the MPLS OAM FMS sub-TLV has been set as indication of the request to create the association at each intermediate node of the client LSP. The corresponding Server MEP needs to be configured by its own LSP-ping session or, alternatively, via a Network Management system (NMS) or RSVP-TE.

2.2. MPLS OAM Functions TLV

The MPLS OAM Functions TLV presented in Figure 1 is carried as a TLV of the MPLS Echo Request/Reply messages [RFC4379].

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  MPLS OAM Func. Type (TBA1)   |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    MPLS OAM Function Flags                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                                 |
|                                                               |
|     sub-TLVs                                                |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: MPLS OAM Functions TLV format

The MPLS OAM Functions TLV contains MPLS OAM Function Flags field. The MPLS OAM Function Flags indicates which OAM functions should be activated as well as OAM function specific sub-TLVs with configuration parameters for the particular function.

Type: indicates the MPLS OAM Functions TLV Section 4.

Length: the length of the MPLS OAM Function Flags field including the total length of the sub-TLVs in octets.

MPLS OAM Function Flags: a bitmap numbered from left to right as shown in the Figure 2. These flags are managed by IANA (refer to Section 4.2). Flags defined in this document are presented in Table 2. Undefined flags MUST be set to zero and unknown flags MUST be ignored. The flags indicate what OAM is being configured and direct the presence of optional sub-TLVs as set out below.
Sub-TLVs corresponding to the different flags are as follows. No meaning should be attached to the order of sub-TLVs.

- If a flag in the MPLS OAM Function Flags is set and the corresponding sub-TLVs listed below is absent, then this MPLS OAM function MUST be initialized according to its default settings. Default settings of MPLS OAM functions are outside the scope of this document.

- If any sub-TLV is present without the corresponding flag being set, the sub-TLV SHOULD be ignored.

- BFD Configuration sub-TLV, which MUST be included if either the CC, the CV or both MPLS OAM Function flags being set in the MPLS OAM Functions TLV.

- Performance Monitoring sub-TLV MUST be used to carry PM Loss sub-TLV and/or PM Delay sub-TLV. If neither one of these sub-TLVs is present then Performance Monitoring sub-TLV SHOULD NOT be included. Empty, i.e. no enclosed sub-TLVs, Performance Monitoring sub-TLV SHOULD be ignored.

- PM Loss sub-TLV MAY be included if the PM/Loss OAM Function flag is set. If the "PM Loss sub-TLV" is not included, default configuration values are used. Such sub-TLV MAY also be included in case the Throughput function flag is set and there is the need to specify a measurement interval different from the default ones. In fact, the throughput measurement makes use of the same tool as the loss measurement, hence the same TLV is used.

- PM Delay sub-TLV MAY be included if the PM/Delay OAM Function flag is set. If the "PM Delay sub-TLV" is not included, default configuration values are used.

- FMS sub-TLV, which MAY be included if the FMS OAM Function flag is set. If the "FMS sub-TLV" is not included, default configuration values are used.

If all flags in the MPLS OAM Function Flags field have the same value of zero, that MUST be interpreted as the MPLS OAM Functions TLV not
present in the MPLS Echo Request. If more than one MPLS OAM Functions TLV is present in the MPLS Echo request packet, then the first TLV SHOULD be processed and the rest be ignored. Any parsing error within nested sub-TLVs that is not specified in Section 3 SHOULD be treated as described in [RFC4379].

2.2.1. BFD Configuration Sub-TLV

The BFD Configuration sub-TLV, depicted in Figure 3, is defined for BFD OAM specific configuration parameters. The "BFD Configuration sub-TLV" is carried as a sub-TLV of the "OAM Functions TLV".

This TLV accommodates generic BFD OAM information and carries sub-TLVs.

![Figure 3: BFD Configuration sub-TLV format](image)

Sub-type: indicates a new sub-type, the BFD Configuration sub-TLV (value 100).

Length: indicates the length of the Value field in octets.

Version: identifies the BFD protocol version. If a node does not support a specific BFD version an error must be generated: "OAM Problem/Unsupported OAM Version".

BFD Negotiation (N): If set timer negotiation/re-negotiation via BFD Control Messages is enabled, when cleared it is disabled and timer configuration is achieved using Negotiation Timer Parameters sub-TLV as described in Section 2.2.3.

Symmetric session (S): If set the BFD session MUST use symmetric timing values. If cleared the BFD session MAY use any timing values either negotiated or explicitly configured.
Integrity (I): If set BFD Authentication MUST be enabled. If the BFD Configuration sub-TLV does not include a BFD Authentication sub-TLV the authentication MUST use Keyed SHA1 with an empty pre-shared key (all 0s). If the egress LSR does not support BFD Authentication an error MUST be generated: "OAM Problem/BFD Authentication unsupported". If the Integrity flag is clear, then Authentication MUST NOT be used.

Encapsulation Capability (G): if set, it shows the capability of encapsulating BFD messages into G-ACh channel. If both the G bit and U bit are set, configuration gives precedence to the G bit.

Encapsulation Capability (U): if set, it shows the capability of encapsulating BFD messages into IP/UDP packets. If both the G bit and U bit are set, configuration gives precedence to the G bit.

If the egress LSR does not support any of the ingress LSR Encapsulation Capabilities an error MUST be generated: "OAM Problem/Unsupported BFD Encapsulation format".

Bidirectional (B): if set, it configures BFD in the Bidirectional mode. If it is not set it configures BFD in unidirectional mode. In the second case, the source node does not expect any Discriminator values back from the destination node.

Reserved: Reserved for future specification and set to 0 on transmission and ignored when received.

The BFD Configuration sub-TLV MUST include the following sub-TLVs in the MPLS Echo Request message:

- Local Discriminator sub-TLV, if B flag is set in the MPLS Echo Request;
- Negotiation Timer Parameters sub-TLV if the N flag is cleared.

The BFD Configuration sub-TLV MUST include the following sub-TLVs in the MPLS Echo Reply message:

- Local Discriminator sub-TLV;
- Negotiation Timer Parameters sub-TLV if:
  - the N and S flags are cleared, or if:
  - the N flag is cleared and the S flag is set, and the Negotiation Timer Parameters sub-TLV received by the egress contains unsupported values. In this case an updated
Negotiation Timer Parameters sub-TLV, containing values supported by the egress node [RFC7419], is returned to the ingress.

2.2.2. Local Discriminator Sub-TLV

The Local Discriminator sub-TLV is carried as a sub-TLV of the "BFD Configuration sub-TLV" and is depicted in Figure 4.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Locl. Discr. sub-Type (101) | Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Local Discriminator                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Local Discriminator sub-TLV format

Type: indicates a new type, the "Local Discriminator sub-TLV" (value 101).

Length: indicates the length of the Value field in octets. (4)

Local Discriminator: A nonzero discriminator value that is unique in the context of the transmitting system that generates it. It is used to demultiplex multiple BFD sessions between the same pair of systems.

2.2.3. Negotiation Timer Parameters Sub-TLV

The Negotiation Timer Parameters sub-TLV is carried as a sub-TLV of the BFD Configuration sub-TLV and is depicted in Figure 5.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Nego. Timer sub-type (102) | Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Acceptable Min. Asynchronous TX interval |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Acceptable Min. Asynchronous RX interval |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Required Echo TX Interval |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Negotiation Timer Parameters sub-TLV format
Sub-type: indicates a new sub-type, the Negotiation Timer Parameters sub-TLV (value 102).

Length: indicates the length of the Value field in octets (12).

Acceptable Min. Asynchronous TX interval: in case of S (symmetric) flag set in the BFD Configuration sub-TLV, defined in Section 2.2.1, it expresses the desired time interval (in microseconds) at which the ingress LER intends to both transmit and receive BFD periodic control packets. If the receiving edge LSR cannot support such value, it SHOULD reply with an interval greater than the one proposed.

In case of S (symmetric) flag cleared in the BFD Configuration sub-TLV, this field expresses the desired time interval (in microseconds) at which a edge LSR intends to transmit BFD periodic control packets in its transmitting direction.

Acceptable Min. Asynchronous RX interval: in case of S (symmetric) flag set in the BFD Configuration sub-TLV, Figure 3, this field MUST be equal to Acceptable Min. Asynchronous TX interval and has no additional meaning respect to the one described for "Acceptable Min. Asynchronous TX interval".

In case of S (symmetric) flag cleared in the BFD Configuration sub-TLV, it expresses the minimum time interval (in microseconds) at which edge LSRs can receive BFD periodic control packets. In case this value is greater than the value of Acceptable Min. Asynchronous TX interval received from the other edge LSR, such edge LSR MUST adopt the interval expressed in this Acceptable Min. Asynchronous RX interval.

Required Echo TX Interval: the minimum interval (in microseconds) between received BFD Echo packets that this system is capable of supporting, less any jitter applied by the sender as described in [RFC5880] sect. 6.8.9. This value is also an indication for the receiving system of the minimum interval between transmitted BFD Echo packets. If this value is zero, the transmitting system does not support the receipt of BFD Echo packets. If the receiving system cannot support this value the "Unsupported BFD TX Echo rate interval" error MUST be generated. By default the value is set to 0.

2.2.4. BFD Authentication Sub-TLV

The "BFD Authentication sub-TLV" is carried as a sub-TLV of the "BFD Configuration sub-TLV" and is depicted in Figure 6.
Sub-type: indicates a new type, the BFD Authentication sub-TLV (value 103).

Length: indicates the length of the Value field in octets (4).

Auth Type: indicates which type of authentication to use. The same values as are defined in section 4.1 of [RFC5880] are used. Simple Password SHOULD NOT be used if other authentication types are available.

Auth Key ID: indicates which authentication key or password (depending on Auth Type) should be used. How the key exchange is performed is out of scope of this document. If the egress LSR does not support this Auth Key ID an "OAM Problem/Mismatch of BFD Authentication Key ID" error MUST be generated.

Reserved: Reserved for future specification and set to 0 on transmission and ignored when received.

An implementation MAY change mode of authentication if an operator re-evaluates the security situation in and around the administrative domain. If BFD Authentication sub-TLV used for a BFD session in Up state, then the Sender of the MPLS LSP Echo Request SHOULD ensure that old and new modes of authentication, i.e., combination of Auth.Type and Auth. Key ID, are used to send and receive BFD control packets, until the Sender can confirm that its peer has switched to the new authentication.

2.2.5. Traffic Class Sub-TLV

The Traffic Class sub-TLV is carried as a sub-TLV of the "BFD Configuration sub-TLV" and "Fault Management Signal sub-TLV" Section 2.2.9 and is depicted in Figure 7.
Figure 7: Traffic Class sub-TLV format

Type: indicates a new type, the "Traffic Class sub-TLV" (value 104).

Length: indicates the length of the Value field in octets . (4)

TC: Identifies the Traffic Class (TC) [RFC5462] for periodic continuity monitoring messages or packets with fault management information.

If the TC sub-TLV is present, then the sender of any periodic continuity monitoring messages or packets with fault management information on the LSP, with a FEC that corresponds to the FEC for which fault detection is being performed, MUST use the value contained in the TC field of the sub-TLV as the value of the TC field in the top label stack entry of the MPLS label stack. If the TC sub-TLV is absent from either "BFD Configuration sub-TLV" or "Fault Management Signal sub-TLV", then selection of the TC value is local decision.

2.2.6. Performance Measurement Sub-TLV

If the MPLS OAM Functions TLV has any of the L (Loss), D (Delay) and T (Throughput) flag set, the Performance Measurement sub-TLV MUST be present. Failure to include the correct sub-TLVs MUST result in an "OAM Problem/ Configuration Error" error being generated.

The Performance Measurement sub-TLV provides the configuration information mentioned in Section 7 of [RFC6374]. It includes support for the configuration of quality thresholds and, as described in [RFC6374], "the crossing of which will trigger warnings or alarms, and result in reporting and exception notification will be integrated into the system-wide network management and reporting framework."

In case the values need to be different than the default ones, the Performance Measurement sub-TLV MAY include the following sub-TLVs:

- PM Loss sub-TLV if the L flag is set in the MPLS OAM Functions TLV;
- PM Delay sub-TLV if the D flag is set in the MPLS OAM Functions TLV.

The Performance Measurement sub-TLV depicted in Figure 8 is carried as a sub-TLV of the MPLS OAM Functions TLV.

```
+-------------------------------+-------------------------------+
| Perf Monitoring Type (200)    |          Length               |
|-------------------------------+-------------------------------|
+-------------------------------+-------------------------------|
| PM Configuration Flags        | sub-TLVs                      |
|-------------------------------+-------------------------------|
+-------------------------------+-------------------------------+
```

Figure 8: Performance Measurement sub-TLV format

Sub-type: indicates a new sub-type, the Performance Management sub-TLV" (value 200).

Length: indicates the length of the Value field in octets, including PM Configuration Flags and optional sub-TLVs.

```
+-------------------------------+-------------------------------+
| D|L|J|Y|K|C|            Reserved (set to all 0s)               |
|-------------------------------+-------------------------------|
+-------------------------------+-------------------------------+
```

Figure 9: Performance Measurement sub-TLV format

PM Configuration Flags, format is presented in Figure 9, for the specific function description please refer to [RFC6374]:

- D: Delay inferred/direct (0=INFERRED, 1=DIRECT). If the egress LSR does not support specified mode an "OAM Problem/Unsupported Delay Mode" error MUST be generated.

- L: Loss inferred/direct (0=INFERRED, 1=DIRECT). If the egress LSR does not support specified mode an "OAM Problem/Unsupported Loss Mode" error MUST be generated.

- J: Delay variation/jitter (1=ACTIVE, 0=NOT ACTIVE). If the egress LSR does not support Delay variation measurements and the J
flag is set, an "OAM Problem/Delay variation unsupported" error MUST be generated.

- Y: Dyadic (1=ACTIVE, 0=NOT ACTIVE). If the egress LSR does not support Dyadic mode and the Y flag is set, an "OAM Problem/Dyadic mode unsupported" error MUST be generated.

- K: Loopback (1=ACTIVE, 0=NOT ACTIVE). If the egress LSR does not support Loopback mode and the K flag is set, an "OAM Problem/Loopback mode unsupported" error MUST be generated.

- C: Combined (1=ACTIVE, 0=NOT ACTIVE). If the egress LSR does not support Combined mode and the C flag is set, an "OAM Problem/Combined mode unsupported" error MUST be generated.

Reserved: Reserved for future specification and set to 0 on transmission and ignored when received.

2.2.7. PM Loss Measurement Sub-TLV

The PM Loss Measurement sub-TLV depicted in Figure 10 is carried as a sub-TLV of the Performance Measurement sub-TLV.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 10: PM Loss Measurement sub-TLV format

Sub-type: indicates a new sub-type, the PM Loss Measurement sub-TLV (value 201).

Length: indicates the length of the Value field in octets (16).

OTF: Origin Timestamp Format of the Origin Timestamp field described in [RFC6374]. By default it is set to IEEE 1588 version 1. If the egress LSR cannot support this value an "OAM Problem/Unsupported Timestamp Format" error MUST be generated.
Configuration Flags, please refer to [RFC6374] for further details:

- T: Traffic-class-specific measurement indicator. Set to 1 when the measurement operation is scoped to packets of a particular traffic class (DSCP value), and 0 otherwise. When set to 1, the DS field of the message indicates the measured traffic class. By default it is set to 1.

- B: Octet (byte) count. When set to 1, indicates that the Counter 1-4 fields represent octet counts. When set to 0, indicates that the Counter 1-4 fields represent packet counts. By default it is set to 0.

Reserved: Reserved for future specification and set to 0 on transmission and ignored when received.

Measurement Interval: the time interval (in milliseconds) at which Loss Measurement query messages MUST be sent on both directions. If the edge LSR receiving the Path message cannot support such value, it SHOULD reply with a higher interval. By default it is set to (100) as per [RFC6375].

Test Interval: test messages interval in milliseconds as described in [RFC6374]. By default it is set to (10) as per [RFC6375].

Loss Threshold: the threshold value of measured lost packets per measurement over which action(s) SHOULD be triggered.

### 2.2.8. PM Delay Measurement Sub-TLV

The "PM Delay Measurement sub-TLV" depicted in Figure 11 is carried as a sub-TLV of the Performance Monitoring sub-TLV.

```plaintext
type 202
+----------------------------------+-
| OTF |T|B| Reserved (set to all 0s) |
+----------------------------------+-
| Measurement Interval |
+----------------------+-
| Test Interval |
+-------------+-
| Delay Threshold |
```

Figure 11: PM Delay Measurement sub-TLV format
Sub-type: indicates a new sub-type, the "PM Delay Measurement sub-TLV" (value 202).

Length: indicates the length of the Value field in octets (16).

OTF: Origin Timestamp Format of the Origin Timestamp field described in [RFC6374]. By default it is set to IEEE 1588 version 1. If the egress LSR cannot support this value, an "OAM Problem/Unsupported Timestamp Format" error MUST be generated.

Configuration Flags, please refer to [RFC6374] for further details:

- T: Traffic-class-specific measurement indicator. Set to 1 when the measurement operation is scoped to packets of a particular traffic class (DSCP value), and 0 otherwise. When set to 1, the DS field of the message indicates the measured traffic class. By default it is set to 1.

- B: Octet (byte) count. When set to 1, indicates that the Counter 1-4 fields represent octet counts. When set to 0, indicates that the Counter 1-4 fields represent packet counts. By default it is set to 0.

Reserved: Reserved for future specification and set to 0 on transmission and ignored when received.

Measurement Interval: the time interval (in milliseconds) at which Delay Measurement query messages MUST be sent on both directions. If the edge LSR receiving the Path message cannot support such value, it can reply with a higher interval. By default it is set to (1000) as per [RFC6375].

Test Interval: test messages interval (in milliseconds) as described in [RFC6374]. By default it is set to (10) as per [RFC6375].

Delay Threshold: the threshold value of measured two-way delay (in milliseconds) over which action(s) SHOULD be triggered.

2.2.9. Fault Management Signal Sub-TLV

The FMS sub-TLV depicted in Figure 12 is carried as a sub-TLV of the MPLS OAM Configuration sub-TLV. When both working and protection paths are configured, both LSPs SHOULD be configured with identical settings of the E flag, T flag, and the refresh timer. An implementation MAY configure the working and protection LSPs with different settings of these fields in case of 1:N protection.
Sub-type: indicates a new sub-type, the FMS sub-TLV (value 300).

Length: indicates the length of the Value field in octets.

FMS Signal Flags are used to enable the FMS signals at end point MEPs and the Server MEPs of the links over which the LSP is forwarded. In this document only the S flag pertains to Server MEPs.

The following flags are defined:

- E: Enable Alarm Indication Signal (AIS) and Lock Report (LKR) signaling as described in [RFC6427]. Default value is 1 (enabled). If the egress MEP does not support FMS signal generation, an "OAM Problem/Fault management signaling unsupported" error MUST be generated.

- S: Indicate to a server MEP that it should transmit AIS and LKR signals on the client LSP. Default value is 0 (disabled). If a Server MEP which is capable of generating FMS messages is for some reason unable to do so for the LSP being signaled, an "OAM Problem/Unable to create fault management association" error MUST be generated.

- T: Set timer value, enabled the configuration of a specific timer value. Default value is 0 (disabled).

- Remaining bits: Reserved for future specification and set to 0.

Refresh Timer: indicates the refresh timer of fault indication messages, in seconds. The value MUST be between 1 to 20 seconds as specified for the Refresh Timer field in [RFC6427]. If the edge LSR receiving the Path message cannot support the value it SHOULD reply with a higher timer value.
FMS sub-TLV MAY include Traffic Class sub-TLV Section 2.2.5. If TC sub-TLV is present, the value of the TC field MUST be used as the value of the TC field of an MPLS label stack entry for FMS messages. If the TC sub-TLV is absent, then selection of the TC value is local decision.

2.2.10. Source MEP-ID Sub-TLV

The Source MEP-ID sub-TLV depicted in Figure 13 is carried as a sub-TLV of the MPLS OAM Functions TLV.

Note that support of ITU IDs is out-of-scope.

<table>
<thead>
<tr>
<th>Source MEP-ID sub-type (400)</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Node ID</td>
<td></td>
</tr>
<tr>
<td>Tunnel ID</td>
<td>LSP ID</td>
</tr>
</tbody>
</table>

Figure 13: Source MEP-ID sub-TLV format

Sub-type: indicates a new sub-type, the Source MEP-ID sub-TLV (value 400).

Length: indicates the length of the Value field in octets (8).

Source Node ID: 32-bit node identifier as defined in [RFC6370].

Tunnel ID: a 16-bit unsigned integer unique to the node as defined in [RFC6370].

LSP ID: a 16-bit unsigned integer unique within the Tunnel_ID as defined in [RFC6370].

3. Summary of MPLS OAM Configuration Errors

This is the summary of Return Codes [RFC4379] defined in this document:

- If an egress LSR does not support the specified BFD version, an error MUST be generated: "OAM Problem/Unsupported BFD Version".

- If an egress LSR does not support the specified BFD Encapsulation format, an error MUST be generated: "OAM Problem/Unsupported BFD Encapsulation format".

- If an egress LSR does not support BFD Authentication, and it is requested, an error MUST be generated: "OAM Problem/BFD Authentication unsupported".

- If an egress LSR does not support the specified BFD Authentication Type, an error MUST be generated: "OAM Problem/Unsupported BFD Authentication Type".

- If an egress LSR is not able to use the specified Authentication Key ID, an error MUST be generated: "OAM Problem/Mismatch of BFD Authentication Key ID".

- If an egress LSR does not support the specified Timestamp Format, an error MUST be generated: "OAM Problem/Unsupported Timestamp Format".

- If an egress LSR does not support specified Delay mode, an "OAM Problem/Unsupported Delay Mode" error MUST be generated.

- If an egress LSR does not support specified Loss mode, an "OAM Problem/Unsupported Loss Mode" error MUST be generated.

- If an egress LSR does not support Delay variation measurements, and it is requested, an "OAM Problem/Delay variation unsupported" error MUST be generated.

- If an egress LSR does not support Dyadic mode, and it is requested, an "OAM Problem/Dyadic mode unsupported" error MUST be generated.

- If an egress LSR does not support Loopback mode, and it is requested, an "OAM Problem/Loopback mode unsupported" error MUST be generated.

- If an egress LSR does not support Combined mode, and it is requested, an "OAM Problem/Combined mode unsupported" error MUST be generated.

- If an egress LSR does not support Fault Monitoring Signals, and it is requested, an "OAM Problem/Fault management signaling unsupported" error MUST be generated.

- If an intermediate server MEP supports Fault Monitoring Signals but is unable to create an association, when requested to do so,
an "OAM Problem/Unable to create fault management association" error MUST be generated.

Ingress LSR MAY combine multiple MPLS OAM configuration TLVs and sub-TLVs into single MPLS echo request. In case an egress LSR doesn’t support any of the requested modes it MUST set the return code to report the first unsupported mode in the list of TLVs and sub-TLVs. And if any of the requested OAM configuration is not supported the egress LSR SHOULD NOT process OAM Configuration TLVs and sub-TLVs listed in the MPLS echo request.

4. IANA Considerations

4.1. TLV and Sub-TLV Allocation

IANA maintains the Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters registry, and within that registry a sub-registry for TLVs and sub-TLVs.

IANA is requested to allocate a new MPLS OAM Functions TLV from the standards action range (0-16383) and sub-TLVs as follows from sub-registry presented in Table 1, called "Sub-TLVs for TLV [TBA1]".

Registration procedures for Sub-TLVs from ranges 0-16383 and 32768-49161 are by Standards Action, and from ranges 16384-31743 and 49162-64511 are through Specification Required (Experimental RFC Needed).

<table>
<thead>
<tr>
<th>Type</th>
<th>Sub-type</th>
<th>Value Field</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MPLS OAM Functions</td>
<td>This document</td>
</tr>
<tr>
<td>TBA1</td>
<td>100</td>
<td>BFD Configuration</td>
<td>This document</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>BFD Local Discriminator</td>
<td>This document</td>
</tr>
<tr>
<td></td>
<td>102</td>
<td>BFD Negotiation Timer</td>
<td>This document</td>
</tr>
<tr>
<td></td>
<td>103</td>
<td>BFD Authentication</td>
<td>This document</td>
</tr>
<tr>
<td></td>
<td>104</td>
<td>Traffic Class</td>
<td>This document</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>Performance Measurement</td>
<td>This document</td>
</tr>
<tr>
<td></td>
<td>201</td>
<td>PM Loss Measurement</td>
<td>This document</td>
</tr>
<tr>
<td></td>
<td>202</td>
<td>PM Delay Measurement</td>
<td>This document</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>Fault Management Signal</td>
<td>This document</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>Source MEP-ID</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 1: IANA TLV Type Allocation
4.2. MPLS OAM Function Flags Allocation

IANA is requested to create a new registry called the "MPLS OAM Function Flags" registry. Assignments of bit positions 0 through 31 are via Standards Action. The new registry to be populated as follows.

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>MPLS OAM Function Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>Continuity Check (CC)</td>
</tr>
<tr>
<td>1</td>
<td>V</td>
<td>Connectivity Verification (CV)</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>Fault Management Signal (FMS)</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>Performance Measurement/Loss (PM/Loss)</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>Performance Measurement/Delay (PM/Delay)</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>Throughput Measurement</td>
</tr>
<tr>
<td>6-30</td>
<td></td>
<td>Unassigned (Must be zero)</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 2: MPLS OAM Function Flags

4.3. OAM Configuration Errors

IANA maintains a registry "Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters" registry, and within that registry a sub-registry "Return Codes".

IANA is requested to assign new Return Codes from the Standards Action range (0-191) as follows:
### Error Value Sub-codes

<table>
<thead>
<tr>
<th>Error Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA3</td>
<td>OAM Problem/Unsupported BFD Version</td>
<td>This document</td>
</tr>
<tr>
<td>TBA4</td>
<td>OAM Problem/Unsupported BFD Version</td>
<td>This document</td>
</tr>
<tr>
<td>TBA5</td>
<td>OAM Problem/Unsupported BFD Version</td>
<td>This document</td>
</tr>
<tr>
<td>TBA6</td>
<td>Authentication Type</td>
<td>This document</td>
</tr>
<tr>
<td>TBA7</td>
<td>OAM Problem/Unsupported Timestamp</td>
<td>This document</td>
</tr>
<tr>
<td>TBA8</td>
<td>OAM Problem/Unsupported Delay Mode</td>
<td>This document</td>
</tr>
<tr>
<td>TBA9</td>
<td>OAM Problem/Unsupported Loss Mode</td>
<td>This document</td>
</tr>
<tr>
<td>TBA10</td>
<td>OAM Problem/Delay variation unsupported</td>
<td>This document</td>
</tr>
<tr>
<td>TBA11</td>
<td>OAM Problem/Dyadic mode unsupported</td>
<td>This document</td>
</tr>
<tr>
<td>TBA12</td>
<td>OAM Problem/Loopback mode unsupported</td>
<td>This document</td>
</tr>
<tr>
<td>TBA13</td>
<td>OAM Problem/Combined mode unsupported</td>
<td>This document</td>
</tr>
<tr>
<td>TBA14</td>
<td>OAM Problem/Fault management signaling unsupported</td>
<td>This document</td>
</tr>
<tr>
<td>TBA15</td>
<td>OAM Problem/Unable to create fault management association</td>
<td>This document</td>
</tr>
</tbody>
</table>

**Table 3: IANA Return Codes Allocation**

5. Security Considerations

The signaling of OAM related parameters and the automatic establishment of OAM entities introduces additional security considerations to those discussed in [RFC4379]. In particular, a network element could be overloaded if an attacker were to request high frequency liveliness monitoring of a large number of LSPs, targeting a single network element. Implementations must be made cognizant of available OAM resources and MAY refuse new OAM configurations that would overload a node. Additionally, policies to manage OAM resources may be used to provide some fairness in OAM resource distribution among monitored LSPs.
Security of OAM protocols configured with extensions to LSP Ping described in this document are discussed in [RFC5880], [RFC5884], [RFC6374], [RFC6427], and [RFC6428].

In order that the configuration of OAM functionality can be achieved securely through the techniques described in this document, security mechanisms must already be in place and operational for LSP Ping. Thus the exchange of security parameters (such as keys) for use in securing OAM is outside the scope of this document and is assumed to use an off-line mechanism or an established secure key-exchange protocol.

Additional discussion of security for MPLS protocols can be found in [RFC5920].

6. Acknowledgements

The authors would like to thank Nobo Akiya, David Allan and Adrian Farrel for their thorough reviews and insightful comments.

7. References

7.1. Normative References


7.2. Informative References


Authors’ Addresses

Elisa Bellagamba
Email: elisa.bellagamba@gmail.com

Gregory Mirsky
Ericsson
Email: Gregory.Mirsky@ericsson.com

Loa Andersson
Huawei Technologies
Email: loa@mail01.huawei.com
LSP Self-Ping
draft-ietf-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 document 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 a new protocol. It is not an extension of LSP Ping. Although LSP Ping and LSP Self-ping are named similarly, each is designed for a unique purpose. Each protocol listens on its own UDP port and executes its own procedures.

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 May 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 .................................................. 3
2. Applicability .................................................. 4
3. The LSP Self-ping Message ..................................... 5
4. LSP Self Ping Procedures ...................................... 6
5. Bidirectional LSP Procedures ................................... 8
6. IANA Considerations ............................................ 8
7. Security Considerations ....................................... 8
8. Contributors .................................................. 8
9. Acknowledgements .............................................. 9
10. References ................................................... 9
  10.1. Normative References ..................................... 9
  10.2. Informative References .................................. 10
Appendix A. Rejected Approaches ................................. 11
Authors’ Addresses .............................................. 11
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 a 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.

Referring to any LSR, RFC 3209 says, "The node SHOULD be prepared to forward packets carrying the assigned label prior to sending the RESV message". However, RFC 3209 does not strictly require this behavior.

Some implementations optimize the above-described procedure by allowing LSRs to send RESV messages before installing forwarding state [RFC6383]. 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 have not yet installed forwarding...
state. If the ingress LSR forwards traffic through the LSP before forwarding state has been installed on all downstream nodes, traffic can be lost.

This document 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 a new protocol. It is not an extension of LSP Ping [RFC4379]. Although LSP Ping and LSP Self-ping are named similarly, each is designed for a unique purpose. Each protocol listens on its own UDP port and executes its own procedures.

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

2. Applicability

LSP Self-ping is applicable in the following scenario:

- The ingress LSR signals a point-to-point LSP
- The ingress LSR receives a RESV message
- The RESV message indicates that all downstream nodes have begun the process of forwarding state installation
- The RESV message does not guarantee that all downstream nodes have completed the process of forwarding state installation
- The ingress LSR needs to confirm that all downstream nodes have completed the process for forwarding state installation
- 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
- Control plane resources on the egress LSR may be scarce
- 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 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. In another example, if a downstream LSR erroneously forwards a packet without an MPLS label, LSP Self-ping will not detect an error.

Furthermore, LSP Self-ping fails when either of the following conditions are true:

- The LSP under test is signaled by the Label Distribution Protocol (LDP) Independent Mode [RFC5036]
- 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:

- LSP Ping consumes control plane resources on the egress LSR
- 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 valuable 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 that encapsulates a session ID. 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:
The IP Source Address MAY be configurable. By default, it MUST be the address of the egress LSR

The IP Destination Address MUST be the address of the ingress LSR

The IP Time to Live (TTL) / Hop Count MAY be configurable. By default, it MUST be 255

The IP DSCP MAY be configurable. By default, it MUST be CS6 (Ox48) [RFC4594]

The UDP Source Port MUST be selected from the dynamic range (49152-65535) [RFC6335]

The UDP Destination Port MUST be lsp-self-ping (8503) [IANA.PORTS]

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Session-ID                             |
|                        (64 bits)                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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:

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

Implementations 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:

- Format a LSP Self-ping message.
- Set the Session-ID in the LSP Self-ping message to the Session-ID mentioned above
- Send the LSP Self-ping message through the LSP under test
- Set a timer to expire in Retry Timer milliseconds
- 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:

- The active side calculates ERO, signals LSP and runs LSP Self-ping
- The passive side reverses ERO, signals LSP and runs another instance of LSP Self-ping
- 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 its local LSP Self-ping returns TRUE. Endpoints are not required to wait until both LSP Self-ping sessions have returned TRUE.

6. IANA Considerations

IANA has assigned UDP Port Number 8503 [IANA.PORTS] for use by LSP Self-ping.

7. 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, operators SHOULD filter LSP Self-ping packets at the edges of the MPLS signaling domain. Furthermore, implementations SHOULD NOT assign Session-ID’s in a predictable manner. In order to avoid predictability, implementations can leverage a Cryptographically Secure Pseudo-random Number Generator (CSPRNG) [NIST-CSPRNG]

8. Contributors

The following individuals contributed significantly to this document:

Mark Wygant
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


10.2. Informative References


Appendix A. 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. Instead, 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.

Authors’ Addresses

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
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 a key goal of the MPLS Fabric architecture.

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].

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.

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 1, 2015.
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, RSVP and BGP, 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 Centre; 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.
manner. Existing signaling/routing protocols do not always meet this need. Labeled ARP (L-ARP) is a proposal to fill that gap.

[TODO-MPLS-FABRIC] describes the motivation for using MPLS as the fabric technology.

1.1. 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 query 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 query. All the nodes, except the node that owns the IP address, ignore the ARP query. 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. T is a member of the MPLS Fabric that may not be an L-ARP server. Within the MPLS Fabric, the usual MPLS protocols (IGP, 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.

```
      . . . . .
  . . . . .
H1 --- T1   T4
  \
  .   .    .
  .   .    .
  \ .   .   .
  \ .   .   .
  \ . Fabric .
H2 --- T2   T3 --- H3
      . . . . .
```

Figure 1

3.1. Basic Operation

A node (say H1) that needs an MPLS tunnel to a destination (say H3) broadcasts over all its interfaces an L-ARP query with the Target Protocol Address set to H3. A node that has reachability to H3 (such as T1 or T2) sends an L-ARP reply with the Source Hardware Address set to a locally-allocated MPLS label plus its Ethernet MAC address.
After receiving one or more L-ARP replies, H1 can select either T1 or T2 to send MPLS packets that are destined to H3. As described later, the L-ARP response may contain certain parameters that enable the client to make an informed choice of the routers.

As with standard ARP, the validity of the MPLS label obtained using L-ARP is time-bound. The client should periodically resend its L-ARP requests to obtain the latest information, and time out entries in its ARP cache if such an update is not forthcoming. Once an L-ARP server has advertised a label binding, it MUST NOT change the binding until expiry of the binding's validity time.

The mechanism defined here is simplistic; see Section 4.

3.2. Asynchronous operation

The preceding sections described a request-response based model. In some cases, the L-ARP server may want to asynchronously update its clients. L-ARP uses the gratuitous ARP model [RFC2002] to "push" such changes.

In a pure "push" model, a device may send out updates for all prefixes it knows about. This naive approach will not scale well. This memo specifies a mode of operation that is somewhere between "push" and "pull" model. An L-ARP server does not advertise any binding for a prefix until at least one L-ARP client expresses interest in that prefix (by initiating an L-ARP query). As long as the server has at least one interested client for a prefix, the server sends unsolicited (aka gratuitous, though the term is less appropriate in this context) L-ARP replies when a prefix's reachability changes. The server will deem the client's interest in a prefix to have ceased when it does not hear any L-ARP queries for some configured timeout period.

3.3. Client-Server Synchronization

In an L-ARP reply, the server communicates several pieces of information to the client: its hardware address, the MPLS label, Entropy Label capability and metric. Since ARP is a stateless protocol, it is possible that one of these changes without the client knowing, which leads to a loss of synchronization between the client and the server. This loss of synchronization can have several bad effects.

If the server's hardware address changes or the MPLS label is repurposed by the server for a different purpose, then packets may be sent to the wrong destination. The consequences can range from suboptimally routed packets to dropped packets to packets being
delivered to the wrong customer, which may be a security breach. This last may be the most troublesome consequence of loss of synchronization.

If a destination transitions from entropy label capable to entropy label incapable (an unlikely event) without the client knowing, then packets encapsulated with entropy labels will be dropped. A transition in the other direction is relatively benign.

If the metric changes without the client knowing, packets may be suboptimally routed. This may be the most benign consequence of loss of synchronization.

3.4. Applicability

L-ARP can be used between a host and its Top-of-Rack switch in a Data Center. L-ARP can also be used between a DSLAM and its aggregation switch going to the B-RAS. More generally, L-ARP can be used between an "access node" and its first hop MPLS-enabled device in the context of Seamless MPLS [reference]. In all these cases, L-ARP can handle the presence of multiple connections between the access device and its first hop devices.

ARP is not a routing protocol. The use of L-ARP should be limited to cases where the L-ARP client has a small number of one-hop connections to L-ARP servers. The presence of a complex topology between the L-ARP client and server suggests the use of a different protocol.

3.5. Backward Compatibility

Since L-ARP uses a new hardware type, it is backward compatible with "regular" ARP. ARP servers and clients MUST be able to send out, receive and process ARP messages based on hardware type. They MAY choose to ignore requests and replies of some hardware types; they MAY choose to log errors if they encounter hardware types they do not recognize; however, they MUST handle all hardware types gracefully. For hardware types that they do understand, ARP servers and clients MUST handle operation codes gracefully, processing those they understand, and ignoring (and possibly logging) others.

4. For Future Study

The L-ARP specification is quite simple, and the goal is to keep it that way. However, inevitably, there will be questions and features that will be requested. Some of these are:
1. Keeping L-ARP clients and servers in sync. In particular, dealing with:

   A. client and/or server restart
   B. lost packets
   C. timeouts

2. Withdrawing a response.

3. Dealing with scale.

4. If there are many servers, which one to pick?

5. How can a client make best use of underlying ECMP paths?

6. and probably many more.

In all of these, it is important to realize that, whenever possible, a solution that places most of the burden on the server rather than on the client is preferable.

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 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$pln octets)                  //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                     ar$lst (variable...)                   //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                     ar$att (variable...)                    //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: L-ARP Packet Format
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)

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.

Hardware Length: 6.

Protocol Address Length: for an IPv4 address, the value is 4; for an IPv6 address, it is 16.

Operation Code: set to 1 for request, 2 for reply, and 10 for ARP-NAK. Other op codes may be used as needed.

Source Hardware Address: In an L-ARP message, Source Hardware Address is the 6 octet sender’s MAC address.

Source Protocol Address: In an L-ARP message, this field carries the sender’s IP address.

Target Hardware Address: In an L-ARP query message, Target Hardware Address is the all-ones Broadcast MAC address; in an L-ARP reply message, it is the client’s MAC address.

Target Protocol Address: In an L-ARP message, this field carries the IP address for which the client is seeking an MPLS label.

Label Stack: In an L-ARP request, this field is empty. In an L-ARP reply, this field carries the MPLS label stack as an ARP TLV in the format below.

Attributes: In an L-ARP request, this field is empty. In an L-ARP reply, this field carries attributes for the MPLS label stack as an ARP TLV in the format below.

This document introduces the notion of ARP TLVs. These take the form as in Figure 3. Figure 4 describes the format of Label Stack TLV carried in L-ARP. Figure 5 describes the format of Attributes TLV carried in L-ARP.
Type is the type of the TLV; Length is the length of the value field in octets; Value is the value field.

Figure 3: ARP TLVs

Label Stack: Type = TLV-LST; Length = n*3 octets, where n is the number of labels. The Value field contains the MPLS label stack for the client to use to get to the target. Each label is 3 octets. This field is valid only in an L-ARP reply message.

E-bit: Entropy Label Capable: this flag indicates whether the corresponding label in the label stack can be followed 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.

Figure 4: MPLS Label Stack Format

Figure 5: Attribute TLV
Attributes TLV: Type = TLV-ATT; Length = 4 octets. The Value field contains the metric (typically, IGP distance) from the responder to the destination (device with the requested IP address). This field is valid only in an L-ARP reply message.

If other parameters are deemed useful in the ATT TLV, they will be added as needed.

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 the registry hrd) for HTYPE-MPLS.

IANA is also requested to create a new registry ARP-TLV ("tlv"). This is a registry of one octet numbers. Allocation policies: 0 is not to be allocated; the range 1-127 is Standards Action; the values 128-251 are FCFS; and the values 252-255 are Experimental.

Finally, IANA is requested to allocate two values in the ARP-TLV registry, one for TLV-LST and another for TLV-ATT.

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. Normative References


Authors’ Addresses

Kireeti Kompella
Juniper Networks
1194 N. Mathilda Avenue
Sunnyvale, CA  94089
USA
Email: kireeti.kompella@gmail.com

Balaji Rajagopalan
Juniper Networks
Prestige Electra, Exora Business Park
Marathahalli - Sarjapur Outer Ring Road
Bangalore  560103
India
Email: balajir@juniper.net
George Swallow  
Cisco Systems  
1414 Massachusetts Ave  
Boxborough, MA  01719  
US  

Email: swallow@cisco.com
Resilient MPLS Rings
draft-kompella-mpls-rmr-01

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 September 9, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

Rings are a very common topology in transport networks. A ring is the simplest topology offering link and node resilience. Rings are nearly ubiquitous in access and aggregation networks. As MPLS increases its presence in such networks, and takes on a greater role in transport, it is imperative that MPLS handles rings well; this is not the case today.
This document describes the special nature of rings, and the special needs of MPLS on rings. It then shows how these needs can be met in several ways, some of which involve extensions to protocols such as IS-IS [RFC5305], OSPF [RFC3630], RSVP-TE [RFC3209] and LDP [RFC5036].

1.1. Definitions

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

A ring is a subgraph of $G$. A ring consists of a subset of $n$ nodes $(R_i, 0 \leq i < n)$ of $V$. The directed edges $(R_i, R_{i+1})$ and $(R_{i+1}, R_i), 0 \leq i < n-1$ must be a subset of $E$ (note that index arithmetic is done modulo $n$). We define the direction from node $R_i$ to $R_{i+1}$ as "clockwise" (CW) and the reverse direction as "anticlockwise" (AC). As there may be several rings in a graph, we number each ring with a distinct ring ID $RID$.

```
R0 . . . R1
   .    .
R7    R2
Anti- | .    Ring | .
Clockwise | .   . | Clockwise
   v    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. An RID of 0 means the node is a "promiscuous" node.

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 connect ring neighbors.

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

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

- UN: 00 undefined link
- CW: 01 clockwise ring link
- AC: 10 anticlockwise ring link
- 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 overridden by configuration. For example, a link that would otherwise be classified by auto-discovery as a ring link might be configured not to be used for ring LSPs.

3.2. Ring Nodes

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

To simplify ring provisioning even further, a node N may be made "promiscuous" by being assigned an RID of 0. A promiscuous node listens to RIDs in its IGP neighbors' link-state updates. 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 lambda's or fibers between those nodes.

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. 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.

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.
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 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
```
OSPF Ring TLV Format

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV</td>
</tr>
<tr>
<td>----------------------------------</td>
</tr>
</tbody>
</table>

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)

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 re-advertising its ring node TLV with the M bit set.

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

4.4. Ring Identification Phase

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

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

A future version of this document will specify how ring changes are detected and handled.

5. Ring Signaling

A future version of this document will specify 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


10.2. Informative References


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/
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 ............................................. 3
   1.1. Terminology ........................................... 3
   1.2. Conventions used in this document .................. 4
2. Theory of Operation ...................................... 5
   2.1. Multi-path Label Switched Paths ...................... 5
   2.2. ECMP .................................................. 6
   2.3. Discussion ............................................ 8
   2.4. The Capabilities of TE-based Load Balancing ........ 9
3. Operation of MLSPs ....................................... 10
   3.1. Signaling MLSPs ....................................... 10
       3.1.1. Indicating Equi-bandwidth (EB) nature ........ 10
   3.2. Label Allocation ...................................... 10
   3.3. Bandwidth Accounting .................................. 10
   3.4. MLSP Data Plane Actions ............................... 11
4. Manageability ............................................ 13
5. Security Considerations ................................... 14
6. Acknowledgments .......................................... 15
7. IANA Considerations ..................................... 16
8. References ................................................ 17
    8.1. Normative References .................................. 17
    8.2. Informative References ............................... 17
Authors’ Addresses ......................................... 19
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

An example network illustrating ECMP. Assume that paths AMB, AXYPTB, AXYQTB, AXYRB and AXSB all have the same path length (cost).

![Diagram of an example network illustrating ECMP.](image)

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.

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 120Gbps of traffic to S and M sends 180Gbps to S, S can redistribute L’s traffic by sending 40Gbps to each of P, Q and R; and can similarly send 60Gbps of M’s traffic to each of P, Q and R. Alternatively, S can load balance the aggregate 300Gbps of traffic received from L and M to each of P, Q and R, thus sending 100Gbps 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 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


8.2. Informative References


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
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 ............................................... 3
   1.1. Conventions used in this document .................. 3
         1.1.1. Terminology .................................. 3
         1.1.2. Requirements Language ....................... 4
2. Residence Time Measurement ................................. 4
3. G-ACh for Residence Time Measurement ..................... 4
   3.1. PTP Packet Sub-TLV .................................. 6
4. Control Plane Theory of Operation ........................ 7
   4.1. RTM Capability ..................................... 7
   4.2. RTM Capability Sub-TLV ............................. 8
   4.3. RTM Capability Advertisement in OSPFv2 .......... 9
   4.4. RTM Capability Advertisement in OSPFv3 .......... 9
   4.5. RTM Capability Advertisement in IS-IS ........... 9
   4.6. RSVP-TE Control Plane Operation to Support RTM .... 10
   4.7. RTM_SET Object .................................... 11
         4.7.1. RSO Sub-objects ............................. 12
5. Data Plane Theory of Operation ........................... 14
6. Applicable PTP Scenarios ................................ 15
7. One-step Clock and Two-step Clock Modes .................. 15
8. IANA Considerations .................................... 17
   8.1. New RTM G-ACh .................................... 17
   8.2. New RTM TLV Registry ............................. 18
   8.3. New RTM Sub-TLV Registry .......................... 18
   8.4. RTM Capability sub-TLV ............................ 19
   8.5. IS-IS RTM Application ID .......................... 19
   8.6. RTM_SET Object RSVP Class Number, Class Type and Sub-
        object Types .................................... 19
9. Security Considerations ................................. 20
10. Acknowledgements ...................................... 21
11. References ........................................... 21
    11.1. Normative References ............................. 21
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
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 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

- First four octets are defined as G-ACh Header in [RFC5586]
- The Version field is set to 0, as defined in RFC 4385 [RFC4385].
- The Reserved field MUST be set to 0 on transmit and ignored on receipt.
- The RTM G-ACh field, value to be allocated by IANA, identifies the packet as such.
- 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.
- 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.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Type</td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flags</td>
<td>PTPType</td>
<td></td>
</tr>
<tr>
<td>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Port ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sequence ID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Figure 2: PTP Sub-TLV format

where Flags field has format

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>Reserved</td>
</tr>
<tr>
<td>+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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].
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

```
+----------------------------------+
| Type(TBA5) | Length |
+----------------------------------+
| RTM | Reserved |
+----------------------------------+
```

Figure 4: RTM Capability sub-TLV

- Type value will be assigned by IANA from appropriate registries.
- Length MUST be set to 4.
- 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.
- 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:

- The S bit MUST be cleared to prevent the RTM Capability sub-TLV from leaking between levels.
- The D bit of the Flags field MUST be cleared as required by [RFC6823].
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

- **A** is a Boundary Clock with its egress port in Master state. Node A transmits PTP messages;
- **B** is the ingress LER for the MPLS LSP and is not RTM capable;
- **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;
- **D** is a transit LSR that is not RTM capable; it passes along the RTM ACH encapsulated PTP message unmodified;
- **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;
- **F** is the egress LER for the MPLS LSP and is not RTM capable;
- **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 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 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Flags</th>
<th>IPv6 address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>20</td>
<td>TBD</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Flags</th>
<th>Router ID</th>
<th>Interface ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>20</td>
<td>TBD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 request-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:
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>This document</td>
</tr>
<tr>
<td>1</td>
<td>No payload</td>
<td>This document</td>
</tr>
<tr>
<td>2</td>
<td>PTPv2</td>
<td>This document</td>
</tr>
<tr>
<td>3</td>
<td>NTP</td>
<td>This document</td>
</tr>
<tr>
<td>4-127</td>
<td>Reserved</td>
<td>IETF Consensus</td>
</tr>
<tr>
<td>128 - 191</td>
<td>Reserved</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>192 - 255</td>
<td>Reserved</td>
<td>Private Use</td>
</tr>
</tbody>
</table>

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:
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>This document</td>
</tr>
<tr>
<td>1</td>
<td>PTP 2-step</td>
<td>This document</td>
</tr>
<tr>
<td>2-127</td>
<td>Reserved</td>
<td>IETF Consensus</td>
</tr>
<tr>
<td>128 - 191</td>
<td>Reserved</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>192 - 255</td>
<td>Reserved</td>
<td>Private Use</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA2</td>
<td>RTM Capability</td>
<td>This document</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA3</td>
<td>RTM</td>
<td>This document</td>
</tr>
</tbody>
</table>

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:
Table 6: RTM_SET object Class

IANA is requested to assign a new Class Type for RTM_SET object as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA4</td>
<td>RTM_SET object</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 7: RTM_SET object Class Type

IANA requested to create new sub-registry for sub-object types of RTM_SET object as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>IPv4 address</td>
<td>This document</td>
</tr>
<tr>
<td>2</td>
<td>IPv6 address</td>
<td>This document</td>
</tr>
<tr>
<td>3</td>
<td>Unnumbered interface</td>
<td>This document</td>
</tr>
<tr>
<td>4-127</td>
<td>Reserved</td>
<td>IETF Consensus</td>
</tr>
<tr>
<td>128 - 191</td>
<td>Reserved</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>192 - 255</td>
<td>Reserved</td>
<td>Private Use</td>
</tr>
</tbody>
</table>

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 thoughtful comments.

11. References

11.1. Normative References

[I-D.ietf-ospf-ospfv3-lsa-extend]

[I-D.ietf-ospf-prefix-link-attr]

[IEEE.1588.2008]

11.2. Informative References


[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.


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

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.
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TBD-1)</td>
<td>Bypass Summary FRR Association (B-SFRR)</td>
</tr>
</tbody>
</table>

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:
### IPv4 Extended Association ID field

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-15</td>
<td>Bypass_Tunnel_ID</td>
</tr>
<tr>
<td>16-31</td>
<td>Reserved</td>
</tr>
<tr>
<td>32-63</td>
<td>Bypass_Source_IPv4_Address</td>
</tr>
<tr>
<td>64-95</td>
<td>Bypass_Destination_IPv4_Address</td>
</tr>
<tr>
<td>96-127</td>
<td>Bypass_Group_Identifier</td>
</tr>
<tr>
<td>128</td>
<td>MESSAGE_ID</td>
</tr>
</tbody>
</table>

**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].

### IPv6 Extended Association ID

The IPv6 Extended Association ID field for the Summary FRR information has the following format:
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 a 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)
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:
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


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
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 ................................................. 3
2. Specification of Requirements ............................... 3
3. LDP YANG Model ............................................. 3
   3.1. Overview ............................................... 4
   3.2. Configuration ......................................... 7
      3.2.1. Configuration Hierarchy .......................... 11
      3.2.2. All-VRFs Configuration .......................... 14
   3.3. Operational State ..................................... 14
      3.3.1. Derived States .................................. 21
   3.4. Notifications ......................................... 26
   3.5. Actions .............................................. 26
4. mLDP YANG Model ............................................. 27
   4.1. Overview ............................................... 27
   4.2. Configuration ......................................... 28
      4.2.1. Configuration Hierarchy .......................... 28
      4.2.2. mldp container .................................. 30
      4.2.3. Leveraging LDP containers ........................ 31
      4.2.4. YANG tree ...................................... 31
   4.3. Operational State ..................................... 33
      4.3.1. Derived states .................................. 38
   4.4. Notifications ......................................... 42
   4.5. Actions .............................................. 43
5. Open Items .................................................. 43
6. YANG Specification .......................................... 43
7. Security Considerations ..................................... 110
8. IANA Considerations ....................................... 110
9. Acknowledgments ........................................... 110
10. References ................................................ 110
   10.1. Normative References .................................. 110
   10.2. Informative References ............................... 113
Appendix A. Additional Contributors .......................... 113
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:

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

- Read-Write parameters for configuration (Discussed in Section 3.2)
- Read-only parameters for operational state (Discussed in Section 3.3)
- Notifications for events (Discussed in Section 3.4)
- 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:
Before going into data model details, it is important to take note of the following points:

- This module aims to address only the core LDP/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.

- Multi-topology LDP [RFC7307] and Multi-topology mLDP [I-D.iwijnand-mpls-mldp-multi-topology] are beyond the scope of this document.

- 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.

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

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

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

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

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

- 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.

- 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:
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-assignment)?
          ---rw enable? boolean
        ---rw graceful-restart
          ---rw enable? boolean
          ---rw helper-enable? boolean (graceful-restart-helper-mod)
        ---rw reconnect-time? uint16
        ---rw recovery-time? uint16
        ---rw forwarding-holdtime? uint16
        ---rw igp-synchronization-delay? uint16
        ---rw lsr-id? yang:dotted-quad
```
Internet-Draft      YANG Data Model for LDP and mLDP           July 2016

| +--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-timer-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
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-assignment)?
            | | +--ro enable? boolean
            | +--ro graceful-restart
              +--ro enable? boolean
              +--ro helper-enable? boolean (graceful-restart-helper-mod)
              +--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 ipv6
    +++-ro transport-address? inet:ipv6-address
    +++-ro binding
      |    +++-ro address* [address]
      |      |    +++-ro address inet:ipv6-address
      |      |    +++-ro advertisement-type? advertised-received
Internet-Draft      YANG Data Model for LDP and mLDP           July 2016

|  |  |  +--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
|  +--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-timer-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 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 ipv4
  ++--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 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
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.
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.
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.
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.
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:

- mLDP Base Specification [RFC6388]
- mLDP Recursive FEC [RFC6512]
- Targeted mLDP [RFC7060]
- mLDP Fast-Reroute (FRR)
  * Node Protection [RFC7715]
  * Multicast-only
- Hub-and-Spoke Multipoint LSPs [RFC7140]
- mLDP In-band Signaling [RFC6826] (future revision)
- 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:
Parameters that leverage/extend LDP containers and parameters

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:

- 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.

- 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:

- **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.

- **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.

- **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?
            +--rw enable?  boolean
            +--rw merge-point
              +--rw enable?  boolean
              +--rw targeted-session-teardown-delay?  uint16
        +--rw mldp {mldp}?
          |  +--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-lsp
            +--rw p2mp
              |  +--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-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 root* [root-address]
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:

```yang
define module ietf-mpls-ldp
  define 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
            +--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-address 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-address 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
      |  |  +++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 recur-root-address recur-rd]
      |  |  |      +++ro root-address  inet:ipv4-address
      |  |  |      +++ro rp         inet:ip-address
      |  |  |      +++ro group-address  inet:ip-address-no-zon
      |  |  |  +++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-address recur-rd]
          ++-ro root-address inet:ipv6-address
          ++-ro lsp-id uint32
          ++-ro recur-root-address inet:ipv6-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-address rd recur-root-address recur-rd]
          ++-ro root-address inet:ipv6-address
          ++-ro source-address inet:ipv6-address
          ++-ro group-address inet:ip-address-no-zon
          ++-ro rd route-distinguisher
          ++-ro recur-root-address inet:ipv6-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 recur-root-address recur-rd]
          ++-ro root-address inet:ipv6-address
          ++-ro rp inet:ipv6-address
          ++-ro group-address inet:ip-address-no-zon
          ++-ro rd route-distinguisher
          ++-ro recur-root-address inet:ipv6-address
          ++-ro recur-rd route-distinguisher
4.3.1. Derived states

Following are main areas for which mLDP operational derived state is defined:

- Root
- Bindings (FEC-label)
- 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-family, 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:
<table>
<thead>
<tr>
<th>Opaque Type</th>
<th>Key</th>
<th>RFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic LSP Identifier</td>
<td>LSP Id</td>
<td>[RFC6388]</td>
</tr>
<tr>
<td>Transit IPv4 Source</td>
<td>Source, Group</td>
<td>[RFC6826]</td>
</tr>
<tr>
<td>Transit IPv6 Source</td>
<td>Source, Group</td>
<td>[RFC6826]</td>
</tr>
<tr>
<td>Transit IPv4 Bidir</td>
<td>RP, Group</td>
<td>[RFC6826]</td>
</tr>
<tr>
<td>Transit IPv6 Bidir</td>
<td>RP, Group</td>
<td>[RFC6826]</td>
</tr>
<tr>
<td>Transit VPNv4 Source</td>
<td>Source, Group, RD</td>
<td>[RFC7246]</td>
</tr>
<tr>
<td>Transit VPNv6 Source</td>
<td>Source, Group, RD</td>
<td>[RFC7246]</td>
</tr>
<tr>
<td>Transit VPNv4 Bidir</td>
<td>RP, Group, RD</td>
<td>[RFC7246]</td>
</tr>
<tr>
<td>Transit VPNv6 Bidir</td>
<td>RP, Group, RD</td>
<td>[RFC7246]</td>
</tr>
<tr>
<td>Recursive Opaque</td>
<td>Root</td>
<td>[RFC6512]</td>
</tr>
<tr>
<td>VPN-Recursive Opaque</td>
<td>Root, RD</td>
<td>[RFC6512]</td>
</tr>
</tbody>
</table>

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:

```yang
+--rw mpls-ldp!
  +--rw mldp
    +--rw address-family* [afi]
      +--rw afi address-family
        +--ro state
          +--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-recursive) 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, recur-RD] and VPN-recursive type will specify [R, S,G, -, recur-R, recur-RD].

It is important to take note of the following:

- 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.

- 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]

- 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)?  inet:ip-address
      +--:(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:

- Close on augmentation off "mpls" list in "ietf-mpls" defined in [I-D.saad-mpls-base-yang]
- Align operational state modeling with other routing protocols and [I-D.openconfig-netmod-opstate]
- Complete the section on Protocol-centric implementations and all-vrf
- Specify default values for configuration parameters
- Revisit and cut down on the scope of the document and number of features it is trying to cover
- Split the model into a base and extended items
- Add statistics for mLDP root LSPs and bindings
- Extend the "Configured Leaf LSPs" for various type of opaque-types
- Extend mLDP notifications for other types of opaque values as well
- 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
This YANG module defines the essential components for the management of Multi-Protocol Label Switching (MPLS) Label Distribution Protocol (LDP) and Multipoint LDP (mLDP).

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

/*
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;
  }
} // binding-address-state-attributes
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
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.";
    }
} // binding-label-state-attributes
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."
        } // enable
        leaf helper-enable {
            if-feature graceful-restart-helper-mode;
            type boolean;
            description "Enable or disable graceful restart helper mode."
        } // helper-enable
        leaf reconnect-time {
            type uint16 {
                range 10..1800;
            } // reconnect-time
            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."
        } // reconnect-time
        leaf recovery-time {
            type uint16 {
                range 30..3600;
            } // recovery-time
            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."
        } // recovery-time
        leaf forwarding-holdtime {
            type uint16 {
                range 10..1800;
            } // forwarding-holdtime
            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."
        } // forwarding-holdtime
    } // graceful-restart
} // graceful-restart-attributes
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;
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
    }
  }
}
grouping ldp-fec-event {
    description
    "A LDP FEC event.";
    leaf prefix {
        type inet:ip-prefix;
        description
        "FEC.";
    }
}

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.";
    }
}

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:ipv6-address;
        description
        "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 multipoint, 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.";
  }
}
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
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.";
    }
}

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
      } // assign
    } // independent-mode
  } // label-policy
} // policy-container
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 {
    description "";
  }
} // statistics-peer-received-sent

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.";
  }
}

container recursive-fec {
  description
  "Recursive FEC policy.";
  leaf prefix-list {
    type prefix-list-ref;
    description
    "Enables recursive FEC for the specified access list.";
  }
}

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
        "Whether the root is self.";
      }
    }
  }
}

"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

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;
}

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
container policies {
  description "LDP policy information.";

  leaf policy-container {
    type leafref;
    description "LDP policy container.";
  }
}

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 {
      description "LDP address and label binding information.";
    }
  }

  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.";
    }
  }
}
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  
        "Address binding.
..."
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 "Neighbor discovery configuration.";
}

container interfaces {
  description "A list of interfaces for basic discovery.";
  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 "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

"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."
        }
      }
    }
  }
}

Internet-Draft YANG Data Model for LDP and mLDP July 2016

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]/"
  + "ipv: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.";
        }
      }
    }
  }
}
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."
  }
}

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 {

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;
}

container ipv4 {
  when "../../afi = 'ipv4'" {
    description
    "LDP peer from this adjacency.";
  }
  description
  "LDP peer from this adjacency.";
} // hello-adjacencies
} // ipv6
} // state

container ipv4 {
  when "../../afi = 'ipv4'" {
    description
    "LDP peer from this adjacency.";
  }
  description
  "LDP peer from this adjacency.";
} // hello-adjacencies
} // ipv4
} // state

"For IPv4."

} // list

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

class 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
}
namespace ldp;

container 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."
    }
  }
} // targeted

} // address-family

} // ipv6

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
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";
    }
  }
}

crpc mpls-ldp-clear-hello-adjacency {
  description
    "Clears the hello adjacency";
  input {
    container hello-adjacency {
      description
        "Link adjacency or targeted adjacency. If this is not
         provided then all hello adjacencies are cleared";
      choice hello-adjacency-type {
        description "Adjacency type.";
        case targeted {
          container targeted {
            presence "Present to clear targeted adjacencies.";
            description
              "Clear targeted adjacencies.";
            leaf target-address {
              type inet:ip-address;
              description
                "The target address. If this is not provided then
                 all targeted adjacencies are cleared";
            }
          } // targeted
        }
        case link {
          container link {
            // Link adjacency
          }
        }
      }
    }
  }
}
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>
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 specification 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]

[I-D.rtgyangdt-rtgwg-ni-model]

[I-D.saad-mpls-base-yang]


10.2. Informative References


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
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.
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] 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
      +--: (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)          empty
        | +--:(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)           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
      |  ++--:(vplsp)
      |     |  ++--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)?
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 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 string;
    description "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{
        description "Used together with the corresponding data-size value to determine how to fill the data portion of a probe packet.";
    }
}
leaf description {
    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";
    }
  }
}
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:

```xml
<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-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:

```xml
<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>
      </result-info>
      <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>
        </probe-result>
      </probe-results>
    </lsp-ping>
  </lsp-pings>
</data>
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
12. References

12.1. Normative References


12.2. Informative References


Authors' Addresses

Lianshu Zheng
Huawei Technologies
China

Email: vero.zheng@huawei.com
