MPLS Working Group                                         S. Bryant
Internet-Draft                                               M. Chen
Intended status: Standards Track                              Z. Li
Expires: October 28, 2017                                    Huawei
                                                         G. Swallow
                                                       S. Sivabalan
                                                      Cisco Systems
                                                          G. Mirsky
                                                          ZTE Corp.
                                                        G. Fioccola
                                                      Telecom Italia
                                                      April 26, 2017

RFC6374 Synonymous Flow Labels
draft-bryant-mpls-rfc6374-sfl-04

Abstract

   This document describes a method of making RFC6374 performance
   measurements on flows carried over an MPLS Label Switched path.  This
   allows loss and delay 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

Copyright Notice

Table of Contents

1.  Introduction

   [RFC6374] was originally designed for use as an OAM protocol for use
   with MPLS Transport Profile (MPLS-TP) [RFC5921] LSPs.  MPLS-TP only
   supports point-to-point and point-to-multi-point LSPs.  This document
   describes how to use RFC6374 in the general MPLS case, and also
   introduces a number of more sophisticated measurements of
   applicability to both cases.

[I-D.ietf-mpls-flow-ident] describes the requirement for introducing
flow identities when using RFC6374 [RFC6374] packet Loss Measurements
(LM).  In summary RFC6374 uses the loss-measurement (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 PWs) 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.

[I-D.bryant-mpls-sfl-framework] describes a method of encoding per
flow instructions in an MPLS label stack using a technique called
Synonymous Flow Labels (SFL) in which labels which mimic the
behaviour of other labels provide the packet batch identifiers and
enable the per batch packet accounting.  This memo specifies how SFLs

are used to perform RFC6374 packet loss and RFC6374 delay
measurements.

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.  RFC6374 Packet Loss Measurement with SFL

The data service packets of the flow being instrumented are grouped
into batches, and all the packets within a batch are marked with the
SFL [I-D.ietf-mpls-flow-ident] corresponding to that batch.  The
sender counts the number of packets in the batch.  When the batch has
completed and the sender is confident that all of the packets in that
batch will have been received, the sender issues an RFC6374 Query
message to determine the number actually received and hence the
number of packets lost.  The RFC6374 Query message is sent using the
same SFL as the co-responding batch of data service packets.  The
format of the Query and Response packet is described in Section 9.

4.  RFC6374 Single Packet Delay Measurement

RFC6374 describes how to measure the packet delay by measuring the
transit time of an RFC6374 packet over an LSP.  Such a packet may not
need to be carried over an SFL since the delay over a particular LSP
should be a function of the TC bits.

However where SFLs are being used to monitor packet loss or where
label inferred scheduling is used [RFC3270] then the SFL would be
REQUIRED to ensure that the RFC6374 packet which was being used as a
proxy for a data service packet experienced a representative delay.
The format of an RFC6374 packet carried over the LSP using an SFL is
shown in Section 9.

5.  Data Service Packet Delay Measurement

Where it is desired to more thoroughly instrument a packet flow and
to determine the delay of a number of packets it is undesirable to
send a large number of RFC6374 packets acting as proxy data service
packets Section 4.  A method of directly measuring the delay
characteristics of a batch of packets is therefore needed.

Given the long intervals over which it is necessary to measure packet
loss, it is not necessarily the case that the batch times for the two
measurement types would be identical.  This it is proposed that the

two measurements are relatively independent.  The notion that they
are relatively independent arises for the potential for the two
batches to overlap in time, in which case either the delay batch time
will need to be cut short or the loss time will need to be extended
to allow correct reconciliation of the various counters.

The problem is illustrated in Figure 1 below:

(1) AAAAAAAAAABBBBBBBBBBAAAAAAAAAABBBBBBBBBB

 SFL Marking of a packet batch for loss measurement

(2) AADDDDAAAABBBBBBBBBBAAAAAAAAAABBBBBBBBBB

 SFL Marking of a subset if the packets for delay

(3) AAAAAAADDDDBBBBBBBBBAAAAAAAAAABBBBBBBBBB

 SFL Marking of a subset of the packets across a
 packet loss measurement boundary

(4) AACDCDCDAABBBBBBBBBBAAAAAAAAAABBBBBBBBBB

 The case of multiple delay measurements within
 a packet loss measurement

                Figure 1: RFC6734 Query Packet with SFL

In case 1 of Figure 1 we show the case were loss measurement alone is
being carried out on the flow under analysis.  For illustrative
purposes consider that in the time interval being analyzed, 10
packets always flow.

Now consider case 2 of Figure 1 where a small batch of packets need
to analyzed for delay.  These are marked with a different SFL type
indicating that they are to be monitored for both loss and delay.
The SFL=A indicates loss batch A, SFL=D indicates a batch of packets
that are to be instrumented for delay, but SFL D is synonymous with
SFL A, which in turn is synonymous with the underlying FEC.  Thus a
packet marked D will be accumulated into the A loss batch, into the
delay statistics and will be forwarded as normal.  Whether the packet
is actually counted twice (for loss and delay) or whether the two
counters are reconciled during reporting is a local matter.

Now consider case 3 of Figure 1 where a small batch of packets are
marked for delay across a loss batch boundary.  These packets need to
considered as part of batch A or a part of batch B, and any RFC6374

Query needs to take place after all the packets A or D (which ever
option is chosen) have arrived at the receiving LSR.

Now consider case 4 of Figure 1.  Here we have a case where it is
required to take a number of delay measurements within a batch of
packets that we are measuring for loss.  To do this we need two SFLs
for delay (C and D) and alternate between them (on a delay batch by
delay batch basis) for the purposes of measuring the delay
characteristics of the different batches of packets.

6.  Some Simplifying Rules

It is possible to construct a large set of overlapping measurement
type, in terms of loss, delay, loss and delay and batch overlap.  If
we allow all combination of cases, this leads to configuration,
testing and implementation complexity and hence increased operation
and capital cost.  The following simplifying rules represent the
default case:

1.  Any system that needs to measure delay MUST be able to measure
    loss.

2.  Any system that is to measure delay MUST be configured to measure
    loss.  Whether the loss statistics are collected or not is a
    local matter.

3.  A delay measurement MAY start at any point during a loss
    measurement batch, subject to rule 4.

4.  A delay measurement interval MUST be short enough that it will
    complete before the enclosing loss batch completes.

5.  The duration of a second delay (D in Figure 1 batch must be such
    that all packets from the packets belonging to a first delay
    batch (C in Figure 1)will have been received before the second
    delay batch completes.

Given that the sender controls both the start and duration of a loss
and a delay packet batch, these rules are readily implemented in the
control plane.

7.  Multiple Packet Delay Characteristics

A number of methods are described.  The expectation is that the MPLS
WG possibly with the assistance of the IPPM WG will select one or
maybe more than one of these methods for standardization.

Three Methods are discussed:

   1.  Time Buckets

   2.  Classic Standard Deviation

   3.  Average Delay

7.1.  Method 1: Time Buckets

   In this method the receiving LSR measures the inter-packet gap,
   classifies the delay into a number of delay buckets and records the
   number of packets in each bucket.  As an example, if the operator
   were concerned about packets with a delay of up to 1us, 2us, 4us,
   8us, and over 8us then there would be five buckets and packets that
   arrived up to 1us would cause the 1us bucket counter to increase,
   between 1us and 2us the 2us bucket counter would increase etc.  In
   practice it might be better in terms of processing and potential
   parallelism if, when a packet had a delay relative to its predecessor
   of 2us both the up to 1us and the 2us counter were incremented and
   any more detailed information was calculated in the analytics system.

   This method allows the operator to see more structure in the jitter
   characteristics than simply measuring the average jitter, and avoids
   the complication of needing to perform a per packet multiply, but
   will probably need to time intervals between buckets to be
   programmable by the operator.

   The packet format of an RFC6374 Bucket Jitter Measurement Message is
   shown below:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Flags | Control Code  |          Message Length       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  QTF  |  RTF  | RPTF  |              Reserved                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Session Identifier            |    DS     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Number of     |          Reserved                             |
| Buckets       |                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Interval in 10ns units                     |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Number pkts in Bucket                      |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
~                         TLV Block                             ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
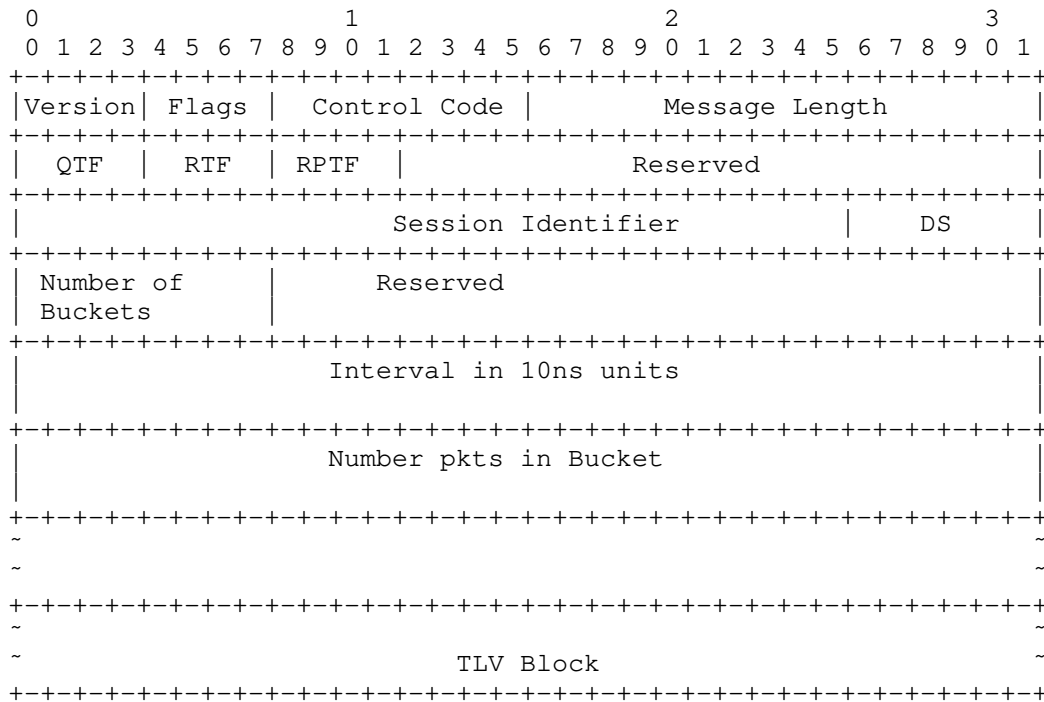
              Figure 2: Bucket Jitter Measurement Message Format

   The Version, Flags, Control Code, Message Length, QTF, RTF, RPTF,
   Session Identifier, and DS Fields are as defined in section 3.7 of
   RFC6374.  The remaining fields are as follows:

   o Number of Buckets in the measurement

   o Reserved must be sent as zero and ignored on receipt

   o Interval in 10ns units is the inter-packet interval for
     this bucket

   o Number Pkts in Bucket is the number of packets found in
     this bucket.

   There will be a number of Interval/Number pairs depending on the
   number of buckets being specified by the Querier.  If an RFC6374
   message is being used to configure the buckets, (i.e. the responder
   is creating or modifying the buckets according to the intervals in
   the Query message), then the Responder MUST respond with 0 packets in
   each bucket until it has been configured for a full measurement
   period.  This indicates that it was configured at the time of the

last response message, and thus the response is valid for the whole
interval.  As per the [RFC6374] convention the Number of pkts in
Bucket fields are included in the Query message and set to zero.

Out of band configuration is permitted by this mode of operation.

Note this is a departure from the normal fixed format used in
RFC6374.  We need to establish if this is problematic or not.

This RFC6374 message is carried over an LSP in the way described in
[RFC6374] and over an LSP with an SFL as described in Section 9.

7.2.  Method 2 Classic Standard Deviation

In this method, provision is made for reporting the following delay
characteristics:

1.  Number of packets in the batch (n).

2.  Sum of delays in a batch (S)

3.  Maximum Delay.

4.  Minimum Delay.

5.  Sum of squares of Inter-packet delay (SS).

Characteristic's 1 and 2 give the mean delay.  Measuring the delay of
each pair in the batch is discussed in Section 7.3.

Characteristics 3 and 4 give the outliers.

Characteristics 1, 2 and 5 can be used to calculate the variance of
the inter-packet gap and hence the standard deviation giving a view
of the distribution of packet delays and hence the jitter.  The
equation for the variance (var) is given by:

var = (SS - S*S/n)/(n-1)

There is some concern over the use of this algorithm for measuring
variance, because SS and S*S/n can be similar numbers, particularly
where variance is low.  However the method commends it self by not
requiring a division in the hardware.  A future version of this
document will look at using improved statistical methods such as the
assumed mean.

7.2.1.  RFC6374 Multi-Packet Delay Measurement Message Format

   The packet format of an RFC6374 Multi-Packet Delay Measurement
   Message is shown below:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version| Flags | Control Code  |         Message Length        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  QTF  |  RTF  | RPTF  |                 Reserved              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Session Identifier         |      DS       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Number of Packets                        |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Sum of Delays for Batch                    |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Minimum Delay                          |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Maximum Delay                          |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |              Sum of squares of Inter-packet delay             |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                                                               ~
   ~                         TLV Block                             ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
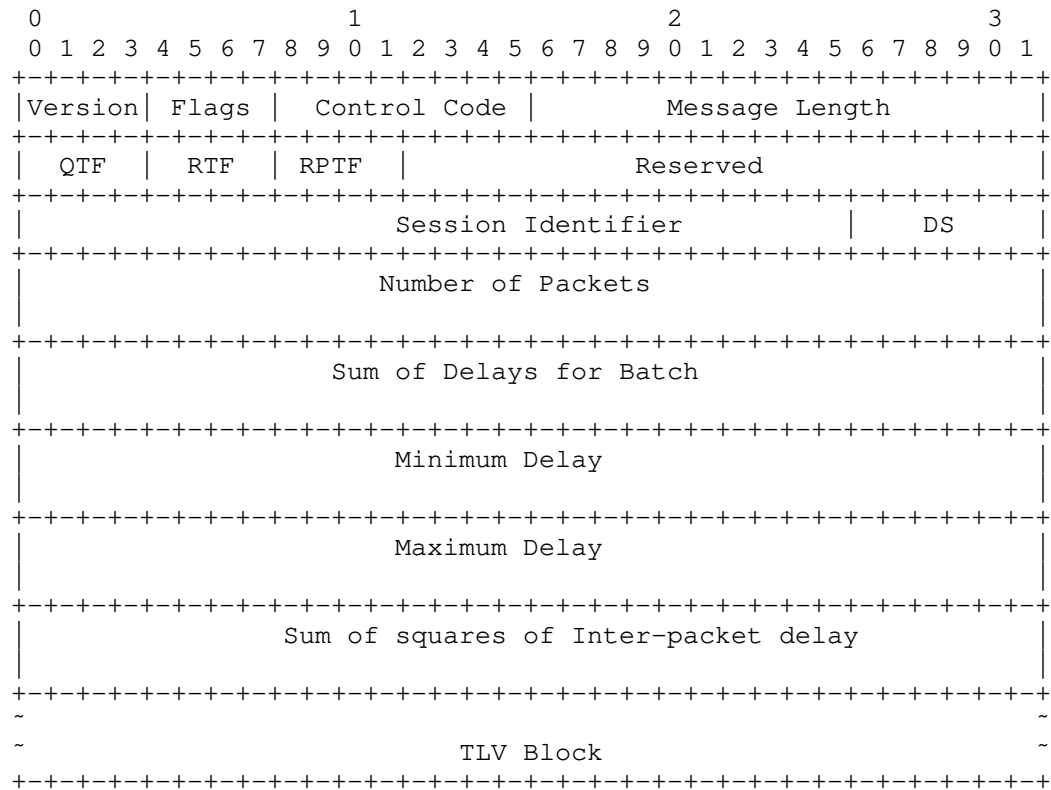
          Figure 3: Multi-packet Delay Measurement Message Format

   The Version, Flags, Control Code, Message Length, QTF, RTF, RPTF,
   Session Identifier, and DS Fields are as defined in section 3.7 of
   RFC6374.  The remaining fields are as follows:

   o Number of Packets is the number of packets in this batch

   o Sum of Delays for Batch is the duration of the batch in the
     time measurement format specified in the RTF field.

   o Minimum Delay is the minimum inter-packet gap observed during
     the batch in the time format specified in the RTF field.

   o Maximum Delay is the maximum inter-packet gap observed during
     the batch in the time format specified in the RTF field.

   This RFC6374 message is carried over an LSP in the way described in
   [RFC6374] and over an LSP with an SFL as described in Section 9.

## 7.3.  Per Packet Delay Measurement

   If detailed packet delay measurement is required then it might be
   possible to record the inter-packet gap for each packet pair.  In
   other that exception cases of slow flows or small batch sizes, this
   would create a large demand on storage in the instrumentation system,
   bandwidth to such a storage system and bandwidth to the analytics
   system.  Such a measurement technique is outside the scope of this
   document.

## 7.4.  Average Delay

   Introduced in [I-D.ietf-ippm-alt-mark] is the concept of a one way
   delay measurement in which the average time of arrival of a set of
   packets is measured.  In this approach the packet is time-stamped at
   arrival and the Responder returns the sum of the time-stamps and the
   number of times-tamps.  From this the analytics engine can determine
   the mean delay.  An alternative model is that the Responder returns
   the time stamp of the first and last packet and the number of
   packets.  This method has the advantage of allowing the average delay
   to be determined at a number of points along the packet path and
   allowing the components of the delay to be characterized.

   The packet format of an RFC6374 Average Delay Measurement Message is
   shown below:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Flags |  Control Code |          Message Length       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  QTF  |  RTF  | RPTF  |              Reserved                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Session Identifier            |      DS       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Number of Packets                        |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Time of First Packet                      |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Time of Last Packet                      |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Sum of Timestamps of Batch                  |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
~                                                               ~
~                           TLV Block                           ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
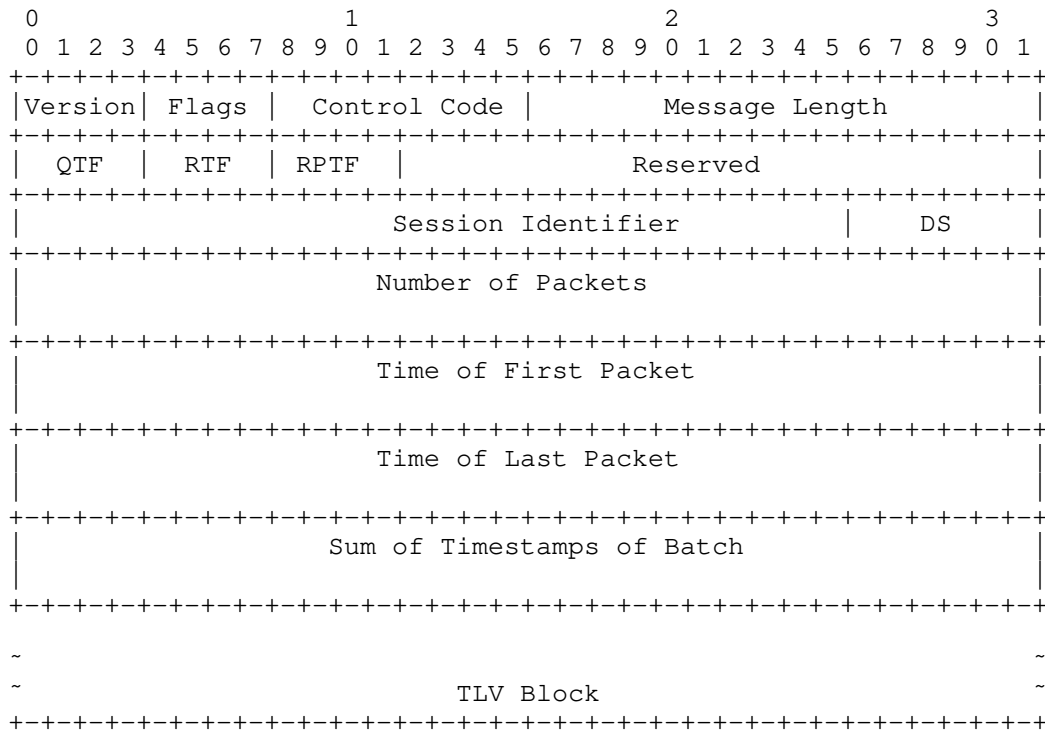
        Figure 4: Average Delay Measurement Message Format

The Version, Flags, Control Code, Message Length, QTF, RTF, RPTF,
Session Identifier, and DS Fields are as defined in section 3.7 of
RFC6374.  The remaining fields are as follows:

o Number of Packets is the number of packets in this batch.

o Time of First Packet is the time of arrival of the first
  packet in the batch.

o Time of Last Packet is the time of arrival of the last
  packet in the batch.

o Sum of Timestamps of Batch.

This RFC6374 message is carried over an LSP in the way described in
[RFC6374] and over an LSP with an SFL as described in Section 9.  As
is the convention with RFC6374, the Query message contains
placeholders for the Response message.  The placeholders are sent as
zero.

8.  Sampled Measurement

   In the discussion so far it has been assumed that we would measure
   the delay characteristics of every packet in a delay measurement
   interval defined by an SL of constant colour.  In
   [I-D.ietf-ippm-alt-mark] the concept of a sampled measurement is
   considered.  That is the Responder only measures a packet at the
   start of a group of packets being marked for delay measurement by a
   particular colour, rather than every packet in the marked batch.  A
   measurement interval is not defined by the duration of a marked batch
   of packets but the interval between a pair of RFC6374 packets taking
   a readout of the delay characteristic.  This approach has the
   advantage that the measurement is not impacted by ECMP effects.

9.  Carrying RFC6374 Packets over an LSP using an SFL

   The packet format of an RFC6374 Query message using SFLs is shown in
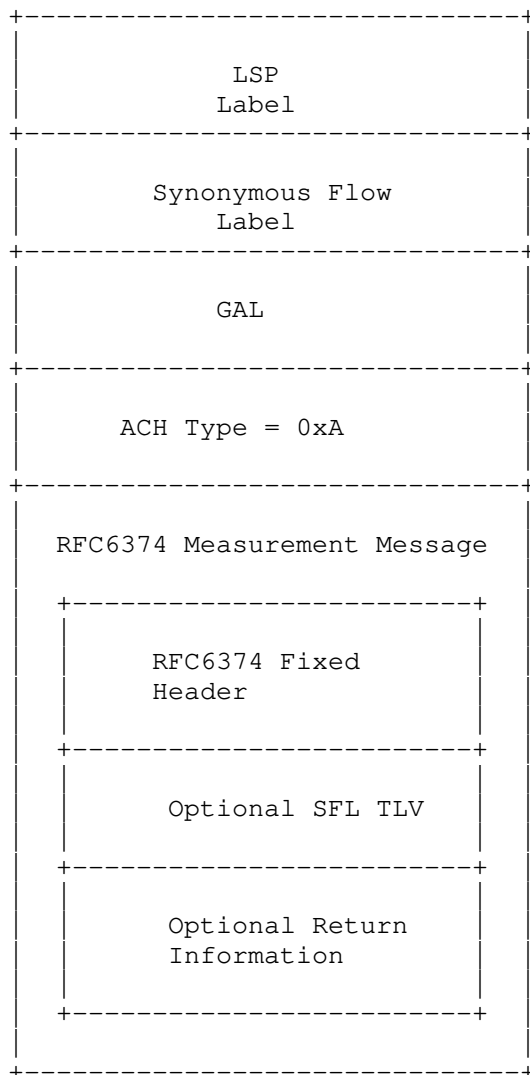   Figure 5.

```
+----------------------------+
|                            |
|            LSP             |
|           Label            |
+----------------------------+
|                            |
|     Synonymous Flow        |
|           Label            |
+----------------------------+
|                            |
|            GAL             |
|                            |
+----------------------------+
|                            |
|     ACH Type = 0xA         |
|                            |
+----------------------------+
|                            |
|  RFC6374 Measurement Message
|                            |
|    +----------------------+ |
|    |                      | |
|    |    RFC6374 Fixed     | |
|    |     Header           | |
|    |                      | |
|    +----------------------+ |
|    |                      | |
|    |   Optional SFL TLV   | |
|    |                      | |
|    +----------------------+ |
|    |                      | |
|    |  Optional Return     | |
|    |   Information        | |
|    |                      | |
|    +----------------------+ |
|                            |
+----------------------------+
```

                Figure 5: RFC6734 Query Packet with SFL

   The MPLS label stack is exactly the same as that used for the user
   data service packets being instrumented except for the inclusion of
   the GAL [RFC5586] to allow the receiver to distinguish between normal
   data packets and OAM packets.  Since the packet loss measurements are
   being made on the data service packets, an RFC6374 direct loss
   measurement is being made, and which is indicated by the type field
   in the ACH (Type = 0x000A).

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 9.1.  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
[RFC7876].  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 [RFC7876].

## 9.1.  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 6.  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.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Length    |MBZ| SFL Batch |   SFL Index   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              SFL              |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              FEC                                              |
.                                                              .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
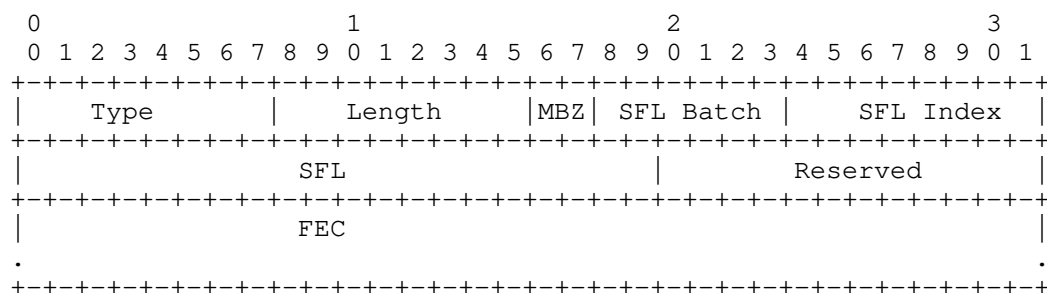
Figure 6: SFL TLV

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 [I-D.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 TBD

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.

10.  Applicability to Pro-active and On-demand Measurement

A future version of the this document will discuss the applicability
of the various methods to pro-active and on-demand Measurement.

11.  RFC6374 Combined Loss-Delay Measurement

This mode of operation is not currently supported by this
specification.

12.  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.

13.  Security Considerations

   The issue noted in Section 5 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.

14.  IANA Considerations

14.1.  Allocation of PW Associated Channel Type

   As per the IANA considerations in [RFC5586], IANA is requested to
   allocate the following Channel Type in the "PW Associated Channel
   Type" registry:

| Value | Description | TLV Follows | Reference |
| ----- | ------------------------------- | ----------- | --------- |
| TBD | RFC6374 Bucket Jitter Measurement | No | This |
| TBD | RFC6374 Multi-Packet Delay Measurement | No | This |
| TBD | RFC6374 Average Delay Measurement | No | This |

14.2.  MPLS Loss/Delay TLV Object

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

| Type | Description | Reference |
| ---- | ------------------------------- | --------- |
| TBD | Synonymous Flow Label | This |

   A value of 4 is recommended.

15.  References

15.1.  Normative References

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

   [RFC3032]  Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y.,
              Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack
              Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001,
              <http://www.rfc-editor.org/info/rfc3032>.

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

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

   [RFC7876]  Bryant, S., Sivabalan, S., and S. Soni, "UDP Return Path
              for Packet Loss and Delay Measurement for MPLS Networks",
              RFC 7876, DOI 10.17487/RFC7876, July 2016,
              <http://www.rfc-editor.org/info/rfc7876>.

15.2.  Informative References

   [I-D.bryant-mpls-sfl-control]
              Bryant, S., Swallow, G., and S. Sivabalan, "MPLS Flow
              Identification Considerations", draft-bryant-mpls-sfl-
              control-01 (work in progress), March 2017.

   [I-D.bryant-mpls-sfl-framework]
              Bryant, S., Chen, M., Li, Z., Swallow, G., Sivabalan, S.,
              and G. Mirsky, "Synonymous Flow Label Framework", draft-
              bryant-mpls-sfl-framework-04 (work in progress), April
              2017.

   [I-D.chen-ippm-coloring-based-ipfpm-framework]
              Chen, M., Zheng, L., Mirsky, G., Fioccola, G., and T.
              Mizrahi, "IP Flow Performance Measurement Framework",
              draft-chen-ippm-coloring-based-ipfpm-framework-06 (work in
              progress), March 2016.

   [I-D.ietf-ippm-alt-mark]
             Fioccola, G., Capello, A., Cociglio, M., Castaldelli, L.,
             Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi,
             "Alternate Marking method for passive performance
             monitoring", draft-ietf-ippm-alt-mark-04 (work in
             progress), March 2017.

   [I-D.ietf-mpls-flow-ident]
             Bryant, S., Pignataro, C., Chen, M., Li, Z., and G.
             Mirsky, "MPLS Flow Identification Considerations", draft-
             ietf-mpls-flow-ident-04 (work in progress), February 2017.

   [I-D.tempia-ippm-p3m]
             Capello, A., Cociglio, M., Fioccola, G., Castaldelli, L.,
             and A. Bonda, "A packet based method for passive
             performance monitoring", draft-tempia-ippm-p3m-03 (work in
             progress), March 2016.

   [RFC3270]  Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen,
             P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-
             Protocol Label Switching (MPLS) Support of Differentiated
             Services", RFC 3270, DOI 10.17487/RFC3270, May 2002,
             <http://www.rfc-editor.org/info/rfc3270>.

   [RFC5921]  Bocci, M., Ed., Bryant, S., Ed., Frost, D., Ed., Levrau,
             L., and L. Berger, "A Framework for MPLS in Transport
             Networks", RFC 5921, DOI 10.17487/RFC5921, July 2010,
             <http://www.rfc-editor.org/info/rfc5921>.

Authors' Addresses

   Stewart Bryant
   Huawei

   Email: stewart.bryant@gmail.com


   Mach Chen
   Huawei

   Email: mach.chen@huawei.com


   Zhenbin Li
   Huawei

   Email: lizhenbin@huawei.com

George Swallow
Cisco Systems

Email: swallow.ietf@gmail.com


Siva Sivabalan
Cisco Systems

Email: msiva@cisco.com


Gregory Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com


Giuseppe Fioccola
Telecom Italia

Email: giuseppe.fioccola@telecomitalia.it

CCAMP Working Group                                      Italo Busi
Internet Draft                                               Huawei
Intended status: Informational                       Sergio Belotti
Expires: January 2017                                         Nokia
                                                       Victor Lopez
                                            Oscar Gonzalez de Dios
                                                        Telefonica
                                                     Anurag Sharma
                                                          Infinera
                                                            Yan Shi
                                                      China Unicom
                                                    Ricard Vilalta
                                                              CTTC
                                                Karthik Sethuraman
                                                               NEC


                                                      July 7, 2016

                          Path Computation API
                draft-busibel-ccamp-path-computation-api-00.txt

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

This Internet-Draft will expire on January 7, 2016.

Abstract

There are scenarios, typically in a hierarchical SDN context, in
which an orchestrator may not have detailed information to be able
to perform an end-to-end path computation and would need to request
lower layer/domain controllers to calculate some (partial) feasible
paths.

Multiple protocol solutions can be used for communication between
different controller hierarchical levels. This document assumes that
the controllers are communicating using YANG-based Application
Programming Interface (APIs).

This document describes some use cases for an Application
Programming Interface for path computation. A related yang model
will be proposed in a next version or in another document.

Table of Contents

1. Introduction

   There are scenarios, typically in a hierarchical SDN context, in
   which an orchestrator may not have detailed information to be able
   to perform an end-to-end path computation and would need to request
   lower layer/domain controllers to calculate some (partial) feasible
   paths.

   Multiple protocol solutions can be used for communication between
   different controller hierarchical levels. This document assumes that
   the controllers are communicating using YANG-based Application
   Programming Interface (APIs).

   Path Computation Elements, Controllers and Orchestrators perform
   their operations based on Traffic Engineering Databases (TED). Such
   TEDs can be described, in a technology agnostic way, with the YANG
   Data Model for TE Topologies [TE-TOPO]. Furthermore, the technology
   specific details of the TED are modeled in the augmented TE topology
   models (e.g. [L1-TOPO] for Layer-1 ODU technologies).

   The availability of such topology models allows providing the TED
   via Netconf or Restconf API. Furthermore, it enables that a
   PCE/Controller performs the necessary abstractions or modifications
   and offer this customized topology to another PCE/Controller or high
   level orchestrator.

   The tunnels that can be provided over the networks described with
   the topology models can be also set-up, deleted and modified via
   Netconf or Restconf API using the TE-Tunnel Yang model [TE-TUNNEL].

   This document describes some use cases where a path computation
   function, also using Netconf or Restconf API, can be needed. A
   related yang model will be proposed in a next version or in another
   document.

2. Use Cases

   This document presents different use cases, where an API for path
   computation is required. The presented uses cases have been grouped,
   depending on the different underlying topologies: a) IP-Optical
   integration; b) Multi-domain Optical Networks; and c) Data center
   interconnections.

2.1. IP-Optical integration

   In these use cases, there is an Optical domain which is used to
   provide connectivity between IP routers which are connected with the
   Optical domains using access links (see Figure 1).


        ----------------------------------------------------------------
        I                                                              I
        I                                                              I
        I                                                              I
        I                       IP+Optical Use Cases                   I
        I                                                              I
        I                                                              I
        I                                                              I
        I                                                              I
        I                       (only in PDF version)                  I
        I                                                              I
        I                                                              I
        I                                                              I
        I                                                              I
        I                                                              I
        ----------------------------------------------------------------
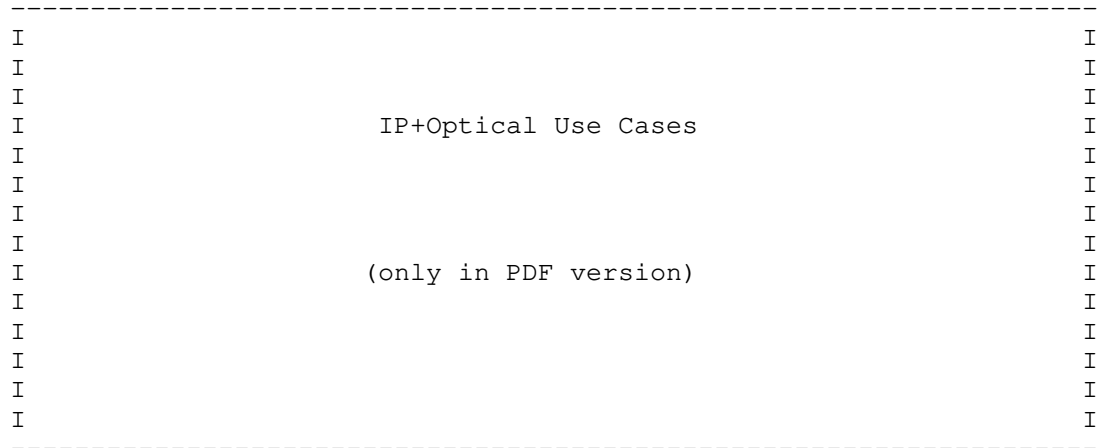

                     Figure 1 - IP+Optical Use Cases

   It is assumed that the Optical domain controller provides to the
   orchestrator an abstracted view of the Optical network. A possible
   abstraction shall be representing the optical domain as one "virtual
   node" with "virtual ports" connected to the access links.

   The path computation request helps the orchestrator to know which
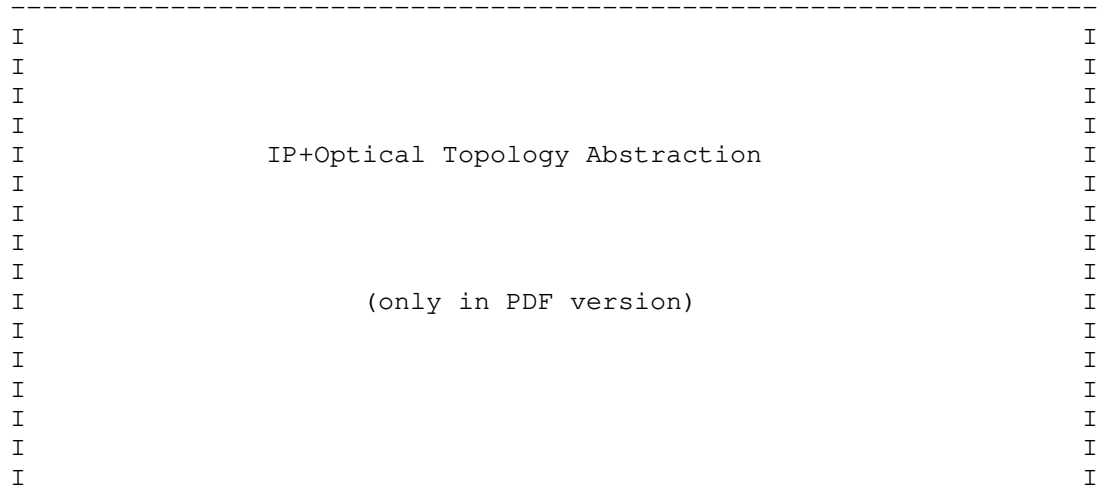   are the real connections that can be provided at the optical domain.

```
    -------------------------------------------------------------------
    I                                                                 I
    I                                                                 I
    I                                                                 I
    I                                                                 I
    I                    IP+Optical Topology Abstraction              I
    I                                                                 I
    I                                                                 I
    I                                                                 I
    I                                                                 I
    I                        (only in PDF version)                    I
    I                                                                 I
    I                                                                 I
    I                                                                 I
    I                                                                 I
    I                                                                 I
    I                                                                 I
    -------------------------------------------------------------------
```

            Figure 2 - IP+Optical Topology Abstraction

2.1.1. Inter-layer path computation

   In this use case the orchestrator needs to setup an optimal path
   between two IP routers R1 and R2.

   As depicted in Figure 2, the Orchestrator has only an "abstracted
   view" of the physical network, and it does not know the feasibility
   or the cost of the possible optical paths (e.g., VP1-VP4 and VP2-
   VP5), which depend from the current status of the physical resources
   within the optical network and on vendor-specific optical
   attributes.

   However, the orchestrator can ask the underlying Optical domain
   controller to compute a set of potential optimal paths, taking into
   account optical constraints Then, based on its own constraints,
   policy and knowledge (e.g. cost of the access links), it can choose

which one of these potential paths to use to setup the optimal e2e
path crossing optical network.

```
     -------------------------------------------------------------------
     I                                                                 I
     I               IP+Optical Path Computation Example               I
     I                                                                 I
     I                                                                 I
     I                      (only in PDF version)                      I
     I                                                                 I
     -------------------------------------------------------------------
```

Figure 3 - IP+Optical Path Computation Example

For example, in Figure 3, the Orchestrator can request the Optical
domain controller to compute the paths between VP1-VP4 and VP2-VP5
and then decide to setup the optimal end-to-end path which passes
through the VP2-VP5 Optical path even this is not the optimal path
from the Optical domain perspective.

An alternative approach could be to have the Optical domain
controller making the information shown in Figure 3 available to the
Orchestrator.

One possibility, under discussion within the TEAS WG, is to provide
a "detailed connectivity matrix" which extends the "connectivity
matrix" defined in [RFC7446] and describes not only the valid
inbound-outbound TE link switching combinations, but also specifies
a vector of various costs (in terms of delay, OSNR, intra-node SRLGs
and summary TE metrics) a potential TE path associated with the
connectivity matrix entry.

The information provided by the "detailed abstract connectivity
matrix" would be equivalent to the information that should be
provided by "virtual link model" as defined in [TE-INTERCONNECT].

In this case, the Path Computation Element (PCE) within the
Orchestrator could use this information to calculate by its own the
optimal path between routers R1 and R2, without requesting any
additional information to the Optical Domain Controller.

However, there is a tradeoff between the accuracy (i.e., providing "all" the information that might be needed by the Orchestrator's PCE) and scalability to be considered when designing the amount of information to provide within the "detailed abstract connectivity matrix".

Figure 4 below shows another example, similar to the one in Figure 3, but where there are two possible Optical paths between VP1 and VP4 with different properties (e.g., available bandwidth and cost).

```
  -------------------------------------------------------------------
  I                                                                 I
  I               IP+Optical Path Computation Example              I
  I                     with multiple choices                      I
  I                                                                 I
  I                                                                 I
  I                       (only in PDF version)                    I
  I                                                                 I
  -------------------------------------------------------------------
```

 Figure 4 - IP+Optical Path Computation Example with multiple choices

Reporting all the information, as in Figure 4, using the "detailed abstract connectivity matrix" is quite challenging from a scalability perspective since the amount of this information is not just based on number of end points (which would scale as N-square), but also on many other parameters, including client rate, user constraints / policies for the service, e.g. max latency < N ms, max cost, etc., exclusion policies to route around busy links, min OSNR margin, max preFEC BER etc. All these constraints could be different based on connectivity requirements.

It is also worth noting that the "connectivity matrix" has been originally defined in WSON, [RFC7446] to report the connectivity constrains of a physical node within the WDM network: the information it contains is pretty "static" and therefore, once taken and stored in the TE data base, it can be always being considered valid and up-to-date in path computation request.

Using the "connectivity matrix" with an abstract node to abstract the information regarding the connectivity constraints of an Optical domain, would make this information more "dynamic" since the

connectivity constraints of an Optical domain can change over time because some optical paths that are feasible at a given time may become unfeasible at a later time when e.g., another optical path is established. The information in the "detailed abstract connectivity matrix" is even more dynamic since the establishment of another optical path may change some of the parameters (e.g., delay or available bandwidth) in the "detailed abstract connectivity matrix" while not changing the feasibility of the path.

"Connectivity matrix" is sometimes confused with optical reach table that contain multiple (e.g. k-shortest) regen-free reachable paths for every A-Z node combination in the network. Optical reach tables can be calculated offline, utilizing vendor optical design and planning tools,and periodically uploaded to the Controller: these optical path reach tables are fairly static. However, to get the connectivity matrix, between any two sites, either a regen free path can be used, if one is available, or multiple regen free paths are concatenated to get from src to dest, which can be a very large combination. Additionally, when the optical path within optical domain needs to be computed, it can result in different paths based on input objective, constraints, and network conditions. In summary, even though "optical reachability table" is fairly static, which regen free paths to build the connectivity matrix between any source and destination  is very dynamic, and is done using very sophisticated routing algorithms.

There is therefore the need to keep the information in the "connectivity matrix" updated which means that there another tradeoff between the accuracy (i.e., providing "all" the information that might be needed by the Orchestrator's PCE) and having up-to-date information. The more the information is provided and the longer it takes to keep it up-to-date which increases the likelihood that the Orchestrator's PCE computes paths using not updated information.

It seems therefore quite challenging to have a "detailed abstract connectivity matrix" that provides accurate, scalable and updated information to allow the Orchestrator's PCE to take optimal decisions by its own.

If the information in the "detailed abstract connectivity matrix" is not complete/accurate, we can have the following drawbacks considering for example the case in Figure 4:

   o  If only the VP1-VP4 path with available bandwidth of 2 Gb/s and
      cost 50 is reported, the Orchestrator's PCE will fail to compute
      a 5 Gb/s path between routers R1 and R2, although this would be
      feasible;

   o  If only the VP1-VP4 path with available bandwidth of 10 Gb/s and
      cost 60 is reported, the Orchestrator's PCE will compute, as
      optimal, the 1 Gb/s path between R1 and R2 going through the VP2-
      VP5 path within the Optical domain while the optimal path would
      actually be the one going thought the VP1-VP4 sub-path (with cost
      50) within the Optical domain.

   Instead, using the approach proposed in this document, the
   Orchestrator, when it needs to setup an end-to-end path, it can
   request the Optical domain controller to compute a set of optimal
   paths (e.g., for VP1-VP4 and VP2-VP5) and take decisions based on
   the information received:

   o  When setting up a 5 Gb/s path between routers R1 and R2, the
      Optical domain controller may report only the VP1-VP4 path as the
      only feasible path: the Orchestrator can successfully setup the
      end-to-end path passing though this Optical path;

   o  When setting up a 1 Gb/s path between routers R1 and R2, the
      Optical domain controller (knowing that the path requires only 1
      Gb/s) can report both the VP1-VP4 path, with cost 50, and the
      VP2-VP5 path, with cost 65. The Orchestrator can then compute the
      optimal path which is passing thought the VP1-VP4 sub-path (with
      cost 50) within the Optical domain.

   Considering the dynamicity of the connectivity constraints of an
   Optical domain, it is possible that a path computed by the Optical
   domain controller when requested by the Orchestrator is no longer
   valid when the Orchestrator requests it to be setup up.

   It is worth noting that with the approach proposed in this document,
   the likelihood for this issue to happen can be quite small since the
   time window between the path computation request and the path setup
   request should be quite short (especially if compared with the time
   that would be needed to update the information of a very detailed
   abstract connectivity matrix).

   If this risk is still not acceptable, the Orchestrator may also
   optionally request the Optical domain controller not only to compute
   the path but also to keep track of its resources (e.g., these

resources can be reserved to avoid being used by any other connection). In this case, some mechanism (e.g., a timeout) needs to be defined to avoid having stranded resources within the Optical domain.

These issues and solutions can be fine-tuned during the design of the Path Computation API.

2.1.2. Route Diverse IP Services

This is for further study.

2.2. Multi-domain Optical Networks

In this use case there are two optical domains which are interconnected together by multiple inter-domains links.

```
 --------------------------------------------------------------------
I                                                                    I
I                                                                    I
I                                                                    I
I                                                                    I
I                                                                    I
I                                                                    I
I             Multi-domain multi-link interconnection               I
I                                                                    I
I                                                                    I
I                                                                    I
I                                                                    I
I                        (only in PDF version)                       I
I                                                                    I
I                                                                    I
I                                                                    I
I                                                                    I
I                                                                    I
 --------------------------------------------------------------------
```
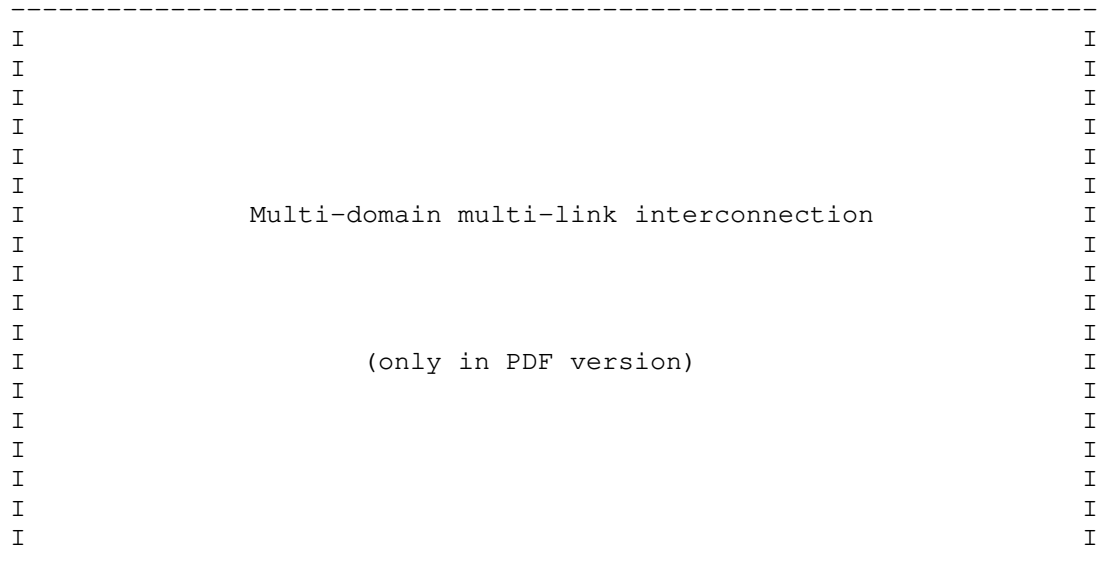
Figure 5 - Multi-domain multi-link interconnection

In order to setup an end-to-end multi-domain Optical path (e.g., between nodes A and H), the orchestrator needs to know the feasibility or the cost of the possible optical paths within the two

optical domains, which depend from the current status of the
physical resources within each optical network and on vendor-
specific optical attributes (which may be different in the two
domains if they are provided by different vendors).

There is a trade-off between having the Orchestrator's PCE being
able to take path computation decisions by its own versus having the
Orchestrator being able to ask the Domain Controllers to provide a
set of feasible optimal optical paths.

Orchestrator could want to select/optimize end-to-end path based on
abstract topology information provided by the domain controllers.
For example:

o  Need to compute a path between A and H

o  That path can go through inter-domain link C-E or through inter-
   domain link D-F

o  Orchestrator's PCE, based on its own information, can compute the
   optimal multi-domain path being A-B-C-E-G-H

o  But, during path setup, the domain controller may find out that
   A-B-C is not optically feasible, while only the path A-B-D is
   feasible

o  So what the hierarchical controller computed is not good and need
   to re-start the path computation from scratch

As discussed in section 3.1, providing more extensive abstract
information from the Optical domain controllers to the multi-domain
Orchestator may lead to scalability problems.

Alternatively the Orchestrator can request the Optical domain
controllers to compute a set of optimal paths and take decisions
based on the information received. For example:

o  Need to compute a path between A and H

o  The Orchestrator asks Optical domain controllers to provide set
   of paths between A-C, A-D, E-H and F-H

o  Optical domain controllers return a set of feasible paths with
   the associated costs: the path A-C would not be part of this set

   o  The Orchestrator will select the path A-B-D-F-G-H since it is the
      only feasible path and then request the Optical domain
      controllers to setup the A-B-D and F-G-H paths

   o  If there are multiple feasible paths, the Orchestrator can select
      the optimal path knowing the cost of the intra-domain paths
      (provided by the Optical domain controllers) and the cost of the
      inter-domain links (known by the Orchestrator)

In a sense this is similar to the problem of routing and wavelength
assignment within an Optical domain. It is possible to do first
routing (step 1) and then wavelength assignment (step 2), but the
chances of ending up with a good path is low. Alternatively, it is
possible to do combined routing and wavelength assignment, which is
known to be a more optimal and effective way for Optical path setup.
Similarly, it is possible to first compute an abstract end-to-end
path within the multi-domain Orchestrator (step 1) and then compute
an intra-domain path within each Optical domain (step 2), but there
are more chances not to find a path or to get a suboptimal path that
performing per-domain path computation and then stitch them.

The approach to request each Optical domain controllers to compute a
set of optimal paths and take decisions based on the information
received may still have some scalability issues when the number of
Optical domains is quite big (e.g. 20).

In this case, it would be worthwhile combining the two approaches and
use the abstract topology information provided by the domain
controllers to limit the number of potential optimal end-to-end paths
and then the Path Computation to decide what is the optimal path
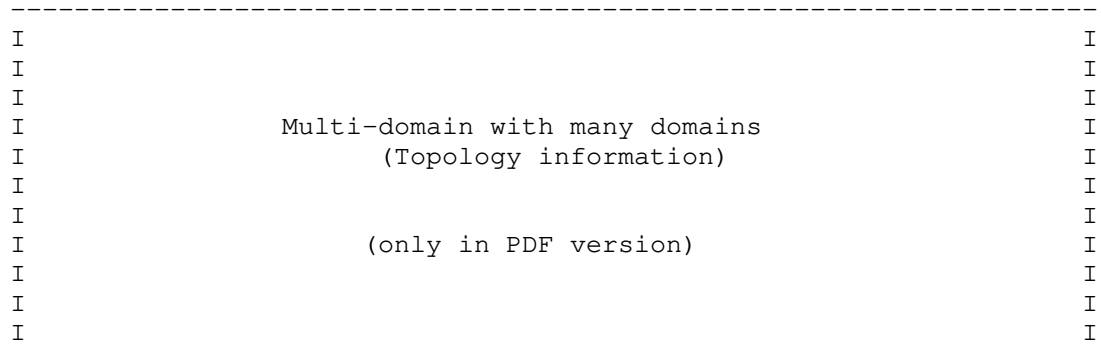within this limited set.

```
     ----------------------------------------------------------------
     I                                                              I
     I                                                              I
     I                                                              I
     I              Multi-domain with many domains                  I
     I                   (Topology information)                     I
     I                                                              I
     I                                                              I
     I                    (only in PDF version)                     I
     I                                                              I
     I                                                              I
     I                                                              I
     ----------------------------------------------------------------
```

   Figure 6 - Multi-domain with many domains (Topology information)

An example can be described considering multi-domain abstract
topology shown in Figure 6. In this example an end-to-end Optical
path between domains A and F needs to be setup. The transit domain
should be selected between domains B, C, D and E.

The actual cost of each intra-domain path is not known a priori from
the abstract topology information. The Orchestrator only knows the
feasibility of some intra-domain paths and some upper-bound and/or
lower-bound cost information. With this information, together with
the cost of inter-domain links, the Orchestrator can decide that:

o  Domain B cannot be selected as the path connecting domains A and
   E is not feasible;

o  Domain E cannot be selected as a transit domain since it is know
   from the abstract topology information provided by domain
   controllers that the cost of the multi-domain path A-E-F (which
   is 100, in the best case) will be always be higher than the cost
   of the multi-domain paths A-D-F (which is 90, in the worst case)
   and A-E-F (which is 80, in the worst case)

Therefore, the Orchestrator can decide by its own that the optimal
multi-domain path could be either A-D-F or A-E-F.

The Orchestrator can therefore request only the Optical domain
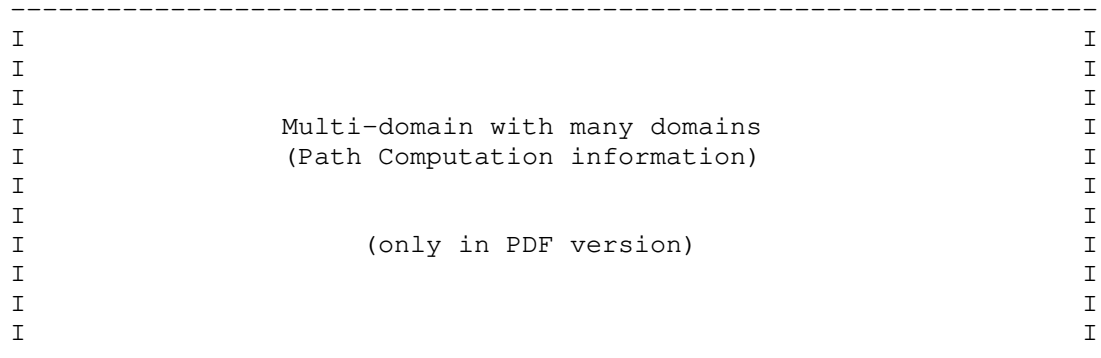controllers A, D, E and F to provide a set of optimal paths.

```
      -----------------------------------------------------------------
      I                                                               I
      I                                                               I
      I                                                               I
      I                 Multi-domain with many domains                I
      I                 (Path Computation information)                I
      I                                                               I
      I                                                               I
      I                    (only in PDF version)                      I
      I                                                               I
      I                                                               I
      I                                                               I
      -----------------------------------------------------------------
```

        Figure 7 – Multi-domain with many domains (Path Computation
                                information)

   Based on these requests, the Orchestrator can know the actual cost
   of each intra-domain paths which belongs to potential optimal end-
   to-end paths, as shown in Figure 7, and then compute the optimal
   end-to-end path (e.g., A-D-F, having total cost of 50, instead of A-
   C-F having a total cost of 70).

2.3. Data center interconnections

   In these use case, there is an Optical domain which is used to
   provide connectivity between data centers which are connected with
   the Optical domains using access links.

```
-------------------------------------------------------------------
I                                                                 I
I                                                                 I
I                                                                 I
I                                                                 I
I                                                                 I
I                                                                 I
I                  Data Center Interconnection Use Case           I
I                                                                 I
I                                                                 I
I                                                                 I
I                      (only in PDF version)                      I
I                                                                 I
I                                                                 I
I                                                                 I
I                                                                 I
I                                                                 I
I                                                                 I
-------------------------------------------------------------------
```
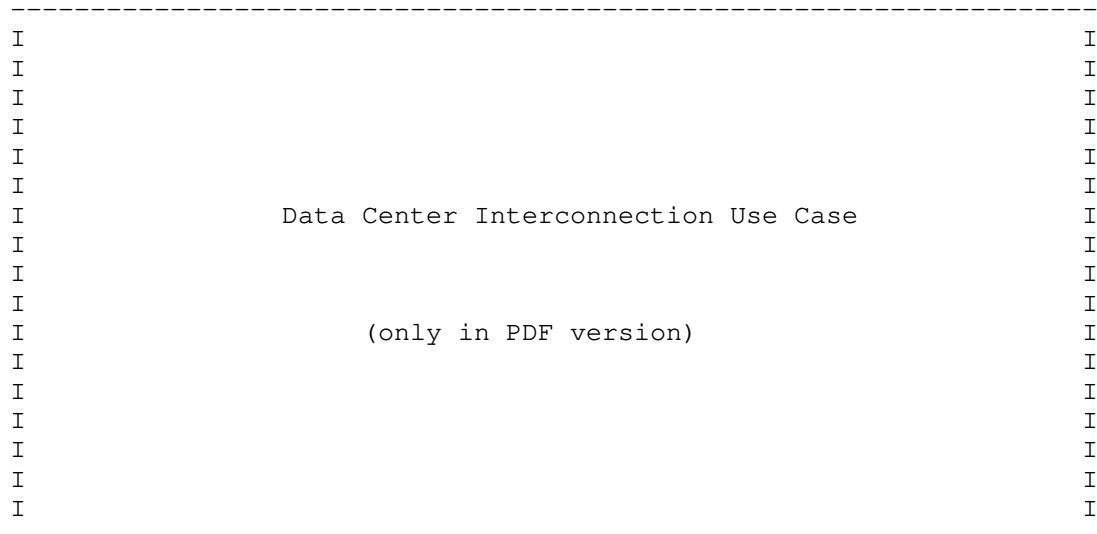
Figure 8 - Data Center Interconnection Use Case

In this use case, a virtual machine within Data Center 1 (DC1) needs
to transfer data to another virtual machine that can reside either
in DC2 or in DC3.

The optimal decision depends both on the cost of the optical path
(DC1-DC2 or DC1-DC3) and of the computing power (data center
resources) within DC2 or DC3.

The Cloud Orchestrator may not be able to make this decision because
it has only an abstract view of the optical network (as in use case
in 3.1).

The cloud orchestrator can request to the Optical domain controller
to compute the cost of the possible optical paths (e.g., DC1-DC2 and
DC1-DC3) and to the DC controller to compute the cost of the
computing power (DC resources) within DC2 and DC3 and then it can
take the decision about the optimal solution based on this
information and its policy.

3. Security Considerations

    This is for further study

4. IANA Considerations

    This document requires no IANA actions.

5. References

5.1. Normative References

    [RFC7446] Lee, Y. et al., "Routing and Wavelength Assignment
              Information Model for Wavelength Switched Optical
              Networks", RFC 7446, February 2015.

5.2. Informative References

    [TE-TOPO] Liu, X. et al., "YANG Data Model for TE Topologies",
              draft-ietf-teas-yang-te-topo, work in progress.

    [L1-TOPO] Zhang, X. et al., "A YANG Data Model for Layer 1 (ODU)
              Network Topology", draft-zhang-ccamp-l1-topo-yang, work in
              progress.

    [TE-TUNNEL]   Xhang, X. et al., "A YANG Data Model for Traffic
              Engineering Tunnels and Interfaces", draft-ietf-teas-yang-
              te, work in progress.

    [TE-INTERCONNECT]   Farrel, A. et al., "Problem Statement and
              Architecture for Information Exchange Between
              Interconnected Traffic Engineered Networks", draft-ietf-
              teas-interconnected-te-info-exchange, work in progress.

6. Acknowledgments

    The authors would like to thank Igor Bryskin and Xian Zhang for
    participating in discussions and providing valuable insights.

    This document was prepared using 2-Word-v2.0.template.dot.

Contributors

   Dieter Beller
   Nokia
   Email: dieter.beller@nokia.com


Authors' Addresses

   Italo Busi
   Huawei
   Email: italo.busi@huawei.com


   Sergio Belotti
   Nokia
   Email: sergio.belotti@nokia.com


   Victor Lopez
   Telefonica
   Email: victor.lopezalvarez@telefonica.com


   Oscar Gonzalez de Dios
   Telefonica
   Email: oscar.gonzalezdedios@telefonica.com


   Anurag Sharma
   Infinera
   Email: AnSharma@infinera.com


   Yan Shi
   China Unicom
   Email: shiyan49@chinaunicom.cn


   Ricard Vilalta
   CTTC
   Email: ricard.vilalta@cttc.es

   Karthik Sethuraman
   NEC
   Email: karthik.sethuraman@necam.com

TEAS Working Group                                          Italo Busi
Internet Draft                                                  Huawei
Intended status: Informational                         Sergio Belotti
Expires: April 2017                                              Nokia
                                                          Victor Lopez
                                              Oscar Gonzalez de Dios
                                                           Telefonica
                                                         Anurag Sharma
                                                             Infinera
                                                               Yan Shi
                                                          China Unicom
                                                        Ricard Vilalta
                                                                  CTTC
                                                    Karthik Sethuraman
                                                                   NEC


                                                      October 28, 2016

                 Yang model for requesting Path Computation
               draft-busibel-teas-yang-path-computation-00.txt

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

This Internet-Draft will expire on April 28, 2016.

Abstract

   There are scenarios, typically in a hierarchical SDN context, in
   which an orchestrator may not have detailed information to be able
   to perform an end-to-end path computation and would need to request
   lower layer/domain controllers to calculate some (partial) feasible
   paths.

   Multiple protocol solutions can be used for communication between
   different controller hierarchical levels. This document assumes that
   the controllers are communicating using YANG-based protocols (e.g.,
   NETCONF or RESTCONF).

   This document describes some use cases for a YANG model to request
   path computation.

Table of Contents

1. Introduction

   There are scenarios, typically in a hierarchical SDN context, in
   which an orchestrator may not have detailed information to be able
   to perform an end-to-end path computation and would need to request
   lower layer/domain controllers to calculate some (partial) feasible
   paths.

   When we are thinking to this type of scenarios we have in mind
   specific level of interfaces on which this request can be applied.

   We can reference ABNO Control Interface [RFC7491] in which an
   Application Service Coordinator can request ABNO controller to take
   in charge path calculation (see Figure 1 in the RFC) and/or ACTN
   [ACTN-frame],where controller hierarchy is defined, the need for
   path computation arises on both interfaces CMI (interface between
   Customer Network Controller(CNC) and Multi Domain Service
   Coordinator (MDSC)) and/or MPI (interface between MSDC-PNC).[ACTN-
   Info] describes an information model for the Path Computation
   request.

   Multiple protocol solutions can be used for communication between
   different controller hierarchical levels. This document assumes that
   the controllers are communicating using YANG-based protocols (e.g.,
   NETCONF or RESTCONF).

Path Computation Elements, Controllers and Orchestrators perform their operations based on Traffic Engineering Databases (TED). Such TEDs can be described, in a technology agnostic way, with the YANG Data Model for TE Topologies [TE-TOPO]. Furthermore, the technology specific details of the TED are modeled in the augmented TE topology models (e.g. [L1-TOPO] for Layer-1 ODU technologies).

The availability of such topology models allows providing the TED using YANG-based protocols (e.g., NETCONF or RESTCONF). Furthermore, it enables a PCE/Controller performing the necessary abstractions or modifications and offering this customized topology to another PCE/Controller or high level orchestrator.

The tunnels that can be provided over the networks described with the topology models can be also set-up, deleted and modified via YANG-based protocols (e.g., NETCONF or RESTCONF)using the TE-Tunnel Yang model [TE-TUNNEL].

This document describes some use cases where a path computation request, via YANG-based protocols (e.g., NETCONF or RESTCONF), can be needed.

2. Use Cases

This section presents different use cases, where an orchestrator needs to request underlying SDN controllers for path computation.

The presented uses cases have been grouped, depending on the different underlying topologies: a) IP-Optical integration; b) Multi-domain Traffic Engineered (TE) Networks; and c) Data center interconnections.

2.1. IP-Optical integration

In these use cases, an Optical domain is used to provide connectivity between IP routers which are connected with the Optical domains using access links (see Figure 1).
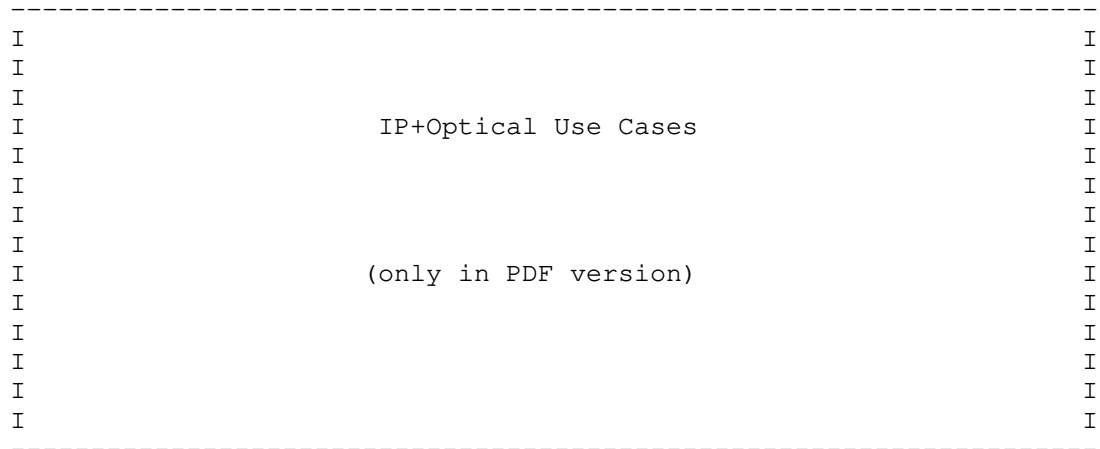
```
       -------------------------------------------------------------------
       I                                                                 I
       I                                                                 I
       I                                                                 I
       I                      IP+Optical Use Cases                       I
       I                                                                 I
       I                                                                 I
       I                                                                 I
       I                                                                 I
       I                      (only in PDF version)                      I
       I                                                                 I
       I                                                                 I
       I                                                                 I
       I                                                                 I
       I                                                                 I
       -------------------------------------------------------------------
```

                     Figure 1 - IP+Optical Use Cases

   It is assumed that the Optical domain controller provides to the
   orchestrator an abstracted view of the Optical network. A possible
   abstraction shall be representing the optical domain as one "virtual
   node" with "virtual ports" connected to the access links.

   The path computation request helps the orchestrator to know which
   are the real connections that can be provided at the optical domain.

```
        ------------------------------------------------------------------
        I                                                                I
        I                                                                I
        I                                                                I
        I                                                                I
        I                                                                I
        I                 IP+Optical Topology Abstraction                I
        I                                                                I
        I                                                                I
        I                                                                I
        I                                                                I
        I                    (only in PDF version)                       I
        I                                                                I
        I                                                                I
        I                                                                I
        I                                                                I
        I                                                                I
        I                                                                I
        ------------------------------------------------------------------
```

Figure 2 - IP+Optical Topology Abstraction

2.1.1. Inter-layer path computation

   In this use case, the orchestrator needs to setup an optimal path
   between two IP routers R1 and R2.

   As depicted in Figure 2, the Orchestrator has only an "abstracted
   view" of the physical network, and it does not know the feasibility
   or the cost of the possible optical paths (e.g., VP1-VP4 and VP2-
   VP5), which depend from the current status of the physical resources
   within the optical network and on vendor-specific optical
   attributes.

   The orchestrator can request the underlying Optical domain
   controller to compute a set of potential optimal paths, taking into
   account optical constraints. Then, based on its own constraints,
   policy and knowledge (e.g. cost of the access links), it can choose

which one of these potential paths to use to setup the optimal e2e
path crossing optical network.

```
---------------------------------------------------------------------
I                                                                   I
I                  IP+Optical Path Computation Example              I
I                                                                   I
I                                                                   I
I                       (only in PDF version)                       I
I                                                                   I
---------------------------------------------------------------------
```

Figure 3 - IP+Optical Path Computation Example

For example, in Figure 3, the Orchestrator can request the Optical
domain controller to compute the paths between VP1-VP4 and VP2-VP5
and then decide to setup the optimal end-to-end path using the VP2-
VP5 Optical path even this is not the optimal path from the Optical
domain perspective.

Considering the dynamicity of the connectivity constraints of an
Optical domain, it is possible that a path computed by the Optical
domain controller when requested by the Orchestrator is no longer
valid when the Orchestrator requests it to be setup up.

It is worth noting that with the approach proposed in this document,
the likelihood for this issue to happen can be quite small since the
time window between the path computation request and the path setup
request should be quite short (especially if compared with the time
that would be needed to update the information of a very detailed
abstract connectivity matrix).

If this risk is still not acceptable, the Orchestrator may also
optionally request the Optical domain controller not only to compute
the path but also to keep track of its resources (e.g., these
resources can be reserved to avoid being used by any other
connection). In this case, some mechanism (e.g., a timeout) needs to
be defined to avoid having stranded resources within the Optical
domain.

These issues and solutions can be fine-tuned during the design of
the YANG model for requesting Path Computation.

2.1.2. Route Diverse IP Services

    This is for further study.

2.2. Multi-domain TE Networks

    In this use case there are two TE domains which are interconnected
    together by multiple inter-domains links.

    A possible example could be a multi-domain optical network.

```
    ---------------------------------------------------------------------
    I                                                                   I
    I                                                                   I
    I                                                                   I
    I                                                                   I
    I                                                                   I
    I                                                                   I
    I              Multi-domain multi-link interconnection              I
    I                                                                   I
    I                                                                   I
    I                                                                   I
    I                                                                   I
    I                      (only in PDF version)                        I
    I                                                                   I
    I                                                                   I
    I                                                                   I
    I                                                                   I
    I                                                                   I
    I                                                                   I
    ---------------------------------------------------------------------
```
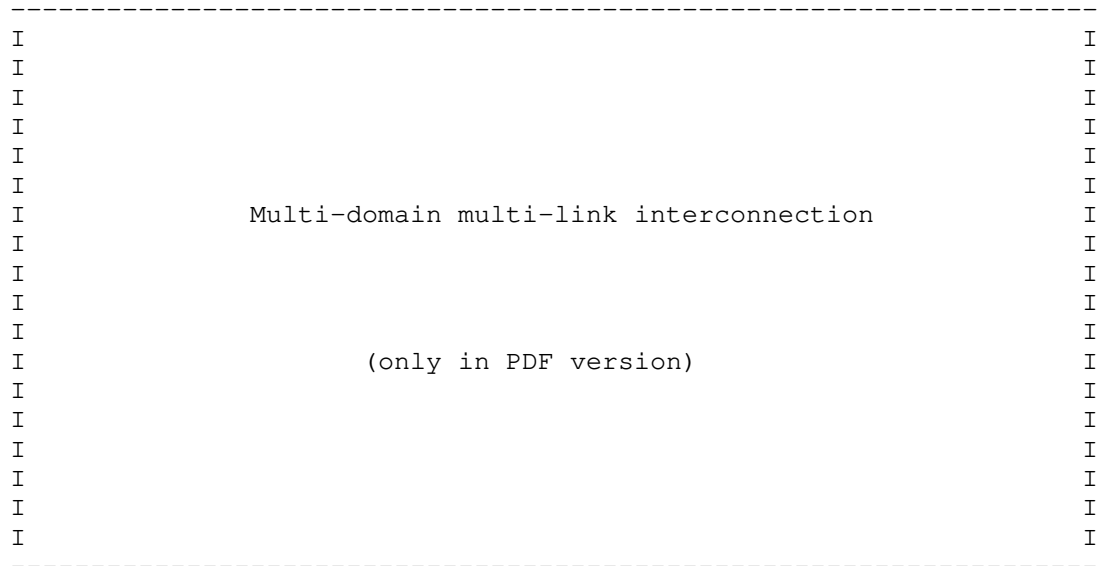
        Figure 4 - Multi-domain multi-link interconnection

    In order to setup an end-to-end multi-domain TEpath (e.g., between
    nodes A and H), the orchestrator needs to know the feasibility or
    the cost of the possible TE paths within the two TE domains, which
    depend from the current status of the physical resources within each
    TE network. This is more challenging in case of optical networks
    because the optimal paths depend also on vendor-specific optical
    attributes (which may be different in the two domains if they are
    provided by different vendors).

In order to setup a multi-domain TE path (e.g., between nodes A and H), Orchestrator can request the TE domain controllers to compute a set of intra-domain optimal paths and take decisions based on the information received. For example:

o  The Orchestrator asks TE domain controllers to provide set of paths between A-C, A-D, E-H and F-H

o  TE domain controllers return a set of feasible paths with the associated costs: the path A-C is not part of this set(in optical networks, it is typical to have some paths not being feasible due to optical constraints that are known only by the optical domain controller)

o  The Orchestrator will select the path A- D-F- H since it is the only feasible multi-domain path and then request the TE domain controllers to setup the A-D and F-H intra-domain paths

o  If there are multiple feasible paths, the Orchestrator can select the optimal path knowing the cost of the intra-domain paths (provided by the TE domain controllers) and the cost of the inter-domain links (known by the Orchestrator)

This approach may  have some scalability issues when the number of TE domains is quite big (e.g. 20).

In this case, it would be worthwhile using the abstract TE topology information provided by the domain controllers to limit the number of potential optimal end-to-end paths and then request path computation to fewer domain controllers in order to decide what the optimal path within this limited set is.

For more details, see section 3.3.

2.3. Data center interconnections

In these use case, there is an TE domain which is used to provide connectivity between data centers which are connected with the TE domain using access links.

```
      ------------------------------------------------------------------
      I                                                                I
      I                                                                I
      I                                                                I
      I                                                                I
      I                                                                I
      I                                                                I
      I                Data Center Interconnection Use Case            I
      I                                                                I
      I                                                                I
      I                                                                I
      I                     (only in PDF version)                      I
      I                                                                I
      I                                                                I
      I                                                                I
      I                                                                I
      I                                                                I
      I                                                                I
      ------------------------------------------------------------------
```
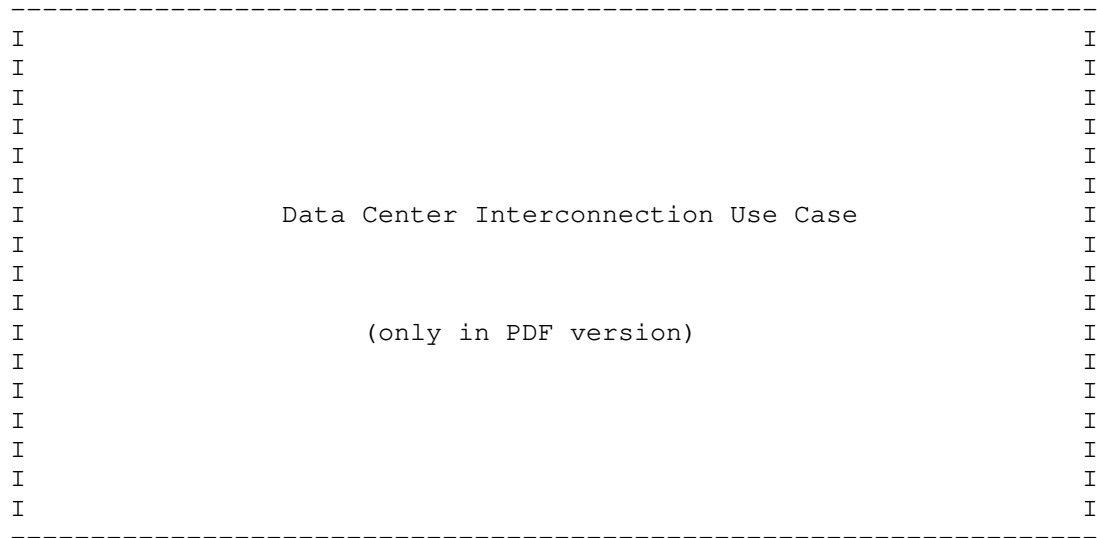
              Figure 5 - Data Center Interconnection Use Case

   In this use case, a virtual machine within Data Center 1 (DC1) needs
   to transfer data to another virtual machine that can reside either
   in DC2 or in DC3.

   The optimal decision depends both on the cost of the TE path (DC1-
   DC2 or DC1-DC3) and of the computing power (data center resources)
   within DC2 or DC3.

   The Cloud Orchestrator may not be able to make this decision because
   it has only an abstract view of the TE network (as in use case in
   2.1).

   The cloud orchestrator can request to the TE domain controller to
   compute the cost of the possible TE paths (e.g., DC1-DC2 and DC1-
   DC3) and to the DC controller to compute the cost of the computing
   power (DC resources) within DC2 and DC3 and then it can take the
   decision about the optimal solution based on this information and
   its policy.

3. Interactions with TE Topology

   The use cases described in section 2 have been described assuming
   that the topology view exported by each underlying SDN controller to
   the orchestrator is aggregated using the "virtual node model",
   defined in [RFC7926].

   TE Topology information, e.g., as provided by [TE-TOPO], could in
   theory be used by an underlying SDN controllers to provide TE
   information to the orchestrator thus allowing the Path Computation
   Element (PCE) within the Orchestrator to perform multi-domain path
   computation by its own, without requesting path computations to the
   underlying SDN controllers.

   This section analyzes the need for an orchestrator to request
   underlying SDN controllers for path computation even in these
   scenarios as well as how the TE Topology information and the path
   computation can be complementary.

   In nutshell, there is a scalability trade-off between providing all
   the TE information needed by the Orchestrator's PCE to take optimal
   path computation decisions by its own versus requesting the
   Orchestrator to ask to too many underlying SDN Domain Controllers a
   set of feasible optimal intra-domain TE paths.

3.1. TE Topology Aggregation using the "virtual link model"

   Using the TE Topology model, as defined in [TE-TOPO], the underlying
   SDN controller can export the whole TE domain as a single abstract
   TE node with a "detailed connectivity matrix", which extends the
   "connectivity matrix", defined in [RFC7446], with specific TE
   attributes (e.g., delay, SRLGs and summary TE metrics).

   The information provided by the "detailed abstract connectivity
   matrix" would be equivalent to the information that should be
   provided by "virtual link model" as defined in [RFC 7926].

   For example, in the IP-Optical integration use case, described in
   section 2.1, the Optical domain controller can make the information
   shown in Figure 3 available to the Orchestrator as part of the TE
   Topology information and the Orchestrator could use this information
   to calculate by its own the optimal path between routers R1 and R2,
   without requesting any additional information to the Optical Domain
   Controller.

However, there is a tradeoff between the accuracy (i.e., providing "all" the information that might be needed by the Orchestrator's PCE) and scalability to be considered when designing the amount of information to provide within the "detailed abstract connectivity matrix".

Figure 6 below shows another example, similar to Figure 3, where there are two possible Optical paths between VP1 and VP4 with different properties (e.g., available bandwidth and cost).

```
   -------------------------------------------------------------------
   I                                                                 I
   I              IP+Optical Path Computation Example                I
   I                      with multiple choices                      I
   I                                                                 I
   I                                                                 I
   I                       (only in PDF version)                     I
   I                                                                 I
   -------------------------------------------------------------------
```

 Figure 6 - IP+Optical Path Computation Example with multiple choices

 Reporting all the information, as in Figure 6, using the "detailed abstract connectivity matrix", is quite challenging from a scalability perspective. The amount of this information is not just based on number of end points (which would scale as N-square), but also on many other parameters, including client rate, user constraints / policies for the service, e.g. max latency < N ms, max cost, etc., exclusion policies to route around busy links, min OSNR margin, max preFEC BER etc. All these constraints could be different based on connectivity requirements.

 It is also worth noting that the "connectivity matrix" has been originally defined in WSON, [RFC7446] to report the connectivity constrains of a physical node within the WDM network: the information it contains is pretty "static" and therefore, once taken and stored in the TE data base, it can be always being considered valid and up-to-date in path computation request.

 Using the "connectivity matrix" with an abstract node to abstract the information regarding the connectivity constraints of an Optical domain, would make this information more "dynamic" since the

connectivity constraints of an Optical domain can change over time
because some optical paths that are feasible at a given time may
become unfeasible at a later time when e.g., another optical path is
established. The information in the "detailed abstract connectivity
matrix" is even more dynamic since the establishment of another
optical path may change some of the parameters (e.g., delay or
available bandwidth) in the "detailed abstract connectivity matrix"
while not changing the feasibility of the path.

"Connectivity matrix" is sometimes confused with optical reach table
that contain multiple (e.g. k-shortest) regen-free reachable paths
for every A-Z node combination in the network. Optical reach tables
can be calculated offline, utilizing vendor optical design and
planning tools,and periodically uploaded to the Controller: these
optical path reach tables are fairly static. However, to get the
connectivity matrix, between any two sites, either a regen free path
can be used, if one is available, or multiple regen free paths are
concatenated to get from src to dest, which can be a very large
combination. Additionally, when the optical path within optical
domain needs to be computed, it can result in different paths based
on input objective, constraints, and network conditions. In summary,
even though "optical reachability table" is fairly static, which
regen free paths to build the connectivity matrix between any source
and destination  is very dynamic, and is done using very
sophisticated routing algorithms.

There is therefore the need to keep the information in the
"connectivity matrix" updated which means that there another
tradeoff between the accuracy (i.e., providing "all" the information
that might be needed by the Orchestrator's PCE) and having up-to-
date information. The more the information is provided and the
longer it takes to keep it up-to-date which increases the likelihood
that the Orchestrator's PCE computes paths using not updated
information.

It seems therefore quite challenging to have a "detailed abstract
connectivity matrix" that provides accurate, scalable and updated
information to allow the Orchestrator's PCE to take optimal
decisions by its own.

If the information in the "detailed abstract connectivity matrix" is
not complete/accurate, we can have the following drawbacks
considering for example the case in Figure 6:

o  If only the VP1-VP4 path with available bandwidth of 2 Gb/s and
   cost 50 is reported, the Orchestrator's PCE will fail to compute
   a 5 Gb/s path between routers R1 and R2, although this would be
   feasible;

o  If only the VP1-VP4 path with available bandwidth of 10 Gb/s and
   cost 60 is reported, the Orchestrator's PCE will compute, as
   optimal, the 1 Gb/s path between R1 and R2 going through the VP2-
   VP5 path within the Optical domain while the optimal path would
   actually be the one going thought the VP1-VP4 sub-path (with cost
   50) within the Optical domain.

Instead, using the approach proposed in this document, the
Orchestrator, when it needs to setup an end-to-end path, it can
request the Optical domain controller to compute a set of optimal
paths (e.g., for VP1-VP4 and VP2-VP5) and take decisions based on
the information received:

o  When setting up a 5 Gb/s path between routers R1 and R2, the
   Optical domain controller may report only the VP1-VP4 path as the
   only feasible path: the Orchestrator can successfully setup the
   end-to-end path passing though this Optical path;

o  When setting up a 1 Gb/s path between routers R1 and R2, the
   Optical domain controller (knowing that the path requires only 1
   Gb/s) can report both the VP1-VP4 path, with cost 50, and the
   VP2-VP5 path, with cost 65. The Orchestrator can then compute the
   optimal path which is passing thought the VP1-VP4 sub-path (with
   cost 50) within the Optical domain.

3.2. TE Topology Abstraction

Using the TE Topology model, as defined in [TE-TOPO], the underlying
SDN controller can export an abstract TE Topology, composed by a set
of TE nodes and TE links, which are abstracting the topology
controlled by each domain controller.

Considering the example in Figure 4, the TE domain controller 1 can
export a TE Topology encompassing the TE nodes A, B, C and D and the
TE Link interconnecting them. In a similar way, TE domain controller
2 can export a TE Topology encompassing the TE nodes E, F, G and H
and the TE Link interconnecting them.

In this example, for simplicity reasons, each abstract TE node maps
with each physical node, but this is not necessary.

In order to setup a multi-domain TE path (e.g., between nodes A and H), the Orchestrator can compute by its own an optimal end-to-end path based on the abstract TE topology information provided by the domain controllers. For example:

o  Orchestrator's PCE, based on its own information, can compute the optimal multi-domain path being A-B-C-E-G-H, and then request the TE domain controllers to setup the A-B-C and E-G-H intra-domain paths

o  But, during path setup, the domain controller may find out that A-B-C intra-domain path is not feasible (as discussed in section 2.2, in optical networks it is typical to have some paths not being feasible due to optical constraints that are known only by the optical domain controller), while only the path A-B-D is feasible

o  So what the hierarchical controller computed is not good and need to re-start the path computation from scratch

As discussed in section 3.1, providing more extensive abstract information from the TE domain controllers to the multi-domain Orchestator may lead to scalability problems.

In a sense this is similar to the problem of routing and wavelength assignment within an Optical domain. It is possible to do first routing (step 1) and then wavelength assignment (step 2), but the chances of ending up with a good path is low. Alternatively, it is possible to do combined routing and wavelength assignment, which is known to be a more optimal and effective way for Optical path setup. Similarly, it is possible to first compute an abstract end-to-end path within the multi-domain Orchestrator (step 1) and then compute an intra-domain path within each Optical domain (step 2), but there are more chances not to find a path or to get a suboptimal path that performing per-domain path computation and then stitch them.

3.3. Complementary use of TE topology and path computation

As discussed in section 2.2, there are some scalability issues with path computation requests in a multi-domain TE network with many TE domains, in terms of the number of requests to send to the TE domain controllers. It would therefore be worthwhile using the TE topology information provided by the domain controllers to limit the number of requests.

An example can be described considering the multi-domain abstract
topology shown in Figure 7. In this example, an end-to-end TE path
between domains A and F needs to be setup. The transit domain should
be selected between domains B, C, D and E.

```
      ------------------------------------------------------------------
      I                                                                I
      I                                                                I
      I                                                                I
      I                  Multi-domain with many domains                I
      I                     (Topology information)                     I
      I                                                                I
      I                                                                I
      I                     (only in PDF version)                      I
      I                                                                I
      I                                                                I
      I                                                                I
      ------------------------------------------------------------------
```
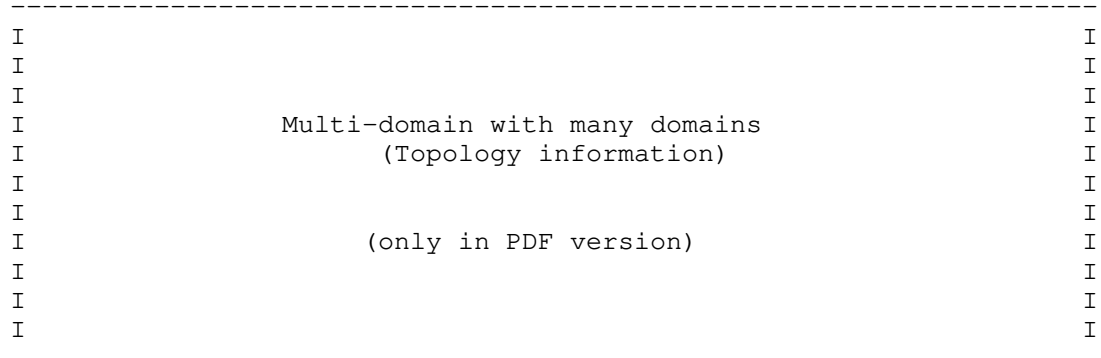
        Figure 7 - Multi-domain with many domains (Topology information)

The actual cost of each intra-domain path is not known a priori from
the abstract topology information. The Orchestrator only knows, from
the TE topology provided by the underlying domain controllers, the
feasibility of some intra-domain paths and some upper-bound and/or
lower-bound cost information. With this information, together with
the cost of inter-domain links, the Orchestrator can understand by
its own that:

o  Domain B cannot be selected as the path connecting domains A and
   E is not feasible;

o  Domain E cannot be selected as a transit domain since it is know
   from the abstract topology information provided by domain
   controllers that the cost of the multi-domain path A-E-F (which
   is 100, in the best case) will be always be higher than the cost
   of the multi-domain paths A-D-F (which is 90, in the worst case)
   and A-E-F (which is 80, in the worst case)

Therefore, the Orchestrator can understand by its own that the
optimal multi-domain path could be either A-D-F or A-E-F but it

cannot known which one of the two possible option actually provides
the optimal end-to-end path.

The Orchestrator can therefore request path computation only to the
TE domain controllers A, D, E and F (and not to all the possible TE
domain controllers).

```
     ------------------------------------------------------------------
    I                                                                  I
    I                                                                  I
    I                                                                  I
    I                 Multi-domain with many domains                  I
    I                  (Path Computation information)                 I
    I                                                                  I
    I                                                                  I
    I                     (only in PDF version)                       I
    I                                                                  I
    I                                                                  I
    I                                                                  I
     ------------------------------------------------------------------
```
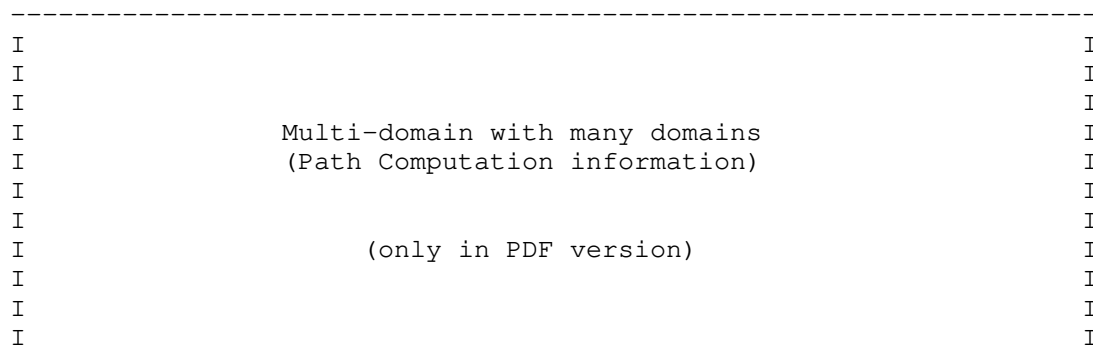
Figure 8 - Multi-domain with many domains (Path Computation
                           information)

Based on these requests, the Orchestrator can know the actual cost
of each intra-domain paths which belongs to potential optimal end-
to-end paths, as shown in Figure 8, and then compute the optimal
end-to-end path (e.g., A-D-F, having total cost of 50, instead of A-
C-F having a total cost of 70).

4. Motivation for a YANG Model

4.1. Benefits of common data models

Path computation requests should be closely aligned with the YANG
data models that provide (abstract) TE topology information, i.e.,
[TE-TOPO] as well as that are used to configure and manage TE
Tunnels, i.e., [TE-TUNNEL]. Otherwise, an error-prone mapping or
correlation of information would be required. For instance, there is
benefit in using the same endpoint identifiers in path computation
requests and in the topology modeling. Also, the attributes used in
path computation constraints could use the same or similar data

models. As a result, there are many benefits in aligning path
computation requests with YANG models for TE topology information
and TE Tunnels configuration and management.

4.2. Benefits of a single interface

A typical use case for path computation requests is the interface
between an orchestrator and a domain controller. The system
integration effort is typically lower if a single, consistent
interface is used between such systems, i.e., one data modeling
language (i.e., YANG) and a common protocol (e.g., NETCONF or
RESTCONF).

Practical benefits of using a single, consistent interface include:

   1. Simple authentication and authorization: The interface between
      different components has to be secured. If different protocols
      have different security mechanisms, ensuring a common access
      control model may result in overhead. For instance, there may
      be a need to deal with different security mechanisms, e.g.,
      different credentials or keys. This can result in increased
      integration effort.
   2. Consistency: Keeping data consistent over multiple different
      interfaces or protocols is not trivial. For instance, the
      sequence of actions can matter in certain use cases, or
      transaction semantics could be desired. While ensuring
      consistency within one protocol can already be challenging, it
      is typically cumbersome to achieve that across different
      protocols.
   3. Testing: System integration requires comprehensive testing,
      including corner cases. The more different technologies are
      involved, the more difficult it is to run comprehensive test
      cases and ensure proper integration.
   4. Middle-box friendliness: Provider and consumer of path
      computation requests may be located in different networks, and
      middle-boxes such as firewalls, NATs, or load balancers may be
      deployed. In such environments it is simpler to deploy a single
      protocol. Also, it may be easier to debug connectivity
      problems.
   5. Tooling reuse: Implementers may want to implement path
      computation requests with tools and libraries that already
      exist in controllers and/or orchestrators, e.g., leveraging the
      rapidly growing eco-system for YANG tooling.

4.3. Extensibility

   Path computation is only a subset of the typical functionality of a
   controller. In many use cases, issuing path computation requests
   comes along with the need to access other functionality on the same
   system. In addition to obtaining TE topology, for instance also
   configuration of services (setup/modification/deletion) may be
   required, as well as:

   1. Receiving notifications for topology changes as well as
      integration with fault management
   2. Performance management such as retrieving monitoring and
      telemetry data
   3. Service assurance, e.g., by triggering OAM functionality
   4. Other fulfilment and provisioning actions beyond tunnels and
      services, such as changing QoS configurations

   YANG is a very extensible and flexible data modeling language that
   can be used for all these use cases.

   Adding support for path computation requests to YANG models would
   seamlessly complement with [TE-TOPO] and [TE-TUNNEL] in the use
   cases where YANG-based protocols (e.g., NETCONF or RESTCONF) are
   used.

5. Path Optimization Request

   This is for further study

6. YANG Model for requesting Path Computation

   Work on extending the TE Tunnel YANG model to support the need to
   request path computation has recently started also in the context of
   the [TE-TUNNEL] draft.

   It is possible to request path computation by configuring a
   "compute-only" TE tunnel and retrieving the computed path(s) in the
   LSP(s) Record-Route Object (RRO) list as described in [TE-TUNNEL].

   This is a stateful solution since the state of each created
   "compute-only" TE tunnel needs to be maintained and updated, when
   underlying network conditions change.

   The need also for a stateless solution, based on an RPC, has been
   recognized.

   The YANG model to support stateless RPC is for further study.

7. Security Considerations

   This is for further study

8. IANA Considerations

   This document requires no IANA actions.

9. References

9.1. Normative References

   [RFC7926] Farrel, A. et al., "Problem Statement and Architecture for
             Information Exchange Between Interconnected Traffic
             Engineered Networks", RFC 7926, July 2016.

   [TE-TOPO] Liu, X. et al., "YANG Data Model for TE Topologies",
             draft-ietf-teas-yang-te-topo, work in progress.

   [TE-TUNNEL] Xhang, X. et al., "A YANG Data Model for Traffic
             Engineering Tunnels and Interfaces", draft-ietf-teas-yang-
             te, work in progress.

9.2. Informative References

   [RFC7446] Lee, Y. et al., "Routing and Wavelength Assignment
             Information Model for Wavelength Switched Optical
             Networks", RFC 7446, February 2015.

   [L1-TOPO] Zhang, X. et al., "A YANG Data Model for Layer 1 (ODU)
             Network Topology", draft-zhang-ccamp-l1-topo-yang, work in
             progress.

   [ACTN-Frame]  D, Ceccarelli and Y. Lee (editors), "Framework for
             Abstraction and Control of Transport Networks," draft-
             ceccarelli-actn-framework, work in progress.

   [ACTN-Info] Y. Lee and S. Belotti, D. Dhody and D. Ceccarelli,
             "Information Model for Abstraction and Control of
             Transport Networks", draft-leebelotti-actn-info, work in
             progress.

10. Acknowledgments

   The authors would like to thank Igor Bryskin and Xian Zhang for
   participating in discussions and providing valuable insights.

   This document was prepared using 2-Word-v2.0.template.dot.

Contributors

   Dieter Beller
   Nokia
   Email: dieter.beller@nokia.com

Authors' Addresses

   Italo Busi
   Huawei
   Email: italo.busi@huawei.com


   Sergio Belotti
   Nokia
   Email: sergio.belotti@nokia.com


   Victor Lopez
   Telefonica
   Email: victor.lopezalvarez@telefonica.com


   Oscar Gonzalez de Dios
   Telefonica
   Email: oscar.gonzalezdedios@telefonica.com


   Anurag Sharma
   Infinera
   Email: AnSharma@infinera.com


   Yan Shi
   China Unicom
   Email: shiyan49@chinaunicom.cn


   Ricard Vilalta
   CTTC
   Email: ricard.vilalta@cttc.es

Karthik Sethuraman
NEC
Email: karthik.sethuraman@necam.com


Michael Scharf
Nokia
Email: michael.scharf@nokia.com


Daniele Ceccarelli
Ericsson
Email: daniele.ceccarelli@ericsson.com

                        RSVP Extensions for RMR
                   draft-deshmukh-rsvp-rmr-extension-00

Abstract

   Rings are the most common topology in access and aggregation
   networks.  However, the use of MPLS as the transport protocol for
   rings is very limited today.  draft-ietf-mpls-rmr-02 describes a
   mechanism to handle rings efficiently using MPLS.  This document
   describes the extensions to the RSVP protocol for signaling MPLS
   label switched paths in rings.

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].

Table of Contents

1.  Introduction

   This document extends RSVP-TE [RFC3209] to establish label-switched
   path (LSP) tunnels in the ring topology.  Rings are auto-discovered
   using the mechanisms mentioned in the [draft-ietf-mpls-rmr-02].
   Either IS-IS [RFC5305] or OSPF[RFC3630] can be used as the IGP for
   auto-discovering the rings.

   After the rings are auto-discovered, each ring node knows its
   clockwise (CW) and anticlockwise (AC) ring neighbors and its ring
   links.  All of the express links in the ring also get identified as
   part of the auto-discovery process.  At this point, every node in the
   ring informs the RSVP protocol to begin the signaling of the ring
   LSPs.

   Section 2 covers the terminology used in this document.  Section 3
   presents the RSVP protocol extensions needed to support MPLS rings.
   Section 4 describes the procedures of RSVP LSP signaling in detail.

2.  Terminology

   A ring consists of a subset of n nodes {R_i, 0 <= i < n}.  We define
   the direction from node R_i to R_i+1 as "clockwise" (CW) and the
   reverse direction as "anti-clockwise" (AC).  As there may be several
   rings in a graph, we number each ring with a distinct ring ID RID.

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

                    Figure 1: Ring with 8 nodes

   The following terminology is used for ring LSPs:

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

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

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

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

   Ring links:  Links that connect ring neighbors.

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

   MP2P LSP:  Each LSP in the ring is a multipoint to point LSP such
      that LSP can have multiple ingress nodes and one egress node.

3.  RSVP Extensions

   Since the procedures of signaling ring LSPs will be different from
   the signaling of regular RSVP LSPs, a new C-Type is defined here for
   the SESSION object.  This new C-Type will help to clearly

   differentiate ring LSPs from regular LSPs.  In addition, new flags
   are introduced in the SESSION object to represent the ring direction
   of the corresponding Path message.

3.1.  Session Object

   Class = SESSION, LSP_TUNNEL_IPv4 C-Type = TBD

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Ring anchor node address                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Ring Flags           |            MBB ID             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            Ring ID                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
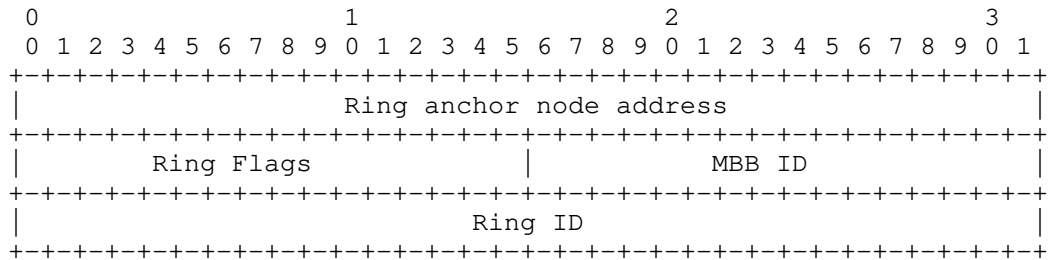
                          SESSION Object

   Ring anchor node address:   IPv4 address of the anchor node.  Each
      anchor node creates a LSP addressed to itself.

   MBB ID:   A 16-bit identifier used in the SESSION.  This "Make-
      before-break" (MBB) ID is useful for graceful ring changes.  If a
      new node is being added to the ring or some existing node goes
      down and we have to signal a smaller ring, in those cases, anchor
      node creates a new tunnel with a different "MBB ID".

   Ring ID:   A 32-bit number that identifies a ring; this is unique in
      some scope of a Service Provider's network.  This number remains
      constant throughout the existence of ring.

   Ring Flags:   For each ring, the anchor node starts signaling of a
      ring LSP.  Ring LSP RL_i, anchored on node R_i, consists of two
      counter-rotating unicast LSPs that start and end at R_i.  One LSP
      will be in the clockwise direction and other LSP will be in the
      anti-clockwise direction.  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).  Two new flags
      are defined in the SESSION object which define the ring direction
      of the corresponding Path message.

   ClockWise(CW) Direction  0x01:   This flag indicates that the
      corresponding Path message is traveling in the ClockWise(CW)
      direction along the ring.

   Anti-ClockWise(AC) Direction  0x02:   This flag indicates that the
      corresponding Path message is traveling in the Anti-ClockWise(AC)
      direction along the ring.

3.2.  SENDER_TEMPLATE,FILTER_SPEC Objects

   There will be no changes to the SENDER_TEMPLATE and FILTER_SPEC
   objects.  The format of the above 2 objects will be similar to the
   definitions in RFC 3209.  [RFC3209] Only the semantics of these
   objects will slightly change.  This will be explained in section
   Section 4.5 below.

4.  Ring Signaling Procedures

   A ring node indicates in its IGP updates the ring LSP signaling
   protocols that it supports.  This can be LDP and/or RSVP-TE.
   Ideally, each node should support both.  If the ring is configured
   with RSVP as the signaling protocol, then once a ring node R_i knows
   the RID, its ring links and directions, it kicks off ring RSVP LSP
   signaling automatically.

4.1.  Differences from regular RSVP-TE LSPs

   Ring LSPs differ from regular RSVP-TE LSPs in several ways:

   1.  Ring LSPs (by construction) form a loop.

   2.  Ring LSPs are multipoint-to-point.  Any ring node can inject
   traffic into a ring LSP.

   3.  The bandwidth of a ring LSP can change hop-to-hop.

   4.  Ring LSPs are protected without the use of bypass or detour LSPs.
   Ring LSP protection is akin to SONET/SDH ring protection.

4.2.  LSP signaling

   After the ring auto-discovery process, each anchor node creates a LSP
   addressed to itself.  This ring LSP contains of a pair of counter-
   rotating unicast LSPs.  So, for a ring containing N nodes, there will
   be 2N total LSPs signaled.

   There is no need for ERO object in the Path message.  The Path
   message for ring LSPs has the following format:

```
<Path Message> ::=  <Common Header> [ <INTEGRITY> ]
                         <SESSION> <RSVP_HOP>
                         <TIME_VALUES>
                         <LABEL_REQUEST>
                         [ <SESSION_ATTRIBUTE> ]
                         <sender descriptor list>

<sender descriptor list> ::= <sender descriptor>|
                                   <sender descriptor list> <sender descri
ptor>
<sender descriptor> ::= <SENDER_TEMPLATE> <SENDER_TSPEC>
```

The anchor node creates 2 Path messages traveling in opposite
directions.  The SESSION format MUST be as per the description in
Section 3.1.  The anchor node which creates the LSP will insert it's
own address in the "Ring node anchor address" field of the SESSION
object.  So effectively, the Path messages are addressed to the
originating node itself.

The SESSION flags of these 2 Path messages are different.  The Path
message sent to the CW neighbor MUST have the CW flag set in the
SESSION object to signal the LSP going in the clockwise direction.
The Path message sent to the AC neighbor MUST have the AC flag set to
signal the LSP in the anti-clockwise direction.  The details for
signaling over express links will be given in a future version.

When an incoming Path message is received at the ring node R_i, it
consults the results of auto-discovery to find the appropriate ring
neighbor.  If the incoming Path message has CW direction flag set,
then R_i sends a Path message to its CW ring neighbor (and vice
versa).  Thus, there is no need of ERO in the Path message.  The Path
message is routed locally at each ring based on the ring auto-
discovery calculations.

The RESV message for ring LSPs also uses the new RING_IPv4 SESSION
object.  When the Path message originated from the anchor node R_i
reaches back to R_i, R_i generates a Resv message.  Note that this
means that anchor node is both Ingress and Egress for the Path
message.  The Resv message copies the same ring flags as received in
the corresponding Path message.  So, a Resv message for a CW LSP goes
in the AC direction (unlike the Path message, which goes CW).  This
is done to correctly match Path and corresponding Resv messages at
transit ring nodes.  Upon receiving Resv message with CW flag set,
the ring node will forward the Resv message to its AC neighbor.

Each ring node R_i allocates CW and AC labels for each ring LSP RL_k.
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.

Consider the following three nodes of the ring, and their signaling
interactions for LSP RL_5 originating from anchor node R5:


```
                        P5_CW ->      P5_CW ->
                        Q5_CW <-      Q5_CW <-
        ... ------ R7 --------- R8 --------- R9 ------ ...
                        P5_AC <-      P5_AC <-
                        Q5_AC ->      Q5_AC ->
```


P corresponds to the Path message and Q corresponds to the Resv
message.

Also, since ring LSPs are MP2P in nature, each ring node SHOULD also
signal a Path message towards anchor node.  The procedure for that is
as follows:

When a ring node R5 receives a Path message initiated by anchor node
R1(for anchor lsp "lsp1"), R5 SHOULD make a copy of the received Path
message for "lsp1".  R5 then modifies the sender-template object from
the copied Path message for "lsp1".  In the sender-template object,
R5 uses the sender address as the loopback address of node R5 and
lsp-id = X.  R5 then forwards this new Path message to it's ring
neighbor.  The original anchor Path message has sender address as
loopback address of R1 and lsp-id = X.

So at this point, there will be 2 different path messages existing
for lsp1 First Path message will be for the anchor LSP with sender
address = node R1.  Second Path message will be for the ring LSP with
sender address = node R5.

When node R1 receives this modified Path message, it replies with the
Resv message containing the same label it advertised for the original
anchor lsp "lsp1".  The SESSION object of the Resv message will also
exactly match with the received Path message.  Only the FILTER_SPEC
object in the Resv message will have the sender address as loopback
of node R5.  As this Resv message propagates back towards R5, all the
transit nodes also send the same label that they have allocated for
the original anchor lsp "lsp1".  So no new label routes get installed
as part of signaling for this ring lsp.  The anchor LSP and all of
their associated ring LSPs share label routes.  The label actions are
described below in Section 4.3.

4.3.  Protection

   In the rings, there are no protection LSPs -- no node or link bypass
   LSPs, no standby LSPs and no detours.  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.

   Since each ring LSP is a MP2P LSP, any ring node can inject traffic
   onto a LSP whose anchor might be a different ring node.  To achieve
   the above, an ingress route will be installed as follows at every
   ring node J, for a given ring-LSP with anchor Rk (say 1.2.3.4).

```
            1.2.3.4  ->  (Push CL_J+1,K, NH: R_J+1)        # CW
                     ->  (Push AL_J-1,K, NH: R_J-1)        # AC

                     CL = Clockwise label
                     AL = Anti-Clockwise label
```

   Traffic will either be load balanced in the CW and AC directions or
   the traffic will be sent on just CW or AC lsp based on parameters
   such as hop-count, policy etc.

   Also, 2 transit routes will be installed for the anchor LSP
   transiting from node Rj as follows:

```
            CL_J,K ->  SWAP(CL_J+1,K,  NH: R_J+1)               #CW
                   ->  SWAP(AL_J-1,K , NH: R_J-1)               #AC

                     CL = Clockwise label
                     AL = Anti-Clockwise label
                     CW NH has weight 1, AC NH has higher-weight.


            AL_J,K -> SWAP(AL_J-1,K , NH: R_J-1)   #AC
                   -> SWAP(CL_J+1,K,  NH: R_J+1)   #CW

                     CL = Clockwise label
                     AL = Anti-Clockwise label
                     AC NH has weight 1, CW NH has higher weight.
```

   Suppose a packet headed in anti-clockwise direction towards R5 and it
   arrives at node R8.  Lets say that now R8 learns there is a link

failure in the AC direction.  R8 reroutes this packet back onto the
clockwise direction.  This reroute action is pre-programmed in the
LFIB, to minimize the time between detection of a fault and the
corresponding recovery action.

At this time, R8 also sends a notification to R7 that the AC
direction is not working, so that R7 can similarly switch traffic to
the CW direction.  These notification SHOULD propagate CW until each
traffic source on the ring CW of the failure uses the CW
direction.For RSVP-TE, this notification is sent in the form of
PathErr message.

To provide this notification, the ring node detecting failure SHOULD
send a Path Error message with error code of "Notify" and an error
value field of ("Tunnel locally repaired").  This Path Error code and
value is same as defined in RFC 4090[RFC4090] for the notification of
local repair.

Note that the failure of a node or a link will not necessarily affect
all ring LSPs.  Thus, it is important to identify the affected LSPs
and only switch the affected LSPs.

## 4.4.  Ring changes

A ring node can go down resulting in a smaller ring or a new node can
be added to the ring which will increase the ring size.  In both of
the above cases, the ring auto-discovery process SHOULD kick in and
it SHOULD calculate a new ring with the changed ring nodes.

When the ring auto-discovery process is complete, IGP will signal
RSVP to begin the MBB process for the existing ring LSPs.  For this
MBB process, the anchor node will create a new Path message with a
different "MBB ID" in the SESSION object.  All other fields in the
SESSION Object will remain same as the existing Path message(before
the ring change).

This new Path message will then propagate along the ring neighbors in
the same way as the original Path message.  Each ring neighbor SHOULD
forward the Path message to it's appropriate neighbor based on the
new auto-discovery calculations.

For the ring links which are common between the old and new LSPs, the
LSPs will share resources(SE style reservation) on those ring links.
Note that here we are using MBB_ID in the SESSION object to share
resources instead of the LSP_ID in the SENDER_TEMPLATE Object(which
is used in RSVP-TE for sharing resources as described in RFC 3209
[RFC4090]).  The LSP_ID use is reserved for a different functionality
as described in section Section 4.5.

4.5.  Bandwidth management

   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
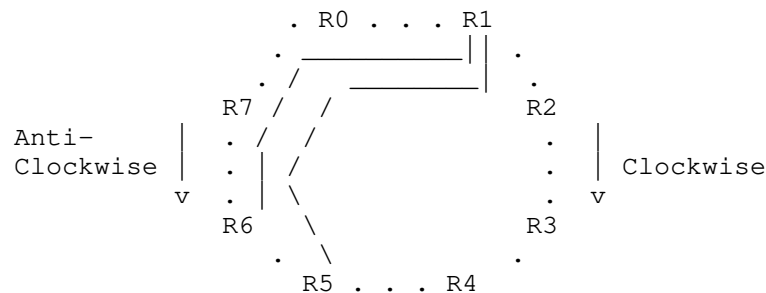   admitted to the ring.

```
                         . R0 . . . R1
                       . _____||  .
                     . / _____||  .
                    R7 / /  _____|        R2
       Anti-        | . / /              .  |
       Clockwise    | . | /              .  | Clockwise
              v     . | \              . v
                    R6   \                R3
                     . \              .
                       R5 . . . R4
```

                 Figure 2: BW Management in Ring with 8 nodes

   Let's say that Ring node R5 wants to increase the BW for the LSP
   whose egress is at node R1.  To achieve this BW increase, Ring node
   R5 has to increase BW along the LSP anchored at node R1(say lsp1).

   R5 makes a copy of the existing ring Path message for lsp1.  R5 then
   modifies the sender-template object from the copied Path message for
   "lsp1".  In the sender-template object, R5 uses the sender address as
   the loopback address of node R5 and lsp-id = X+1.  R5 also modifies
   the TSPEC object which represents the BW increase/decrease in this
   new Path message.  R5 then forwards this new Path message to it's
   ring neighbor.  Note that R5 MUST also continue signaling the
   original anchor Path message received from ring node R1 for lsp1.
   The original anchor Path message has sender address as loopback
   address of R1.

   Now, let's say, node 5 wants to increase BW again for lsp1, then R5
   adds a new SENDER_TEMPLATE object in the existing Path message for
   "lsp1" with sender address as loopback of node 5 and lsp-id = X+2.
   So at this point, there will be 2 different path messages existing
   for lsp1 First Path message will be for the anchor LSP with sender
   address = node 1.  Second Path message will contain 2 SENDER_TEMPLATE
   objects as [node5, lsp-id = X+1] and [node5, lsp-id =X+2].

   Similarly, if node R6 wants to increase the BW for "lsp1", it SHOULD
   create a new Path message containing SENDER_TEMPLATE object with

sender address = loopback of node 6 and lsp-id = Y+1.  Thus, the LSP-
ID field is local to each sender node along the ring.

If sufficient BW is available all the way towards ring node R1, then
this new Path message reaches node R1.  R1 generates a Resv message
with the correct FILTER_SPEC object corresponding to the received
SENDER_TEMPLATE object.  This Resv message will also have the correct
FLOWSPEC object as per the requested bandwidth.

If sufficient BW is not available at some downstream (say node R9),
then ring node R9 SHOULD generate a PathErr message with the
corresponding Sender Template Object.  When node R5 receives this
PathErr message, R5 understands that the BW increase was not
successful.  Note that the existing established bandwidths for lsp1
are not affected by this new PathErr message.

When ring node R5 no longer needs the BW reservation, then ring node
R5 SHOULD originate a PathTear message with the appropriate Sender
Template Object as described above.  Every downstream node SHOULD
then remove bandwidth allocated on the corresponding link on receipt
of this PathTear message.

Also, note that as part of this BW increase or decrease process, any
ring node does not actually change any label associated with the LSP.
So, the label remains same as it was signaled initially when the
anchor LSP came up.

5.  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.

6.  Contributors

Ravi Singh
Juniper Networks, Inc.
1194 N Mathilda Ave
Sunnyvale, CA  94089
USA

Email: ravis@juniper.net

        Santosh Esale
        Juniper Networks, Inc.
        1194 N Mathilda Ave
        Sunnyvale, CA  94089
        USA


        Email: sesale@juniper.net

        Raveendra Torvi
        Juniper Networks, Inc.
        10 Technology Park Dr
        Westford, MA  01886
        USA


        Email: rtorvi@juniper.net

7.  IANA Considerations

   Requests to IANA will be made in a future version of this document.

8.  References

8.1.  Normative References

   [I-D.ietf-mpls-rmr]
              Kompella, K. and L. Contreras, "Resilient MPLS Rings",
              draft-ietf-mpls-rmr-02 (work in progress), July 2016.

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

8.2.  Informative References

   [I-D.dai-mpls-rsvp-te-mbb-label-reuse]
              Dai, M. and M. Chaudhry, "MPLS RSVP-TE MBB Label Reuse",
              draft-dai-mpls-rsvp-te-mbb-label-reuse-01 (work in
              progress), September 2015.

   [RFC2205]  Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S.
              Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1
              Functional Specification", RFC 2205, DOI 10.17487/RFC2205,
              September 1997, <http://www.rfc-editor.org/info/rfc2205>.

   [RFC3209]  Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V.,
              and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP
              Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001,
              <http://www.rfc-editor.org/info/rfc3209>.

   [RFC3630]  Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering
              (TE) Extensions to OSPF Version 2", RFC 3630,
              DOI 10.17487/RFC3630, September 2003,
              <http://www.rfc-editor.org/info/rfc3630>.

   [RFC4090]  Pan, P., Ed., Swallow, G., Ed., and A. Atlas, Ed., "Fast
              Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090,
              DOI 10.17487/RFC4090, May 2005,
              <http://www.rfc-editor.org/info/rfc4090>.

   [RFC5305]  Li, T. and H. Smit, "IS-IS Extensions for Traffic
              Engineering", RFC 5305, DOI 10.17487/RFC5305, October
              2008, <http://www.rfc-editor.org/info/rfc5305>.

Authors' Addresses

   Abhishek Deshmukh
   Juniper Networks, Inc.
   10 Technology Park Dr
   Westford, MA  01886
   USA


   Email: adeshmukh@juniper.net


   Kireeti Kompella
   Juniper Networks, Inc.
   1194 N Mathilda Ave
   Sunnyvale, CA  94089
   USA


   Email: kireeti@juniper.net

A YANG model to manage the optical interface parameters for an external
                    transponder in a WDM network
             draft-dharini-ccamp-dwdm-if-param-yang-00

Abstract

   This memo defines a Yang model related to the Optical Transceiver
   optical parameters characterising the 100G and above interfaces.
   100G and above Transceivers support coherent transmission, different
   modulation format, multiple FEC algorithms not yet specified by ITU-T
   G.698.2 [ITU.G698.2] or any other ITU-T recommendation.  The use
   cases and the state of the Coherent transceivers is well describe in
   draft-many-coherent-DWDM-if-control.

   The Yang model defined in this memo can be used for Optical
   Parameters monitoring and/or configuration of the endpoints of the
   multi-vendor IaDI optical link.

This Internet-Draft will expire on May 4, 2017.

Copyright Notice

Table of Contents

1.  Introduction

   This memo defines a Yang model that translates and obsolete the SNMP
   mib module defined in draft-galikunze-ccamp-dwdm-if-snmp-mib for
   managing single channel optical interface parameters of DWDM
   applications, using the approach specified in G.698.2.  This model is
   to support the optical parameters specified in ITU-T G.698.2
   [ITU.G698.2], plus some parameters related to full coherent
   transmission and not yet specified by ITU-T like modulation format,
   finer Grid provisioning, multiple carrier, etc.  The application
   identifiers specified in ITU-T G.874.1 [ITU.G874.1] and the Optical
   Power at Transmitter and Receiver side.  Note that G.874.1
   encompasses vendor-specific codes, which if used would make the
   interface a single vendor IaDI and could still be managed.

   [Editor's note: In G.698.2 this corresponds to the optical path from
   point S to R; network media channel is also used and explained in
   draft-ietf-ccamp-flexi-grid-fwk-02]

   Management will be performed at the edges of the network media
   channel (i.e., at the transmitters and receivers attached to the S
   and R reference points respectively) for the relevant parameters
   specified in G.698.2 [ITU.G698.2], G.798 [ITU.G798], G.874
   [ITU.G874], and the performance parameters specified in G.7710/Y.1701
   [ITU-T G.7710] and G.874.1 [ITU.G874.1].

   G.698.2 [ITU.G698.2] is primarily intended for metro applications
   that include optical amplifiers.  Applications are defined in G.698.2
   [ITU.G698.2] using optical interface parameters at the single-channel
   connection points between optical transmitters and the optical
   multiplexer, as well as between optical receivers and the optical
   demultiplexer in the DWDM system.  This Recommendation uses a
   methodology which does not explicitly specify the details of the
   optical network between reference point Ss and Rs, e.g., the passive
   and active elements or details of the design.  The Recommendation
   currently includes unidirectional DWDM applications at 2.5 and 10
   Gbit/s (with 100 GHz and 50 GHz channel frequency spacing).  Work is
   still under way for 40, 100 and Higher Gbit/s interfaces.  There is
   possibility for extensions to a lower channel frequency spacing.
   This document specifically refers also to the "application code"
   defined in the G.698.2 [ITU.G698.2] and included in the Application
   Identifier defined in G.874.1 [ITU.G874.1] and G.872 [ITU.G872], plus
   a few optical parameters not included in the G.698.2 application code
   specification.

   This draft refers and supports the draft-ietf-ccamp-dwdm-if-mng-ctrl-
   fwk and draft-many-coherent-DWDM-if-control.

The building of a yang model describing and extending the optical
parameters defined in G.698.2 [ITU.G698.2], and reflected in G.874.1
[ITU.G874.1], allows the different vendors and operator to retrieve,
provision and exchange information across the G.698.2 multi-vendor
IaDI in a standardised way.  In addition to the parameters specified
in ITU recommendations the Yang models support also the "vendor
specific application identifier", the Tx and Rx power at the Ss and
Rs points and the channel frequency and the detailed parameters
described in G.698.2 extending them to the new 100G and higher
coherent interfaces..

The Yang Model, reporting the Optical parameters and their values,
characterizes the features and the performances of the optical
components and allow a reliable link design in case of multi vendor
optical networks.

2.  The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current
Internet-Standard Management Framework, please refer to section 7 of
RFC 3410 [RFC3410].

This memo specifies a Yang model for optical interfaces.
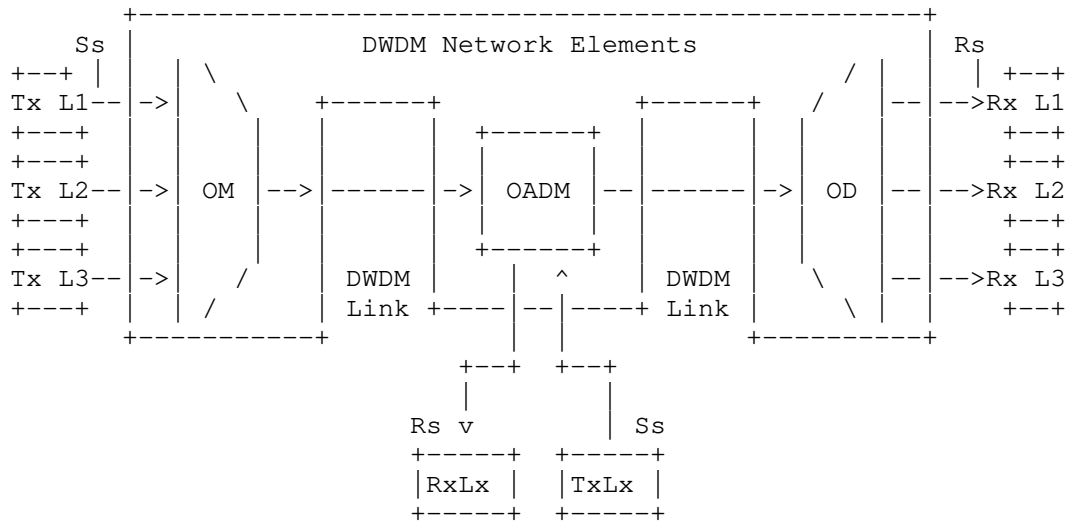
3.  Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in RFC 2119 [RFC2119] In
the description of OIDs the convention: Set (S) Get (G) and Trap (T)
conventions will describe the action allowed by the parameter.

4.  Overview

Figure 1 shows a set of reference points, for single-channel
connection between transmitters (Tx) and receivers (Rx).  Here the
DWDM network elements include an OM and an OD (which are used as a
pair with the opposing element), one or more optical amplifiers and
may also include one or more OADMs.

```
              +-------------------------------------------------+
       Ss     |               DWDM Network Elements      |     Rs
    +--+ |    |  \                                   /   |    |  +--+
    Tx L1--|->|   \        +------+        +------+  /   |--|-->Rx L1
    +---+  |  |    \        |      |  +------+      |  /   |    |   +--+
    +---+  |  |     \       |      |  |      |      |      |    |   +--+
    Tx L2--|->|  OM |-->|------|->| OADM |--|------|->| OD |--|-->Rx L2
    +---+  |  |     /       |      |  |      |      |      |    |   +--+
    +---+  |  |    /        |      |  +------+      |      |    |   +--+
    Tx L3--|->|   /         | DWDM |    ^   | DWDM  |  \   |--|-->Rx L3
    +---+  |  |  /          | Link +----|--|----+ Link |  \ |  |   +--+
           +----------+     +------|--|----+          +----------+
                                  +--+  +--+
                                  |       |
                            Rs v  |       | Ss
                          +-----+  +-----+
                          |RxLx |  |TxLx |
                          +-----+  +-----+
```

Ss = reference point at the DWDM network element tributary output
Rs = reference point at the DWDM network element tributary input
Lx = Lambda x
OM = Optical Mux
OD = Optical Demux
OADM = Optical Add Drop Mux


from Fig. 5.1/G.698.2

          Figure 1: External transponder in WDM netwoks

4.1.  Optical Parameters Description

   The link between the external transponders through a WDM network
   media channels are managed at the edges, i.e. at the transmitters
   (Tx) and receivers (Rx) attached to the S and R reference points
   respectively.  The set of parameters that could be managed are
   defined by the "application code" notation

   The definitions of the optical parameters are provided below to
   increase the readability of the document, where the definition is

ended by (R) the parameter can be retrieve with a read, when (W) it
can be provisioned by a write, (R,W) can be either read or written.

4.1.1.  Table of Application Codes

This table has a list of Application codes supported by this
interface at point R are defined in G.698.2.

Application code Identifier:
   The Identifier for the Application code.

Application code Type:
   This parameter indicates the transceiver type of application
   code at Ss and Rs as defined in [ITU.G874.1], that is used by
   this interface Standard = 0, PROPRIETARY = 1
   If Proprietary the first 6 octets of the printable string will
   be the OUI (organizationally unique identifier) assigned to
   the vendor whose implementation generated the Application
   Identifier Code.

Application code:
   This is the application code that is defined in G.698.2 or the
   vendor generated code which has the OUI.

Number of Single-channel application codes Supported:
   This parameter indicates the number of Single-channel
   application codes supported by this interface

Application code Length:
   The number of octets in the Application Code.


4.1.2.  Rs-Ss Configuration and operating parameters

The Rs-Ss configuration table allows configuration of Central
Frequency, Power and Application codes as described in [ITU.G698.2]
and G.694.1 [ITU.G694.1] and other parameters related to new high
speed coherent interfaces.

Number of subcarriers:
   This parameter indicates the number of subcarriers available for
   the super-channel in case the Transceiver can support multipla
   carrier Circuits.

Current Laser Output power:
   This parameter report the current Transceiver Output power, it can
   be either a setting and measured value (R/W).

Central frequency (see G.694.1 Table 1):
   This parameter indicates the Central frequency value that Ss and
   Rs will be set to work (in THz).  See the details in Section 6/
   G.694.1 or based on "n" and "k" values in case of multicarrier
   transceivers (R/W).

Central frequency franularity:
   This parameter indicates the Central frequency granularity
   supported by the transceiver, this value is combined with K and n
   value to calculate the central frequency on the carrier or sub-
   carriers (R).

Current Laser Input power:
   This parameter report the current Transceiver Input power (G).

Minimum channel spacing:
   This is the minimum nominal difference in frequency (in GHz)
   between two adjacent channels (or carriers) depending on the
   Transceiver characteristics (R).

Bit rate / Baud rate of optical tributary signals:
   Optical tributary signal bit (for NRZ signals) rate or Symbol (for
   Multiple bit per symbol) rate .

FEC Coding:
   This parameter indicate what Forward Error Correction (FEC) code
   is used at Ss and Rs (R/W) (not mentioned in G.698).  .

Maximum bit error ratio (BER):
   This parameter indicate the maximum Bit error rate can be
   supported by the application at the Receiver.  In case of FEC
   applications it is intended after the FEC correction (R) .

Wavelength Range (see G.694.1):  [ITU.G694.1]
   This parameter indicate minimum and maximum wavelength spectrum
   (R) in a definite wavelength Band (L, C and S).

Modulatoin format:
   This parameter indicates the list of supported Modulation Formats
   and the provisioned Modulation Format.  (R/W).

Inter carrier skew:
   This parameter indicates, in case of multi-carrier transceivers
   the maximum skew between the sub-carriers supported by the
   transceiver (R).

4.2.  Parameters at Ss

   The following parameters for the interface at point S are defined in
   G.698.2 [ITU.G698.2].

   Maximum and minimum mean channel output power:
      The mean launched power at Ss is the average power (in dBm) of a
      pseudo-random data sequence coupled into the DWDM link.  It is
      defined as the rhange (Max and Min ) of the parameter (R/W)

   Minimum and maximum central frequency:
      The central frequency is the nominal single-channel frequency (in
      THz) on which the digital coded information of the particular
      optical channel is modulated by use of the NRZ line code.  The
      central frequencies of all channels within an application lie on
      the frequency grid for the minimum channel spacing of the
      application given in ITU-T Rec. G.694.1.  This parameter give the
      Maximum and minimum frequency interval the channel must be
      modulated (R)

   Maximum spectral excursion:
      This is the maximum acceptable difference between the nominal
      central frequency (in GHz) of the channel and the minus 15 dB
      points of the transmitter spectrum furthest from the nominal
      central frequency measured at point Ss.  (R)

   Maximum transmitter (residual) dispersion OSNR penalty (B.3/G.959.1)
     [ITU.G959.1]
      Defines a reference receiver that this penalty is measured with.
      Lowest OSNR at Ss with worst case (residual) dispersion minus the
      Lowest OSNR at Ss with no dispersion.  Lowest OSNR at Ss with no
      dispersion (R)

   Minimum side mode suppression ratio, Minimum channel extinction
   ratio, Eye mask:
      Although are defined in G.698.2 are not supported by this draft
      (R).

   Current Laser Output power:
      This parameter report the current Transceiver Output power, it can
      be either a setting and measured value (R/W) NEED TO DISCUSS ON
      THIS.

4.3.  Interface at point Rs

   The following parameters for the interface at point R are defined in
   G.698.2.

4.3.1.  Mandatory parameters

   Maximum and minimum mean input power:
      The maximum and minimum values of the average received power (in
      dBm) at point Rs.  (R)

   Minimum optical signal-to-noise ratio (OSNR):
      The minimum optical signal-to-noise ratio (OSNR) is the minimum
      value of the ratio of the signal power in the wanted channel to
      the highest noise power density in the range of the central
      frequency plus and minus the maximum spectral excursion (R)

   Receiver OSNR tolerance:
      The receiver OSNR tolerance is defined as the minimum value of
      OSNR at point Rs that can be tolerated while maintaining the
      maximum BER of the application.  (R)

   Maximum reflectance at receiver:
      Although is defined in G.698.2, this parameter is not supported by
      this draft (R).

4.3.2.  Optional parameters

   Current Chromatic Dispersion (CD):
      Residual Chromatic Dispersion measuread at Rx Transceiver port
      (R).

   Current Optical Signal to Noise Ratio (OSNR):
      Current Optical Signal to Noise Ratio (OSNR) estimated at Rx
      Transceiver port (R).

   Current Quality factor (Q):
      "Q" factor estimated at Rx Transceiver port (R).

4.3.3.  Optical path from point Ss to Rs

   The following parameters for the optical path from point S and R are
   defined in G.698.2 and are covered by draft-ggalimbe-ccamp-iv-yang
   [ITU.G698.2].

4.4.  Use Cases

   The use cases are described in draft-ietf-ccamp-dwdm-if-mng-ctrl-fwk

4.5.  Optical Interface for external transponder in a WDM network

   The ietf-ext-xponder-wdm-if is an augment to the ietf-interface.  It
   allows the user to set the application code/vendor transceiver class/
   Central frequency and the output power.  The module can also be used
   to get the list of supported application codes/transceiver class and
   also the Central frequency/output power/input power of the interface.

```
        module: ietf-ext-xponder-wdm-if
        augment /if:interfaces/if:interface:
          +--rw optIfOChRsSs
                +--rw if-current-application-code
                │  +--rw application-code-id    uint8
                │  +--rw application-code-type  uint8
                │  +--rw application-code-length uint8
                │  +--rw application-code?     string
                +--ro if-supported-application-codes
                │  +--ro number-application-codes-supported?   uint32
                │  +--ro application-codes-list* [application-code-id]
                │     +--ro application-code-id    uint8
                │     +--rw application-code-type  uint8
                │     +--rw application-code-length uint8
                │     +--ro application-code?     string
                +--rw output-power?                    int32
                +--ro input-power?                     int32
                +--rw central-frequency?               uint32


         notifications:
        +---n opt-if-och-central-frequency-change
        │  +--ro if-name?      leafref
        │  +--ro new-central-frequency
        │     +--ro central-frequency?   uint32
        +---n opt-if-och-application-code-change
        │  +--ro if-name?             leafref
        │  +--ro new-application-code
        │     +--ro appliation-code-id?  uint8
        │     +--rw application-code-type  uint8
        │     +--rw application-code-length uint8
        │     +--ro application-code?      string
```

5.  Structure of the Yang Module

   ietf-ext-xponder-wdm-if is a top level model for the support of this
   feature.

6.  Yang Module

   The ietf-ext-xponder-wdm-if is defined as an extension to ietf
   interfaces.

```
<CODE BEGINS> file "ietf-ext-xponder-wdm-if.yang"

module ietf-ext-xponder-wdm-if {
    namespace "urn:ietf:params:xml:ns:yang:ietf-ext-xponder-wdm-if";
    prefix ietf-ext-xponder-wdm-if;

    import ietf-interfaces {
      prefix if;
    }

    organization
      "IETF CCAMP
      Working Group";

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


       Editor:   Dharini Hiremagalur
                 <mailto:dharinih@juniper.net>";

    description
      "This module contains a collection of YANG definitions for
       configuring Optical interfaces.

       Copyright (c) 2016 IETF Trust and the persons identified
       as authors of the code.  All rights reserved.

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

    revision "2016-03-17" {
         description
             "Initial revision.";
         reference
             "";
    }



       grouping opt-if-och-application-code  {
```

```
            description "Application code entity.";
            leaf application-code-id {
               type uint8 {
                     range "1..255";
               }
               description
                     "Id for the Application code";
            }
            leaf application-code-type {
               type uint8 {
                     range "0..1";
            }
              description
                     "Type for the Application code
                        0 - Standard, 1 - Proprietory
                        When the Type is Proprietory, then the
                        first 6 octets of the application-code
                        will be the OUI (organizationally unique
                        identifier)";

            }
            leaf application-code-length {
               type uint8 {
                     range "1..255";
               }
               description
                     "Number of octets in the Application code";

            }
            leaf application-code {
               type string {
                     length "1..255";
               }
               description "This parameter indicates the
                     transceiver application code at Ss and Rs as
                     defined in [ITU.G698.2] Chapter 5.3, that
                     is/should be used by this interface.
                     The optIfOChApplicationsCodeList has all the
                     application codes supported by this
                     interface.";

            }
         }

      typedef dbm-t {
        type decimal64 {
          fraction-digits 2;
          range "-50..-30 | -10..5 | 10000000";
```

```
        }
      description "
        Amplifier Power in dBm ";
    }
     grouping opt-if-och-application-code-list {
       description "List of Application codes group.";
       leaf number-application-codes-supported {
          type uint32;
          description "Number of Application codes
                      supported by this interface";
        }
       list application-code-list {
          key "application-code-id";
          description "List of the application codes";
          uses opt-if-och-application-code;
       }
     }


    grouping opt-if-och-power {
       description "Interface optical Power";
       leaf output-power {
          type int32;
          units ".01dbm";
          description "The output power for this interface in
                      .01 dBm.
                      The setting of the output power is
                       optional";
       }

       leaf input-power {
          type int32;
          units ".01dbm";
          config false;
          description "The current input power of this
                interface";
       }
    }

grouping channel-ITU {
description "channel-ITU";
container channel-t {
     description "wavelength notation according to RFC-6205";
      leaf grid {
        type uint32;
        description "grid type e.g.: 0=reserved, 1=DWDM, 2=CWDM";
      }
      leaf channel-spacing {
```

```
               type uint32;
               description "DWDM grid e.g.: 1=100GHz, 2=50GHz, 3=25GHz";
             }
            leaf identifier {
              type uint32;
              description "Channel identifier";
            }
            leaf n {
              type uint32;
              description "N Value (Channel n-m notation)";
            }
          }
        }

     grouping channel-flex {
       description "channel-flex";
       container channel-n-m {
         description "Channel N / M Notation to describe the
                      MEdiachannel";
          leaf grid {
            type uint32;
             description "grid type e.g.: 0=reserved, 1=DWDM, 2=CWDM";
            }
          leaf channel-spacing {
            type uint32;
             description "DWDM grid e.g.: 1=100GHz, 2=50GHz, 3=25GHz";
       }
          leaf n {
             type uint32;
              description "N Value (Channel n-m notation)";
       }
          leaf m {
            type uint32;
               description "M Value (Channel n-m notation)";
          }
        }
      }


     grouping feasibility-limit-list {
       list feasibility-limit {
         key "id";
         description "Feasibility limit power / osnr pair";
         leaf id {
           type uint32;
           description "Unique Identifier";
         }
         leaf power {
```

```
            type decimal64 {
              fraction-digits 2;
            }
            units "dB";
            description "Feasibility power";
          }
          leaf osnr {
            type decimal64 {
              fraction-digits 2;
            }
            description "Feasibility Signal / Noise";
          }
        }
        description "
          Ordered list of feasibility limits
          (should match order of supported FEC types
          given in fec-type-list).
        ";

      }

      grouping power-failure-low-alarm-grp {
        description "
          Optical Power failure alarm ";
          leaf power-failure-low {
          type dbm-t;
          units "dBm";
          default -1;
          description "Power Failure Low Value";
        }
      }

          grouping opt-if-och-central-frequency {
            description "Interface Central Frequency";
              leaf  central-frequency {
                type uint32;
                description "This parameter indicate This parameter
                        indicates the frequency of this interface ";

              }
          }


          notification opt-if-och-central-frequency-change {
            description "A change of Central Frequency has been
                        detected.";
            leaf "if-name" {
               type leafref {
```

```
                    path "/if:interfaces/if:interface/if:name";
                }
                description "Interface name";
           }
           container new-opt-if-och-central-frequency {
           description "The new Central Frequency of the
                       interface";
           uses opt-if-och-central-frequency;
           }
       }

       notification opt-if-och-application-code-change {
           description "A change of Application code has been
                       detected.";
           leaf "if-name" {
               type leafref {
                   path "/if:interfaces/if:interface/if:name";
               }
               description "Interface name";
           }
           container new-application-code {
              description "The new application code for the
                  interface";
              uses opt-if-och-application-code;
           }
       }


       augment "/if:interfaces/if:interface" {
         description "Parameters for an optical interface";
         container optIfOChRsSs {
            description "RsSs path configuration for an interface";
            container if-current-application-code {
                description "Current Application code of the
                            interface";
                uses opt-if-och-application-code;
            }

            container if-supported-application-codes {
                config false;
                description "Supported Application codes of
                            the interface";
                uses opt-if-och-application-code-list;
            }

            uses opt-if-och-power;
```

```
                        uses opt-if-och-central-frequency;

                }
            }
        }

    <CODE ENDS>
```

7.  Security Considerations

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

8.  IANA Considerations

    This document registers a URI in the IETF XML registry [RFC3688].
    Following the format in [RFC3688], the following registration is
    requested to be made:

    URI: urn:ietf:params:xml:ns:yang:ietf-interfaces:ietf-ext-xponder-
    wdm-if

    Registrant Contact: The IESG.

    XML: N/A, the requested URI is an XML namespace.

    This document registers a YANG module in the YANG Module Names
    registry [RFC6020].

    This document registers a YANG module in the YANG Module Names
    registry [RFC6020].

    prefix: ietf-ext-xponder-wdm-if reference: RFC XXXX

9.  Acknowledgements

    Gert Grammel is partly funded by European Union Seventh Framework
    Programme under grant agreement 318514 CONTENT.

10. Contributors

         Dean Bogdanovic
           Juniper Networks
           Westford
           U.S.A.
           email deanb@juniper.net

         Bernd Zeuner
           Deutsche Telekom
           Darmstadt
           Germany
           email B.Zeuner@telekom.de

         Arnold Mattheus
           Deutsche Telekom
           Darmstadt
           Germany
           email a.mattheus@telekom.de

         Manuel Paul
           Deutsche Telekom
           Berlin
           Germany
           email Manuel.Paul@telekom.de

         Walid Wakim
           Cisco
           9501 Technology Blvd
           ROSEMONT, ILLINOIS 60018
           UNITED STATES
           email wwakim@cisco.com

         Kam Lam
           Nokia
           USA
           +1 732 331 3476
           kam.lam@nokia.com

11. References

11.1. Normative References

   [RFC2863]  McCloghrie, K. and F. Kastenholz, "The Interfaces Group
              MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000,
              <http://www.rfc-editor.org/info/rfc2863>.

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

   [RFC2578]  McCloghrie, K., Ed., Perkins, D., Ed., and J.
              Schoenwaelder, Ed., "Structure of Management Information
              Version 2 (SMIv2)", STD 58, RFC 2578,
              DOI 10.17487/RFC2578, April 1999,
              <http://www.rfc-editor.org/info/rfc2578>.

   [RFC2579]  McCloghrie, K., Ed., Perkins, D., Ed., and J.
              Schoenwaelder, Ed., "Textual Conventions for SMIv2",
              STD 58, RFC 2579, DOI 10.17487/RFC2579, April 1999,
              <http://www.rfc-editor.org/info/rfc2579>.

   [RFC2580]  McCloghrie, K., Ed., Perkins, D., Ed., and J.
              Schoenwaelder, Ed., "Conformance Statements for SMIv2",
              STD 58, RFC 2580, DOI 10.17487/RFC2580, April 1999,
              <http://www.rfc-editor.org/info/rfc2580>.

   [RFC3591]  Lam, H-K., Stewart, M., and A. Huynh, "Definitions of
              Managed Objects for the Optical Interface Type", RFC 3591,
              DOI 10.17487/RFC3591, September 2003,
              <http://www.rfc-editor.org/info/rfc3591>.

   [RFC6205]  Otani, T., Ed. and D. Li, Ed., "Generalized Labels for
              Lambda-Switch-Capable (LSC) Label Switching Routers",
              RFC 6205, DOI 10.17487/RFC6205, March 2011,
              <http://www.rfc-editor.org/info/rfc6205>.

   [ITU.G698.2]
              International Telecommunications Union, "Amplified
              multichannel dense wavelength division multiplexing
              applications with single channel optical interfaces",
              ITU-T Recommendation G.698.2, November 2009.

   [ITU.G709]
              International Telecommunications Union, "Interface for the
              Optical Transport Network (OTN)", ITU-T Recommendation
              G.709, March 2003.

   [ITU.G872]
              International Telecommunications Union, "Architecture of
              optical transport networks", ITU-T Recommendation G.872,
              November 2001.

[ITU.G798]
          International Telecommunications Union, "Characteristics
          of optical transport network hierarchy equipment
          functional blocks", ITU-T Recommendation G.798, October
          2010.

[ITU.G874]
          International Telecommunications Union, "Management
          aspects of optical transport network elements",
          ITU-T Recommendation G.874, July 2010.

[ITU.G874.1]
          International Telecommunications Union, "Optical transport
          network (OTN): Protocol-neutral management information
          model for the network element view", ITU-T Recommendation
          G.874.1, January 2002.

[ITU.G959.1]
          International Telecommunications Union, "Optical transport
          network physical layer interfaces", ITU-T Recommendation
          G.959.1, November 2009.

[ITU.G826]
          International Telecommunications Union, "End-to-end error
          performance parameters and objectives for international,
          constant bit-rate digital paths and connections",
          ITU-T Recommendation G.826, November 2009.

[ITU.G8201]
          International Telecommunications Union, "Error performance
          parameters and objectives for multi-operator international
          paths within the Optical Transport Network (OTN)",
          ITU-T Recommendation G.8201, April 2011.

[ITU.G694.1]
          International Telecommunications Union, "Spectral grids
          for WDM applications: DWDM frequency grid",
          ITU-T Recommendation G.694.1, June 2002.

[ITU.G7710]
          International Telecommunications Union, "Common equipment
          management function requirements", ITU-T Recommendation
          G.7710, May 2008.

11.2.  Informative References

   [RFC3410]  Case, J., Mundy, R., Partain, D., and B. Stewart,
              "Introduction and Applicability Statements for Internet-
              Standard Management Framework", RFC 3410,
              DOI 10.17487/RFC3410, December 2002,
              <http://www.rfc-editor.org/info/rfc3410>.

   [RFC2629]  Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
              DOI 10.17487/RFC2629, June 1999,
              <http://www.rfc-editor.org/info/rfc2629>.

   [RFC4181]  Heard, C., Ed., "Guidelines for Authors and Reviewers of
              MIB Documents", BCP 111, RFC 4181, DOI 10.17487/RFC4181,
              September 2005, <http://www.rfc-editor.org/info/rfc4181>.

   [I-D.ietf-ccamp-dwdm-if-mng-ctrl-fwk]
              Kunze, R., Grammel, G., Beller, D., and G. Galimberti, "A
              framework for Management and Control of DWDM optical
              interface parameters", draft-ietf-ccamp-dwdm-if-mng-ctrl-
              fwk-00 (work in progress), April 2016.

   [RFC4054]  Strand, J., Ed. and A. Chiu, Ed., "Impairments and Other
              Constraints on Optical Layer Routing", RFC 4054,
              DOI 10.17487/RFC4054, May 2005,
              <http://www.rfc-editor.org/info/rfc4054>.

Appendix A.  Change Log

   This optional section should be removed before the internet draft is
   submitted to the IESG for publication as an RFC.

   Note to RFC Editor: please remove this appendix before publication as
   an RFC.

Appendix B.  Open Issues

   Note to RFC Editor: please remove this appendix before publication as
   an RFC.

Authors' Addresses

Gabriele Galimberti (editor)
Cisco
Via Santa Maria Molgora, 48 c
20871 - Vimercate
Italy

Phone: +390392091462
Email: ggalimbe@cisco.com


Ruediger Kunze
Deutsche Telekom
Dddd, xx
Berlin
Germany

Phone: +49xxxxxxxxxx
Email: RKunze@telekom.de


Dharini Hiremagalur (editor)
Juniper
1194 N Mathilda Avenue
Sunnyvale - 94089 California
USA

Email: dharinih@juniper.net


Gert Grammel (editor)
Juniper
Oskar-Schlemmer Str. 15
80807 Muenchen
Germany

Phone: +49 1725186386
Email: ggrammel@juniper.net

                        LDP Extensions for RMR
                  draft-esale-ldp-rmr-extensions-00

Abstract

   This document describes LDP extensions to signal Resilient MPLS Ring
   (RMR) Label Switched Paths (LSPs).  An RMR LSP is a multipoint to
   point LSP signaled using LDP (Label Distribution Protocol). RMR
   Architecture document - draft-ietf-mpls-rmr-02 - describes why and
   how MPLS should be used in ring topologies.

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

Table of Contents

1  Introduction

   This document describes LDP extensions to signal resilient MPLS ring
   (RMR) label switched paths (LSPs).  An RMR LSP is a multipoint to
   point LSP signaled using LDP (Label Distribution Protocol).
   Architecture document of RMR – draft-ietf-mpls-rmr-02 – describes why
   and how MPLS should be used in ring topologies.

   The ring is either auto-discovered or configured using IGP protocol
   such as OSPF or ISIS. IGP extensions for RMR will be described in a
   companion documents. After the ring discovery, each ring node acting
   as egress constructs and signals a clockwise (CW) and anti-clockwise
   (AC) ring FEC towards AC and CW direction respectively. Each transit
   node that receives the RMR FEC signals this LSP further in same
   direction using RMR link state database. In addition, it also adds a
   transit and ingress route for this LSP.  Once the signaling is
   complete, every node in a ring should have two counter rotating LSPs
   in CW and AC direction to reach every other node on the ring.


2.  Terminology

   A ring consists of a subset of n nodes {R_i, 0 <= i < n}. The
   direction from node R_i to R_i+1 is defined as as "clockwise" (CW)
   and the reverse direction is defined as "anti-clockwise" (AC). As
   there may be several rings in a graph, each ring is numbered with a
   distinct ring ID (RID).

   The following terminology is used for ring LSPs:

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

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

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

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

     Ring links:  Links that connect ring neighbors.

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

MP2P LSP:  Each LSP in the ring is a multipoint to point LSP such
    that LSP can have multiple ingress nodes and one egress node.


3.  Protocol extensions

   This section describes LDP extensions to signal RMR LSP in a ring.


3.1.  Ring LSP Capability

   RMR LSPs support for a LSR is advertised using LDP capabilities as
   defined in [RFC5561]. An implementation that supports the RMR
   procedures specified in this document MUST add the procedures
   pertaining to Capability Parameters for Initialization messages.

   A new optional capability parameter TLV, RMR Capability, is defined.
   Following is the format of the RMR Capability Parameter:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|U|F|  RMR Capability (TBD)       |         Length (= 1)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S| Reserved    |
+-+-+-+-+-+-+-+-+

    As described in [RFC5561]
    U: set to 1. Ignore, if not known.
    F: Set to 0. Do not forward.
    S: MUST be set to 1 to advertise the RMR Capability TLV.
```

   The RMR Capability TLV MUST be advertised in the LDP Initialization
   message.  If the peer has not advertised the RMR capability, then
   label messages pertaining to RMR FEC Element MUST not be sent to the
   peer.


3.2.  Ring FEC Element

   In order to setup RMR LSP in clockwise and anti-clockwise direction
   for every ring node, this document defines new protocol entity, the
   RMR FEC Element, to be used as a FEC Element in the FEC TLV.

   The RMR FEC Element consists of the ring address, ring identifier and
   ring flags which depicts ring direction.  The combination of ring
   address, ring identifier and ring flags uniquely identifies a ring

LSP within the MPLS network.

The RMR FEC Element value encoding is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    RMR(TBD)   |        Address Family         |    PreLen     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Ring Prefix                           |
~                                                              ~
|                                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Ring ID                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Ring Flags   |                  Reserved                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
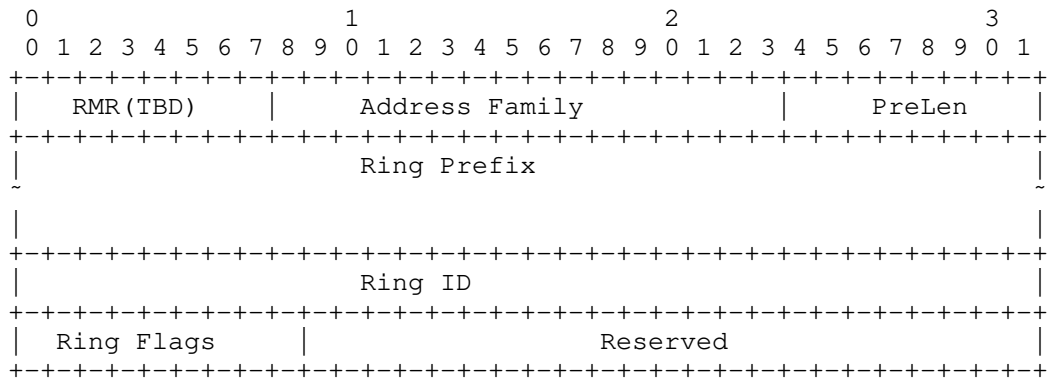
> FEC Type
>     One octet quantity containing a value from FEC Type
>     Name Space that encodes the fec type for a RMR LDP LSP.
>
> Address Family
>     Two octet quantity containing a value from ADDRESS FAMILY
>     NUMBERS in [ASSIGNED_AF] that encodes the address family for
>     the address prefix in the Prefix field.
>
> PreLen
>     One octet unsigned integer containing the length in bits of
>     the address prefix that follows.  A length of zero indicates
>     a prefix that matches all addresses (the default
>     destination); in this case, the Prefix itself is zero
>     octets).
>
> Prefix
>     An address prefix encoded according to the Address Family
>     field, whose length, in bits, was specified in the PreLen
>     field, padded to a byte boundary.
>
> Ring ID (RID)
>     A four-octet non-zero number that identifies a ring; this is
>     unique in some scope of a Service Provider's network.
>
> Ring Flags
>
>        0
>        0 1 2 3 4 5 6 7
>       +-+-+-+-+-+-+-+-+

```
                |RF | Reserved  |
                +-+-+-+-+-+-+-+-+
                The value of ring flags (RF) is defined as follows:
                1: Clockwise (CW) FEC
                2: Anti-clockwise (AC) FEC
```

4.  Ring Procedures

    Resilient MPLS ring architecture needs interaction between MPLS
    protocols such as LDP and RSVP and IGP to signal a RMR LSP.

4.1 Upstream LSR

    This section describes how to select a upstream LSR for RMR LSP.
    Consider MPLS rings as follows:

```
                        R0 . . . R1
                      .             .
                  R7   .  RID = 18 .  R2
                  |   .    .       .    .   |
        Anti-     |   .       R9 . . R8     .   |
        Clockwise v   .                   .  v Clockwise
                  R6       RID =17      R3
                    .                 .
                      R5 . . . R4
```
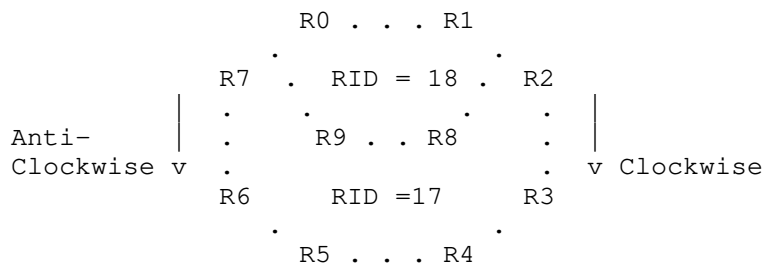
                  Figure 1: Two Rings with 10 nodes

    During the discovery of a MPLS ring, IGP populates its link state
    database with ring information. After the discovery, there are just
    two paths - one in clockwise direction and other in anti-clockwise
    direction - for every ring neighbor on a specific ring. For instance,
    the following table shows router R0's path for ring 17 and 18
    depicted in figure 1.

```
        +------------------------------+
        | RID    |CW neighbor|AC neighbor|
        +------------------------------+
        |  17    | R1        | R7        |
        +------------------------------+
        |  18    | R1        | R9        |
        +------------------------------+
```

                  Figure 2: R0's RMR upstream signaling table

    IGP informs LDP that a new MPLS ring, RID 17, is discovered. A LDP
    transit LSR uses this information to establish RMR LSPs. For

instance, suppose R5 receives a FEC with prefix R0, RID 17 and ring
flags AC. R5 knows that its clockwise path is R6 and anti-clockwise
path is R4 to reach R0 and that the label map arrived from router R4
for a anti-clockwise LSP. Therefore, R5 selects the upstream session
for this LSP as R6.


4.2  Ring Label Mapping Procedures

   The procedures in the subsequent sections are organized by the role
   that a node plays to establish a ring LSP. Each node is egress for
   its own prefixes and transit for every prefix received with a Label
   Mapping message. Every transit node is also a ingress for that LSP.

4.2.1  Definitions

   This section defines the notations for initiation, decoding,
   processing and propagation of RMR FEC Element.

   1. RMR FEC Element <P, R, C> or <P, R, A>: a FEC Element with egress
      prefix P, RID R and clockwise direction C or
      anti-clockwise direction A.
   2. RMR Label Mapping <P, R, C, L> or <P, R, A, L>: a Label Mapping
      message with a FEC TLV with a single RMR FEC Element <P, R, C> or
      <P, R, A> and Label TLV with label L. Label L MUST be allocated
      from the per-platform label space of the LSR sending the Label
      Mapping message. The use of the interface label space is outside
      the scope of this document.
   3. RMR Label Withdraw <P, R, C, L> or <P, R, A, L>: a Label Withdraw
      message with a FEC TLV with a single RMR FEC Element <P, R, C> or
      <P, R, A> and Label TLV with label L.
   4. RMR LSP <P, R, C>  or <P, R, A>: A RMR LSP with egress prefix P,
      Ring ID R and clockwise direction C or anti-clockwise direction A.

4.2.2  Preliminary

   A node X wishing to participate in LDP RMR signaling SHOULD negotiate
   the RMR capability with all its neighbors.  When the IGP informs X of
   its RMR neighbors A and C for RID R, it MUST check that A and C have
   also negotiated the RMR capability with X.  If these conditions are
   not satisfied, X cannot participate in signaling for ring R.  This
   applies for all roles that X may play: ingress, transit and egress.

4.2.3  Egress LSR

   Every ring node initiates two counter-rotating LSPs that egress on
   that node. After the IGP discovers the ring, LDP constructs the
   clockwise RMR FEC <P, R, C> and sends it in a Label Mapping message

to anti-clockwise neighbor. Similarly, LDP constructs a anti-
clockwise RMR FEC <P, R, A> and sends it in a Label Mapping message
to clockwise neighbor. This SHOULD establish a clockwise and anti-
clockwise LSP - in terms of data traffic - in the clockwise and anti-
clockwise direction respectively.

Furthermore, if a label other than implicit or explicit null is
advertised for a LSP, LDP SHOULD add a pop route for this label in
the Incoming Label Map (ILM) MPLS table.

When the node is no longer part of the ring, it SHOULD tear down its
egress LSPs - CW and AC - by sending a label withdraw message.

## 4.2.4  Ingress and Transit LSR

When a transit LSR R5 depicted in figure 1 receives a label map
message with RMR FEC Element <R0, 17, A, L1> from a downstream LDP
session to R4, it SHOULD verify that R4 is indeed its anticlockwise
neighbor for ring 17. If not, it SHOULD stop decoding the FEC TLV,
abort processing the message containing the TLV, send an "Unknown
FEC" Notification message to its LDP peer R4 signaling an error and
close the session.

If the LSR encounters no other error while parsing the RMR FEC
element, it allocates a Label L2 and determines a upstream LDP
session as R6 using the algorithm described in section 'Upstream
LSR'. It also programs a MPLS ILM table with label route L2 swapped
to L1 and Ingress tunnel table with prefix R0 with label push L1 on
all the LDP interfaces to R4, and sends the RMR FEC Element <R0, 17,
A, L2> to R6.

If a session to the anti-clockwise neighbor for RID 17 depicted in
Figure 2, namely R6, does not exist, the RMR FEC Element <R0, 17, A,
L2> SHOULD not be propagated further. Similarly, when the upstream
session changes because of ring topology change, transit LSR should
send a label withdraw for RMR FEC Element <R0, 17, A, L2> to older
upstream session R6 before sending Label Mapping message with RMR FEC
Element <R0, 17, A, L2> to a new upstream session.


## 4.3 Equal Cost Multipath (ECMP)

A transit and ingress LSR of RMR LSP uses all the links between
itself and downstream LSR to program transit and ingress route. Thus,
ECMP works automatically for a LDP RMR LSP. A vendor could provide
exception when necessary to this behavior by disabling certain ring
links for RMR LSPs.

4.4 Protection

   RMR uses the two counter-rotating LSPs to protect the other.  Say
   that R5 wants to protect the LSP to R0 for RID 17.  R5 receives RMR
   FEC Element <R0, 17, A, L1> from R4 and sends RMR FEC Element <R0,
   17, A, L2> to R6.  Then the primary path for the AC LSP is to swap L1
   with L2 with next hop R4.  Also, R5 receives RMR FEC Element <R0, 17,
   C, L3> from R6 and sends RMR FEC Element <R0, 17, C, L4> to R4.  The
   primary path for the CW LSP is to swap L3 with L4.  The protection
   path for the AC LSP is to swap L1 with L4 with next hop R6, thus
   sending the traffic back where it came from, but with a different
   label. The protection path for the CW LSP is to swap L3 with L2 with
   next hop R4.


5.  LSP Hierarchy


```
                        R9  R10  R11
                         .    .    .
                         .  .    .  .
                         .          .
                        R8  .  .  .  R9
                         .          .
                         .          .
                         .          .
                        R0  .  .  .  R1
                      .                 .
                  R7                       R2
       Anti-      │   .         Ring        .  │
       Clockwise  │   .                     .  │ Clockwise
               v  .        RID = 17      .  v
                  R6                       R3
                      .                 .
                        R5  .  .  .  R4
```
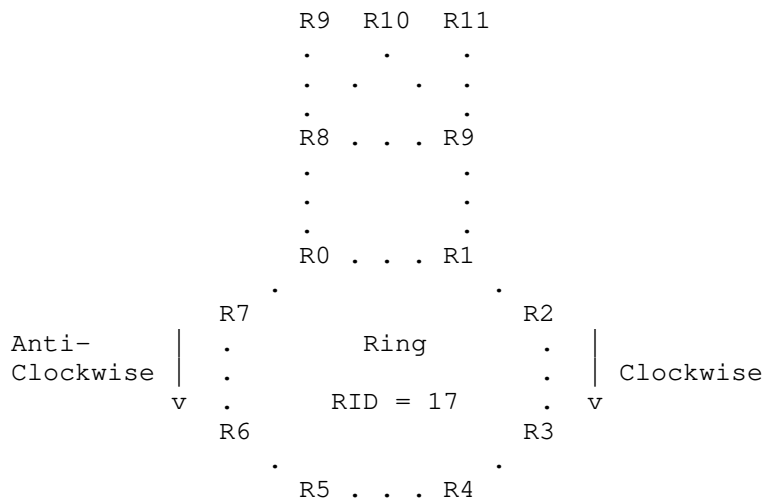
              Figure 3: Ring 17 with rest of the Network

   Suppose R5 needs to reach R10. Only RMR LSPs are setup inside the
   ring 17. Additionally, whenever services on R5 need to reach R10, R5
   dynamically establishes a tLDP session to ring 17 master node R0 and
   R1. Further, suppose it only learns IPv4 and IPv6 FECs only over this
   session using [draft-ietf-mpls-app-aware-tldp-05]. Thus, in order to
   reach R10, R5 uses top label as RMR LSP label to R0 or R1 and bottom
   label as R10's FEC label received over tLDP session of R0 or R1
   respectively.

6.  Ring LSPs

   An RMR LSP consists of two counter-rotating ring LSPs that start and
   end at the same node, say R1.  As such, this appears to cause a loop,
   something that is normally to be avoided by LDP [RSVP-TE].  There are
   some benefits to this.  Having a ring LSP allows the anchor node R1
   to ping itself and thus verify the end-to-end operation of the LSP.
   This, in conjunction with link-level OAM, offers a good indication of
   the operational state of the LSP.  [Also, having  R1 be the ingress
   means that R1 can initiate the Path messages for the two ring LSPs.
   This avoids R1 having to coordinate with its neighbors to signal the
   LSPs, and simplifies the case where a ring update changes R1's ring
   neighbors.]  The cost of this is a little more signaling and a couple
   more label entries in the LFIB.  However, we will let implementation
   guide us to the wisdom of this approach.

7.  Security Considerations

   The Capability and RMR FEC procedures described in this document will
   not introduce any change to LDP Security Considerations section
   described in [RFC5036].


8. IANA Considerations

   This document requires the assignment of a new code point for a
   Capability Parameter TLVs from the IANA managed LDP registry "TLV
   Type Name Space", corresponding to the advertisement of the RMR
   capability. IANA is requested to assign the lowest available value.

      Value   Description          Reference
      -----   ----------------     ---------
      TBD1    RMR capability       [this document]

   This document requires the assignment of a new code point for a FEC
   type from the IANA managed LDP registry "Forwarding Equivalence Class
   (FEC) Type Name Space". IANA is requested to assign the lowest
   available value.

      Value   Description          Reference
      -----   ----------------     ---------
      TBD1    RMR FEC type         [this document]


9. Acknowledgments

   TODO.


10.  Contributors

   Raveendra Torvi
   Juniper Networks
   10 Technology Park Dr
   Westford, MA  01886
   USA
   Email: rtorvi@juniper.net


   Ravi Singh
   Juniper Networks
   1133 Innovation Way
   Sunnyvale, CA  94089
   USA

    Email: ravis@juniper.net


    Abhishek Deshmukh
    Juniper Networks
    10 Technology Park Dr
    Westford, MA  01886
    USA
    Email: adeshmukh@juniper.net


11.  References

11.1  Normative References

    [I-D.ietf-mpls-rmr]  Kompella, K. and L. Contreras, "Resilient MPLS
              Rings", draft-ietf-mpls-rmr-02 (work in progress), July
              2016.

    [RFC5036]   Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed.,
                "LDP Specification", RFC 5036, October 2007,
                <http://www.rfc-editor.org/info/rfc5036>.

    [RFC5561]   Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL.
                Le Roux, "LDP Capabilities", RFC 5561, July 2009,
                <http://www.rfc-editor.org/info/rfc5561>.

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


11.2  Informative References


    [RFC6388]   IJ. Wijnands, I. Minei, K. Kompella, B. Thomas, "Label
                Distribution Protocol Extensions for Point-to-Multipoint
                and Multipoint-to-Multipoint Label Switched Paths", RFC
                6388, November 2011, <http://www.rfc-
                editor.org/info/rfc6388>

    [I-D.draft-deshmukh-rsvp-rmr-extension]  A. Deshmukh, K. Kompella,
                "RSVP Extensions for RMR", draft-deshmukh-rsvp-rmr-
                extension-00 (work in progress), July 2016.

    [I-D.draft-ietf-mpls-app-aware-tldp]  Santosh Esale, Raveendra Torvi,
                Luay Jalil, U. Chunduri, Kamran Raza, "Application-aware

Targeted LDP", draft-ietf-mpls-app-aware-tldp-05 (work in
progress), June 2016.

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


Authors' Addresses

   Santosh Esale
   Juniper Networks
   1133 Innovation Way
   Sunnyvale, CA  94089
   USA
   Email: sesale@juniper.net


   Kireeti Kompella
   Juniper Networks
   1133 Innovation Way
   Sunnyvale, CA  94089
   USA
   Email: kireeti@juniper.net

MPLS Working Group                                      Santosh Esale
INTERNET-DRAFT                                         Raveendra Torvi
Intended Status: Proposed Standard                   Juniper Networks

                                                          Luyuan Fang
                                                            Microsoft

                                                           Luay Jalil
                                                              Verizon

                                                     October 25, 2016

             Fast Reroute for Node Protection in LDP-based LSPs
                    draft-esale-mpls-ldp-node-frr-04

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

http://www.ietf.org/1id-abstracts.html

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

Table of Contents

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
   an 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 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 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

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

[RFC3209] D. Awduche, et al., "RSVP-TE: Extensions to RSVP for LSP
          Tunnels", RFC3209, Decembet 2001

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

[draft-ietf-mpls-app-aware-tldp] Esale, S., et al.,"Application-
          aware Targeted LDP", draft-esale-mpls-app-aware-tldp, work
          in progress

12. Informative References

[draft-ietf-mpls-mldp-node-protection], IJ. Wijnands, et al., "mLDP
          Node Protection", draft-ietf-mpls-mldp-node-protection,
          work in progress

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

MPLS Working Group                                        Santosh Esale
INTERNET-DRAFT                                          Raveendra Torvi
Intended Status: Proposed Standard                    Juniper Networks
Expires: September 14, 2017

                                                           Luyuan Fang
                                                             Microsoft

                                                            Luay Jalil
                                                               Verizon

                                                        March 13, 2017

              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

   http://www.ietf.org/1id-abstracts.html

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

Table of Contents

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
   an 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 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 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

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

[RFC3209] D. Awduche, et al., "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC3209, Decembet 2001.

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

[draft-ietf-mpls-app-aware-tldp] Esale, S., et al.,"Application-aware Targeted LDP", draft-esale-mpls-app-aware-tldp, work in progress.

12. Informative References

   [RFC7715], IJ. Wijnands, et al., "Multipoint LDP (mLDP) Node
              Protection", RFC7715, January 2016.
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

MPLS Working Group                                              K. Raza
Internet-Draft                                                 R. Asati
Intended status: Standards Track                   Cisco Systems, Inc.
Expires: February 19, 2017

                                                                X. Liu
                                                              Ericsson

                                                              S. Esale
                                                      Juniper Networks

                                                               X. Chen
                                                   Huawei Technologies

                                                               H. Shah
                                                     Ciena Corporation

                                                       August 18, 2016

                    YANG Data Model for MPLS LDP and mLDP
                      draft-ietf-mpls-ldp-mldp-yang-00

Abstract

   This document describes a YANG data model for Multi-Protocol Label
   Switching (MPLS) Label Distribution Protocol (LDP) and Multipoint LDP
   (mLDP).

Copyright Notice

Table of Contents

1.  Introduction

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

   This document introduces a YANG data model for MPLS Label
   Distribution Protocol (LDP) [RFC5036] and Multipoint LDP (mLDP)
   [RFC6388].  For LDP, it also covers LDP IPv6 [RFC7552] and LDP
   capabilities [RFC5561].

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

   o  Configuration

   o  Operational State

   o  Executables (Actions)

   o  Notifications

   This document is organized to define the data model for each of the
   above constructs (configuration, state, action, and notifications) in
   the sequence as listed earlier.  Given that mLDP is tightly coupled
   with LDP, mLDP data model is defined under LDP tree and in the same
   sequence as listed above.

2.  Specification of Requirements

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

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

3.  LDP YANG Model

3.1.  Overview

   This document defines a new module named "ietf-mpls-ldp" for LDP/mLDP
   data model where this module augments /rt:routing/rt:control-plane-
   protocols that is defined in [I-D.ietf-netmod-routing-cfg].

   There are four main containers in "ietf-mpls-ldp" module as follows:

   o  Read-Write parameters for configuration (Discussed in Section 3.2)

   o  Read-only parameters for operational state (Discussed in
      Section 3.3)

   o  Notifications for events (Discussed in Section 3.4)

   o  RPCs for executing commands to perform some action (Discussed in
      Section 3.5)

   For the configuration and state data, this model follows the similar
   approach described in [I-D.openconfig-netmod-opstate] to represent
   the configuration (intended state) and operational (applied and
   derived) state.  This means that for every configuration (rw) item,
   there is an associated (ro) item under "state" container to represent
   the applied state.  Furthermore, protocol derived state is also kept
   under "state" tree corresponding to the protocol area (discovery,
   peer etc.).  [Ed note: This document will be (re-)aligned with
   [I-D.openconfig-netmod-opstate] once that specification is adopted as
   a WG document]

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

```
module: ietf-mpls-ldp
    +-- rw routing
      +-- rw control-plane-protocols
        +-- rw mpls-ldp
            +-- rw global
            |   +-- rw config
            |       +-- rw ...
            |   +-- ro state
            |       +-- ro ...
            |       .
            +-- rw ...
            |
            |
            +-- rw ...
            ...

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

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

                        Figure 1

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

   o  This module aims to address only the core LDP/mLDP parameters as
      per RFC specification, as well as some widely used and deployed
      non-RFC features (such as label policies, session authentication
      etc).  Any vendor specific feature should be defined in a vendor-
      specific augmentation of this model.

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

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

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

hence we are treating the default forwarding table as just another
VRF.

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

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

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

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

o  The use of grouping (templates) for bundling and grouping the
   configuration items is not employed in current revision, and is a
   subject for consideration in future.

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

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

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

   *  Peer: An LDP session which has successfully progressed beyond
      its initialization phase and is either already exchanging the
      bindings or is ready to do so.

   It is to be noted that LDP Graceful Restart mechanisms defined in
   [RFC3478] allow keeping the exchanged bindings for some time after
   a session goes down with a peer.  We call such a state -- i.e.
   keeping peer bindings without established or recovered peering --
   a "stale" peer.  When used in this document, the above terms will
   refer strictly to the semantics and definitions defined for them.

A graphical representation of LDP YANG data model is presented in
Figure 3, Figure 5, Figure 11, and Figure 12.  Whereas, the actual
model definition in YANG is captured in Section 6.

While presenting the YANG tree view and actual .yang specification,
this document assumes the reader is familiar with the concepts of
YANG modeling, its presentation and its compilation.

3.2.  Configuration

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

This specification supports VRF-centric configuration.  For
implementations that support protocol-centric configuration, with
provision for inheritance and items that apply to all vrfs, we
recommend an augmentation of this model such that any protocol-
centric or all-vrf configuration is defined under their designated
containers within the standard network-instance (please see
Section 3.2.2)

This model augments /rt:routing/rt:control-plane-protocols that is
defined in [I-D.ietf-netmod-routing-cfg].  For LDP interfaces, this
model refers the MPLS interface as defined under MPLS base
specification [I-D.saad-mpls-base-yang].  Furthermore, as mentioned
earlier, the configuration tree presents read-write intended
configuration leave/items as well as read-only state of the applied
configuration.  The former is listed under "config" container and
latter under "state" container.

Following is high-level configuration organization for LDP/mLDP:

```
module: ietf-mpls-ldp
   +-- routing
     +-- control-plane-protocols
           +-- mpls-ldp
                 +-- global
                 |     +-- ...
                 |     +-- ...
                 |     +-- address-family* [afi]
                 |           +-- . . .
                 |           +-- . . .
                 |     +-- discovery
                 |           +-- . . .
                 +-- peers
                       +-- ...
                       +-- ...
```

                           Figure 2

   Given the configuration hierarchy, the model allows inheritance such
   that an item in a child tree is able to derive value from a similar
   or related item in one of the parent.  For instance, hello holdtime
   can be configured per-VRF or per-VRF-interface, thus allowing
   inheritance as well flexibility to override with a different value at
   any child level.

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

```
+--rw mpls-ldp!
   +--rw global
   |  +--rw config
   |  |  +--rw capability
   |  |  |  +--rw end-of-lib {capability-end-of-lib}?
   |  |  |  |  +--rw enable?   boolean
   |  |  |  +--rw typed-wildcard-fec {capability-typed-wildcard-fec}?
   |  |  |  |  +--rw enable?   boolean
   |  |  |  +--rw upstream-label-assignment {capability-upstream-label-assign
ment}?
   |  |  |        +--rw enable?   boolean
   |  |  +--rw graceful-restart
   |  |  |  +--rw enable?                 boolean
   |  |  |  +--rw helper-enable?          boolean {graceful-restart-helper-mod
e}?
   |  |  |  +--rw reconnect-time?         uint16
   |  |  |  +--rw recovery-time?          uint16
   |  |  |  +--rw forwarding-holdtime?    uint16
   |  |  +--rw igp-synchronization-delay?   uint16
   |  |  +--rw lsr-id?                      yang:dotted-quad
```

```
    │   +--rw address-family* [afi]
    │   │   +--rw afi        ldp-address-family
    │   │   +--rw config
    │   │      +--rw enable?         boolean
    │   │      +--rw label-policy
    │   │      │   +--rw independent-mode
    │   │      │   │   +--rw assign {policy-label-assignment-config}?
    │   │      │   │   │   +--rw (prefix-option)?
    │   │      │   │   │         │  +--rw prefix-list?        prefix-list-ref
    │   │      │   │   │            +--rw host-routes-only?    boolean
    │   │      │   │   +--rw advertise
    │   │      │   │   │   +--rw explicit-null
    │   │      │   │   │   │   +--rw enable?         boolean
    │   │      │   │   │   │   +--rw prefix-list?    prefix-list-ref
    │   │      │   │   │   +--rw prefix-list?      prefix-list-ref
    │   │      │   │   +--rw accept
    │   │      │   │      +--rw prefix-list?    prefix-list-ref
    │   │      │   +--rw ordered-mode {policy-ordered-label-config}?
    │   │      │      +--rw egress-lsr
    │   │      │      │   +--rw prefix-list?    prefix-list-ref
    │   │      │      +--rw advertise
    │   │      │      │   +--rw prefix-list?    prefix-list-ref
    │   │      │      +--rw accept
    │   │      │         +--rw prefix-list?    prefix-list-ref
    │   │      +--rw ipv4
    │   │      │   +--rw transport-address?   inet:ipv4-address
    │   │      +--rw ipv6
    │   │         +--rw transport-address?   inet:ipv6-address
    │   +--rw discovery
    │   │   +--rw interfaces
    │   │   │   +--rw config
    │   │   │   │   +--rw hello-holdtime?   uint16
    │   │   │   │   +--rw hello-interval?   uint16
    │   │   │   +--rw interface* [interface]
    │   │   │      +--rw interface         mpls-interface-ref
    │   │   │      +--rw config
    │   │   │      │   +--rw hello-holdtime?              uint16
    │   │   │      │   +--rw hello-interval?              uint16
    │   │   │      │   +--rw igp-synchronization-delay?   uint16 {per-interface-ti
mer-config}?
    │   │   │      +--rw address-family* [afi]
    │   │   │         +--rw afi        ldp-address-family
    │   │   │         +--rw config
    │   │   │            +--rw enable?   boolean
    │   │   │            +--rw ipv4
    │   │   │            │   +--rw transport-address?   union
    │   │   │            +--rw ipv6
    │   │   │               +--rw transport-address?   union
    │   │   +--rw targeted
```

```
│  │        +--rw config
│  │        │  +--rw hello-holdtime?   uint16
│  │        │  +--rw hello-interval?   uint16
│  │        │  +--rw hello-accept {policy-extended-discovery-config}?
│  │        │     +--rw enable?          boolean
│  │        │     +--rw neighbor-list?   neighbor-list-ref
│  │        +--rw address-family* [afi]
│  │           +--rw afi       ldp-address-family
│  │           +--rw ipv4
│  │           │  +--rw target* [adjacent-address]
│  │           │     +--rw adjacent-address    inet:ipv4-address
│  │           │     +--rw config
│  │           │        +--rw enable?          boolean
│  │           │        +--rw local-address?   inet:ipv4-address
│  │           +--rw ipv6
│  │              +--rw target* [adjacent-address]
│  │                 +--rw adjacent-address    inet:ipv6-address
│  │                 +--rw config
│  │                    +--rw enable?          boolean
│  │                    +--rw local-address?   inet:ipv6-address
│  +--rw forwarding-nexthop {forwarding-nexthop-config}?
│  │  +--rw interfaces
│  │     +--rw interface* [interface]
│  │        +--rw interface         mpls-interface-ref
│  │        +--rw address-family* [afi]
│  │           +--rw afi       ldp-address-family
│  │           +--rw config
│  │              +--rw ldp-disable?    boolean
│  +--rw label-policy
│     +--rw independent-mode
│     │  +--rw assign {policy-label-assignment-config}?
│     │  │  +--rw (prefix-option)?
│     │  │        +--rw prefix-list?        prefix-list-ref
│     │  │        +--rw host-routes-only?   boolean
│     │  +--rw advertise
│     │  │  +--rw explicit-null
│     │  │  │  +--rw enable?        boolean
│     │  │  │  +--rw prefix-list?   prefix-list-ref
│     │  │  +--rw prefix-list?      prefix-list-ref
│     │  +--rw accept
│     │     +--rw prefix-list?   prefix-list-ref
│     +--rw ordered-mode {policy-ordered-label-config}?
│        +--rw egress-lsr
│        │  +--rw prefix-list?   prefix-list-ref
│        +--rw advertise
│        │  +--rw prefix-list?   prefix-list-ref
│        +--rw accept
│           +--rw prefix-list?   prefix-list-ref
```

```
     +--rw peers
        +--rw config
        |  +--rw session-authentication-md5-password?   string
        |  +--rw session-ka-holdtime?                    uint16
        |  +--rw session-ka-interval?                    uint16
        |  +--rw session-downstream-on-demand {session-downstream-on-demand-con
fig}?
        |     +--rw enable?       boolean
        |     +--rw peer-list?    peer-list-ref
        +--rw peer* [lsr-id]
           +--rw lsr-id    yang:dotted-quad
           +--rw config
              +--rw admin-down?                           boolean
              +--rw capability
              +--rw label-policy
              |  +--rw advertise
              |  |  +--rw prefix-list?   prefix-list-ref
              |  +--rw accept
              |     +--rw prefix-list?   prefix-list-ref
              +--rw session-authentication-md5-password?   string
              +--rw graceful-restart
              |  +--rw enable?           boolean
              |  +--rw reconnect-time?   uint16
              |  +--rw recovery-time?    uint16
              +--rw session-ka-holdtime?                  uint16
              +--rw session-ka-interval?                  uint16
              +--rw address-family
                 +--rw ipv4
                 |  +--rw label-policy
                 |     +--rw advertise
                 |     |  +--rw prefix-list?   prefix-list-ref
                 |     +--rw accept
                 |        +--rw prefix-list?   prefix-list-ref
                 +--rw ipv6
                    +--rw label-policy
                       +--rw advertise
                       |  +--rw prefix-list?   prefix-list-ref
                       +--rw accept
                          +--rw prefix-list?   prefix-list-ref
```

Figure 3

3.2.1.  Configuration Hierarchy

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

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


Figure 4

   Following subsections briefly explain these configuration areas.

3.2.1.1.  Per-VRF parameters

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

3.2.1.1.1.  Per-VRF global parameters

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

3.2.1.1.2.  Per-VRF Capabilities parameters

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

container that is provided to override a capability that is enabled/
specified at VRF level.

### 3.2.1.1.3.  Per-VRF Per-Address-Family parameters

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

### 3.2.1.1.4.  Per-VRF Hello Discovery parameters

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

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

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

### 3.2.1.1.5.  Per-VRF Peer parameters

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

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

3.2.1.1.6.  Per-VRF Forwarding parameters

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

3.2.2.  All-VRFs Configuration

   [Ed note: TODO]

3.3.  Operational State

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

   Please note this state tree refers both the configuration "applied"
   state as well as the "derived" state related to the protocol.  [Ed
   note: This is where this model differs presently from
   [I-D.openconfig-netmod-opstate] and subject to alignment in later
   revisions]

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


```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
   +--rw mpls-ldp!
      +--rw global
      │  +--ro state
      │  │  +--ro capability
      │  │  │  +--ro end-of-lib {capability-end-of-lib}?
      │  │  │  │  +--ro enable?   boolean
      │  │  │  +--ro typed-wildcard-fec {capability-typed-wildcard-fec}?
      │  │  │  │  +--ro enable?   boolean
      │  │  │  +--ro upstream-label-assignment {capability-upstream-label-assign
ment}?
      │  │  │  │  +--ro enable?   boolean
      │  │  +--ro graceful-restart
      │  │  │  +--ro enable?                boolean
      │  │  │  +--ro helper-enable?         boolean {graceful-restart-helper-mod
e}?
      │  │  │  +--ro reconnect-time?        uint16
      │  │  │  +--ro recovery-time?         uint16
      │  │  │  +--ro forwarding-holdtime?   uint16
```

```
      │   │   +--ro igp-synchronization-delay?   uint16
      │   │   +--ro lsr-id?                      yang:dotted-quad
      │   +--rw address-family* [afi]
      │   │   +--rw afi        ldp-address-family
      │   │   +--ro state
      │   │      +--ro enable?        boolean
      │   │      +--ro label-policy
      │   │      │   +--ro independent-mode
      │   │      │   │   +--ro assign {policy-label-assignment-config}?
      │   │      │   │   │   +--ro (prefix-option)?
      │   │      │   │   │      +--:(prefix-list)
      │   │      │   │   │      │   +--ro prefix-list?        prefix-list-ref
      │   │      │   │   │      +--:(host-routes-only)
      │   │      │   │   │         +--ro host-routes-only?    boolean
      │   │      │   │   +--ro advertise
      │   │      │   │   │   +--ro explicit-null
      │   │      │   │   │   │   +--ro enable?        boolean
      │   │      │   │   │   │   +--ro prefix-list?   prefix-list-ref
      │   │      │   │   │   +--ro prefix-list?       prefix-list-ref
      │   │      │   │   +--ro accept
      │   │      │   │      +--ro prefix-list?        prefix-list-ref
      │   │      │   +--ro ordered-mode {policy-ordered-label-config}?
      │   │      │      +--ro egress-lsr
      │   │      │      │   +--ro prefix-list?   prefix-list-ref
      │   │      │      +--ro advertise
      │   │      │      │   +--ro prefix-list?   prefix-list-ref
      │   │      │      +--ro accept
      │   │      │         +--ro prefix-list?   prefix-list-ref
      │   │      +--ro ipv4
      │   │      │   +--ro transport-address?   inet:ipv4-address
      │   │      │   +--ro bindings
      │   │      │      +--ro address* [address]
      │   │      │      │   +--ro address         inet:ipv4-address
      │   │      │      │   +--ro advertisement-type?   advertised-received
      │   │      │      │   +--ro peer?         leafref
      │   │      │      +--ro fec-label* [fec]
      │   │      │         +--ro fec      inet:ipv4-prefix
      │   │      │         +--ro peer* [peer advertisement-type]
      │   │      │            +--ro peer                 leafref
      │   │      │            +--ro advertisement-type   advertised-received
      │   │      │            +--ro label?               mpls:mpls-label
      │   │      │            +--ro used-in-forwarding?   boolean
      │   │      +--ro ipv6
      │   │         +--ro transport-address?   inet:ipv6-address
      │   │         +--ro binding
      │   │            +--ro address* [address]
      │   │            │   +--ro address       inet:ipv6-address
      │   │            │   +--ro advertisement-type?   advertised-received
```

```
       │ │               │ +--ro peer?         leafref
       │ │               +--ro fec-label* [fec]
       │ │                  +--ro fec      inet:ipv6-prefix
       │ │                  +--ro peer* [peer advertisement-type]
       │ │                     +--ro peer                 leafref
       │ │                     +--ro advertisement-type   advertised-received
       │ │                     +--ro label?               mpls:mpls-label
       │ │                     +--ro used-in-forwarding?  boolean
       │ +--rw discovery
       │ │  +--rw interfaces
       │ │  │  +--ro state
       │ │  │     +--ro hello-holdtime?   uint16
       │ │  │     +--ro hello-interval?   uint16
       │ │  +--rw interface* [interface]
       │ │  │  +--ro state
       │ │  │  │  +--ro hello-holdtime?            uint16
       │ │  │  │  +--ro hello-interval?            uint16
       │ │  │  │  +--ro igp-synchronization-delay?  uint16 {per-interface-ti
mer-config}?
       │ │  │  │  +--ro next-hello?                uint16
       │ │  │  +--rw address-family* [afi]
       │ │  │     +--rw afi       ldp-address-family
       │ │  │     +--ro state
       │ │  │        +--ro enable?   boolean
       │ │  │        +--ro ipv4
       │ │  │        │  +--ro transport-address?  union
       │ │  │        │  +--ro hello-adjacencies* [adjacent-address]
       │ │  │        │     +--ro adjacent-address    inet:ipv4-address
       │ │  │        │     +--ro flag*               identityref
       │ │  │        │     +--ro hello-holdtime
       │ │  │        │     │  +--ro adjacent?    uint16
       │ │  │        │     │  +--ro negotiated?  uint16
       │ │  │        │     │  +--ro remaining?   uint16
       │ │  │        │     +--ro next-hello?         uint16
       │ │  │        │     +--ro statistics
       │ │  │        │     │  +--ro discontinuity-time   yang:date-and-time
       │ │  │        │     │  +--ro hello-received?      yang:counter64
       │ │  │        │     │  +--ro hello-dropped?       yang:counter64
       │ │  │        │     +--ro peer?               leafref
       │ │  │        +--ro ipv6
       │ │  │           +--ro transport-address?  union
       │ │  │           +--ro hello-adjacencies* [adjacent-address]
       │ │  │              +--ro adjacent-address    inet:ipv6-address
       │ │  │              +--ro flag*               identityref
       │ │  │              +--ro hello-holdtime
       │ │  │              │  +--ro adjacent?    uint16
       │ │  │              │  +--ro negotiated?  uint16
       │ │  │              │  +--ro remaining?   uint16
       │ │  │              +--ro next-hello?         uint16
```

```
│ │ │ │                        +--ro statistics
│ │ │ │                        │  +--ro discontinuity-time   yang:date-and-time
│ │ │ │                        │  +--ro hello-received?       yang:counter64
│ │ │ │                        │  +--ro hello-dropped?        yang:counter64
│ │ │ │                        +--ro peer?                leafref
│ │ │  +--rw targeted
│ │ │     +--ro state
│ │ │     │  +--ro hello-holdtime?   uint16
│ │ │     │  +--ro hello-interval?   uint16
│ │ │     │  +--ro hello-accept {policy-extended-discovery-config}?
│ │ │     │     +--ro enable?          boolean
│ │ │     │     +--ro neighbor-list?   neighbor-list-ref
│ │ │     +--rw address-family* [afi]
│ │ │        +--rw afi       ldp-address-family
│ │ │        +--ro state
│ │ │        │  +--ro ipv4
│ │ │        │  │  +--ro hello-adjacencies* [local-address adjacent-address
]
│ │ │        │  │        +--ro local-address       inet:ipv4-address
│ │ │        │  │        +--ro adjacent-address     inet:ipv4-address
│ │ │        │  │        +--ro flag*                identityref
│ │ │        │  │        +--ro hello-holdtime
│ │ │        │  │        │  +--ro adjacent?     uint16
│ │ │        │  │        │  +--ro negotiated?   uint16
│ │ │        │  │        │  +--ro remaining?    uint16
│ │ │        │  │        +--ro next-hello?          uint16
│ │ │        │  │        +--ro statistics
│ │ │        │  │        │  +--ro discontinuity-time   yang:date-and-time
│ │ │        │  │        │  +--ro hello-received?       yang:counter64
│ │ │        │  │        │  +--ro hello-dropped?        yang:counter64
│ │ │        │  │        +--ro peer?                leafref
│ │ │        │  +--ro ipv6
│ │ │        │     +--ro hello-adjacencies* [local-address adjacent-address
]
│ │ │        │           +--ro local-address       inet:ipv6-address
│ │ │        │           +--ro adjacent-address     inet:ipv6-address
│ │ │        │           +--ro flag*                identityref
│ │ │        │           +--ro hello-holdtime
│ │ │        │           │  +--ro adjacent?     uint16
│ │ │        │           │  +--ro negotiated?   uint16
│ │ │        │           │  +--ro remaining?    uint16
│ │ │        │           +--ro next-hello?          uint16
│ │ │        │           +--ro statistics
│ │ │        │           │  +--ro discontinuity-time   yang:date-and-time
│ │ │        │           │  +--ro hello-received?       yang:counter64
│ │ │        │           │  +--ro hello-dropped?        yang:counter64
│ │ │        │           +--ro peer?                leafref
│ │ │        +--rw ipv4
│ │ │        │  +--rw target* [adjacent-address]
│ │ │        │     +--rw adjacent-address     inet:ipv4-address
```

```
      |  |           |     +--ro state
      |  |           |        +--ro enable?          boolean
      |  |           |        +--ro local-address?   inet:ipv4-address
      |  |           +--rw ipv6
      |  |              +--rw target* [adjacent-address]
      |  |                 +--rw adjacent-address    inet:ipv6-address
      |  |                 +--ro state
      |  |                    +--ro enable?          boolean
      |  |                    +--ro local-address?   inet:ipv6-address
      |  +--rw forwarding-nexthop {forwarding-nexthop-config}?
      |  |  +--rw interfaces
      |  |     +--rw interface* [interface]
      |  |        +--rw interface          mpls-interface-ref
      |  |        +--rw address-family* [afi]
      |  |           +--rw afi        ldp-address-family
      |  |           +--ro state
      |  |              +--ro ldp-disable?    boolean
      +--rw peers
         +--ro state
         |  +--ro session-authentication-md5-password?   string
         |  +--ro session-ka-holdtime?                   uint16
         |  +--ro session-ka-interval?                   uint16
         |  +--ro session-downstream-on-demand {session-downstream-on-demand-con
fig}?
         |     +--ro enable?      boolean
         |     +--ro peer-list?   peer-list-ref
         +--rw peer* [lsr-id]
            +--rw lsr-id    yang:dotted-quad
            +--ro state
               +--ro admin-down?                         boolean
               +--ro capability
               +--ro label-policy
               |  +--ro advertise
               |  |  +--ro prefix-list?   prefix-list-ref
               |  +--ro accept
               |     +--ro prefix-list?   prefix-list-ref
               +--ro session-authentication-md5-password?   string
               +--ro graceful-restart
               |  +--ro enable?          boolean
               |  +--ro reconnect-time?   uint16
               |  +--ro recovery-time?    uint16
               +--ro session-ka-holdtime?                uint16
               +--ro session-ka-interval?                uint16
               +--ro address-family
               |  +--ro ipv4
               |  |  +--ro label-policy
               |  |  |  +--ro advertise
               |  |  |  |  +--ro prefix-list?   prefix-list-ref
               |  |  |  +--ro accept
```

```
            │  │  │     +--ro prefix-list?   prefix-list-ref
            │  │  +--ro hello-adjacencies* [local-address adjacent-address]
            │  │     +--ro local-address      inet:ipv4-address
            │  │     +--ro adjacent-address    inet:ipv4-address
            │  │     +--ro flag*               identityref
            │  │     +--ro hello-holdtime
            │  │     │  +--ro adjacent?     uint16
            │  │     │  +--ro negotiated?   uint16
            │  │     │  +--ro remaining?    uint16
            │  │     +--ro next-hello?         uint16
            │  │     +--ro statistics
            │  │     │  +--ro discontinuity-time   yang:date-and-time
            │  │     │  +--ro hello-received?      yang:counter64
            │  │     │  +--ro hello-dropped?       yang:counter64
            │  │     +--ro interface?         mpls-interface-ref
            │  +--ro ipv6
            │     +--ro label-policy
            │     │  +--ro advertise
            │     │  │  +--ro prefix-list?   prefix-list-ref
            │     │  +--ro accept
            │     │     +--ro prefix-list?   prefix-list-ref
            │     +--ro hello-adjacencies* [local-address adjacent-address]
            │        +--ro local-address      inet:ipv6-address
            │        +--ro adjacent-address    inet:ipv6-address
            │        +--ro flag*               identityref
            │        +--ro hello-holdtime
            │        │  +--ro adjacent?     uint16
            │        │  +--ro negotiated?   uint16
            │        │  +--ro remaining?    uint16
            │        +--ro next-hello?         uint16
            │        +--ro statistics
            │        │  +--ro discontinuity-time   yang:date-and-time
            │        │  +--ro hello-received?      yang:counter64
            │        │  +--ro hello-dropped?       yang:counter64
            │        +--ro interface?         mpls-interface-ref
            +--ro label-advertisement-mode
            │  +--ro local?        label-adv-mode
            │  +--ro peer?         label-adv-mode
            │  +--ro negotiated?   label-adv-mode
            +--ro next-keep-alive?                       uint16
            +--ro peer-ldp-id?                           yang:dotted-quad
            +--ro received-peer-state
            │  +--ro graceful-restart
            │  │  +--ro enable?          boolean
            │  │  +--ro reconnect-time?   uint16
            │  │  +--ro recovery-time?    uint16
            │  +--ro capability
            │     +--ro end-of-lib
```

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

                           Figure 5

3.3.1.  Derived States

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

      Neighbor Adjacencies

      Peer

      Bindings (FEC-label and address)

      Capabilities

3.3.1.1.  Adjacency state

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

   Following captures high level tree hierarchy for neighbor adjacency
   state.

```
   +--rw mpls-ldp!
      +--rw discovery
         +--rw interfaces
         |  +--rw interface* [interface]
         |     +--rw address-family* [af]
         |        +--ro state
         |           +--ro ipv4 (or ipv6)
         |              +--ro hello-adjacencies* [adjacent-address]
         |                 +--ro adjacent-address
         |                   . . . .
         |                   . . . .
         +--rw targeted
            +--rw address-family* [afi]
               +--rw afi      address-family
                  +--ro state
                     +--ro ipv4 (or ipv6)
                        +--ro hello-adjacencies* [local-address adjacent-addres
s]
                           +--ro local-address
                           +--ro adjacent-address
                             . . . .
                             . . . .
```

                              Figure 6

3.3.1.2.  Peer state

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

   Following captures high level tree hierarchy for peer state.

```
+--rw mpls-ldp!
   +--rw peers
      +--rw peer* [lsr-id]
         +--rw lsr-id
         +--ro state
            +--ro session-ka-holdtime?
            +-- . . . .
            +-- . . . .
            +--ro capability
            +  +ro -- . . .
            +--ro address-family
            |  +--ro ipv4 (or ipv6)
            |     +--ro hello-adjacencies* [local-address adjacent-address]
            |        . . . .
            |        . . . .
            +--ro received-peer-state
            |  +--ro . . . .
            |  +--ro capability
            |     +--ro . . . .
            +--ro statistics
               +-- . . . .
               +-- . . . .
```

Figure 7

3.3.1.3.  Bindings state

   Binding state provides information on LDP FEC-label bindings as well
   as address binding for both inbound (received) as well as outbound
   (advertised) direction.  FEC-label bindings are presented as a FEC-
   centric view, and address bindings are presented as an address-
   centric view:

```
    FEC-Label bindings:
        FEC 200.1.1.1/32:
          advertised: local-label 16000
            peer 192.168.0.2:0
            peer 192.168.0.3:0
            peer 192.168.0.4:0
          received:
            peer 192.168.0.2:0, label 16002, used-in-forwarding=Yes
            peer 192.168.0.3:0, label 17002, used-in-forwarding=No
        FEC 200.1.1.2/32:
          . . . .
        FEC 201.1.0.0/16:
          . . . .


    Address bindings:
        Addr 1.1.1.1:
          advertised
        Addr 1.1.1.2:
          advertised
        Addr 2.2.2.2:
          received, peer 192.168.0.2
        Addr 2.2.2.22:
          received, peer 192.168.0.2
        Addr 3.3.3.3:
          received, peer 192.168.0.3
        Addr 3.3.3.33:
          received, peer 192.168.0.3
```

                             Figure 8

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

   Following captures high level tree hierarchy for bindings state.

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

                              Figure 9

3.3.1.4.  Capabilities state

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

   Following captures high level tree hierarchy for LDP capabilities
   state.


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

                              Figure 10

3.4.  Notifications

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

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

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


                              Figure 11

3.5.  Actions

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

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

```
   module: ietf-mpls-ldp
   rpcs:
      +---x mpls-ldp-clear-peer
      │  +---w input
      │     +---w lsr-id?    union
      +---x mpls-ldp-clear-hello-adjacency
      │  +---w input
      │     +---w hello-adjacency
      │        +---w (hello-adjacency-type)?
      │           +--:(targeted)
      │           │  +---w targeted!
      │           │     +---w target-address?   inet:ip-address
      │           +--:(link)
      │              +---w link!
      │                 +---w next-hop-interface?   mpls-interface-ref
      │                 +---w next-hop-address?       inet:ip-address
      +---x mpls-ldp-clear-peer-statistics
         +---w input
            +---w lsr-id?    union
```

                              Figure 12

4.  mLDP YANG Model

4.1.  Overview

   Due to tight dependency of mLDP on LDP, mLDP model builds on top of
   LDP model defined earlier in the document.  Following are the main
   mLDP areas and documents that are within the scope of this model:

   o  mLDP Base Specification [RFC6388]

   o  mLDP Recursive FEC [RFC6512]

   o  Targeted mLDP [RFC7060]

   o  mLDP Fast-Reroute (FRR)

      *  Node Protection [RFC7715]

      *  Multicast-only

   o  Hub-and-Spoke Multipoint LSPs [RFC7140]

   o  mLDP In-band Signaling [RFC6826] (future revision)

   o  mLDP In-band signaling in a VRF [RFC7246]

   o  mLDP In-band Signaling with Wildcards [RFC7438] (future revision)

   o  Configured Leaf LSPs (manually provisioned)

   [Ed Note: Some of the topics in the above list are to be addressed/
   added in later revision of this document].

4.2.  Configuration

4.2.1.  Configuration Hierarchy

   In terms of overall configuration layout, following figure highlights
   extensions to LDP configuration model to incorporate mLDP:

```
+-- mpls-ldp
   +-- ...
   +-- ...
   +-- mldp
   |  +-- ...
   |  +-- ...
   |  +-- address-family* [af]
   |     +-- af
   |        +-- ...
   |        +-- ...
   +-- global
   |  +-- ...
   |  +-- capability
   |     +-- ...
   |     +-- ...
   |     +-- mldp
   |        +-- ...
   |        +-- ...
   +-- discovery
   |  +-- ...
   |  +-- ...
   +-- forwarding-nexthop
   |  +-- interfaces
   |     +-- interface* [interface]
   |        +-- interface
   |        +-- address-family* [af]
   |           +-- af
   |           +-- ...
   |           +-- mldp-disable
   +-- peers
      +-- ...
      +-- ...
      +-- peer* [lsr-id]
         +-- ...
         +-- ...
         +-- capability
            +-- ...
            +-- ...
            +-- mldp
               +-- ...
               +-- ...
```

                         Figure 13

   From above hierarchy, we can categorize mLDP configuration parameters
   into two types:

   o  Parameters that leverage/extend LDP containers and parameters

   o  Parameters that are mLDP specific

   Following subsections first describe mLDP specific configuration
   parameters, followed by those leveraging LDP.

4.2.2.  mldp container

   mldp container resides directly under "mpls-ldp" and holds the
   configuration related to items that are mLDP specific.  The main
   items under this container are:

   o  mLDP enabling: To enable mLDP under a (VRF) routing instance, mldp
      container is enabled under LDP.  Given that mLDP requires LDP
      signalling, it is not sensible to allow disabling LDP control
      plane under a (VRF) network-instance while requiring mLDP to be
      enabled for the same.  However, if a user wishes only to allow
      signalling for multipoint FECs on an LDP/mLDP enabled VRF
      instance, he/she can use LDP label-policies to disable unicast
      FECs under the VRF.

   o  mLDP per-AF features: mLDP manages its own list of IP address-
      families and the features enabled underneath.  The per-AF mLDP
      configuration items include:

      *  Multicast-only FRR: This enables Multicast-only FRR
         functionality for a given AF under mLDP.  The feature allows
         route-policy to be configured for finer control/applicability
         of the feature.

      *  Recursive FEC: The recursive-fec feature [RFC6512] can be
         enabled per AF with a route-policy.

      *  Configured Leaf LSPs: To provision multipoint leaf LSP
         manually, a container is provided per-AF under LDP.  The
         configuration is flexible and allows a user to specify MP LSPs
         of type p2mp or mp2mp with IPv4 or IPv6 root address(es) by
         using either LSP-Id or (S,G).

   Targeted mLDP feature specification [RFC7060] do not require any mLDP
   specific configuration.  It, however, requires LDP upstream-label-
   assignment capability [RFC6389] to be enabled.

4.2.3.  Leveraging LDP containers

   mLDP configuration model leverages following configuration areas and
   containers that are already defined for LDP:

   o  Capabilities: A new container "mldp" is defined under Capabilities
      container.  This new container specifies any mLDP specific
      capabilities and their parameters.  Moreover, a new "mldp"
      container is also added under per-peer capability container to
      override/control mLDP specific capabilities on a peer level.  In
      the scope of this document, the most important capabilities
      related to mLDP are p2mp, mp2mp, make-before-break, hub-and-spoke,
      and node-protection.

   o  Discovery and Peer: mLDP requires LDP discovery and peer
      procedures to form mLDP peering.  A peer is treated as mLDP peer
      only when either P2MP or MP2MP capabilities have been successfully
      exchanged with the peer.  If a user wish to selectively enable or
      disable mLDP with a LDP-enabled peer, he/she may use per-peer mLDP
      capabilities configuration.  [Ed Note: The option to control mLDP
      enabling/disabling on a peer-list is being explored for future ].
      In most common deployments, it is desirable to disable mLDP
      (capabilities announcements) on a targeted-only LDP peering, where
      targeted-only peer is the one whose discovery sources are targeted
      only.  In future revision, a configuration option for this support
      will also be provided.

   o  Forwarding: By default, mLDP is allowed to select any of the LDP
      enabled interface as a downstream interface towards a nexthop
      (LDP/mLDP peer) for MP LSP programming.  However, a configuration
      option is provided to allow mLDP to exclude a given interface from
      such a selection.  Note that such a configuration option will be
      useful only when there are more than one interfaces available for
      the downstream selection.

   This goes without saying that mLDP configuration tree follows the
   same approach as LDP, where the tree comprise leafs for intended
   configuration.

4.2.4.  YANG tree

   The following figure captures the YANG tree for mLDP configuration.
   To keep the focus, the figure has been simplified to display only
   mLDP items without any LDP items.

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
   +--rw mpls-ldp!
```

```
+--rw global
|  +--rw config
|  |  +--rw capability
|  |     +--rw mldp {mldp}?
|  |        +--rw p2mp
|  |        |  +--rw enable?   boolean
|  |        +--rw mp2mp
|  |        |  +--rw enable?   boolean
|  |        +--rw make-before-break
|  |        |  +--rw enable?              boolean
|  |        |  +--rw switchover-delay?    uint16
|  |        |  +--rw timeout?             uint16
|  |        +--rw hub-and-spoke {capability-mldp-hsmp}?
|  |        |  +--rw enable?   boolean
|  |        +--rw node-protection {capability-mldp-node-protection}?
|  |           +--rw plr?           boolean
|  |           +--rw merge-point
|  |              +--rw enable?                           boolean
|  |              +--rw targeted-session-teardown-delay?  uint16
|  +--rw mldp {mldp}?
|  |  +--rw config
|  |  |  +--rw enable?   boolean
|  |  +--rw address-family* [afi]
|  |     +--rw afi                    ldp-address-family
|  |     +--rw config
|  |     |  +--rw multicast-only-frr {mldp-mofrr}?
|  |     |  |  +--rw prefix-list?   prefix-list-ref
|  |     |  +--rw recursive-fec
|  |     |     +--rw prefix-list?   prefix-list-ref
|  |     +--rw configured-leaf-lsps
|  |        +--rw p2mp
|  |        |  +--rw roots-ipv4
|  |        |  |  +--rw root* [root-address]
|  |        |  |     +--rw root-address     inet:ipv4-address
|  |        |  |     +--rw lsp* [lsp-id source-address group-address]
|  |        |  |        +--rw lsp-id          uint16
|  |        |  |        +--rw source-address  inet:ipv4-address
|  |        |  |        +--rw group-address   inet:ipv4-address-no-zone
|  |        |  +--rw roots-ipv6
|  |        |     +--rw root* [root-address]
|  |        |        +--rw root-address     inet:ipv6-address
|  |        |        +--rw lsp* [lsp-id source-address group-address]
|  |        |           +--rw lsp-id          uint16
|  |        |           +--rw source-address  inet:ipv6-address
|  |        |           +--rw group-address   inet:ipv6-address-no-zone
|  |        +--rw mp2mp
|  |           +--rw roots-ipv4
|  |           |  +--rw root* [root-address]
```

```
│  │                 │     +--rw root-address      inet:ipv4-address
│  │                 │     +--rw lsp* [lsp-id source-address group-address]
│  │                 │        +--rw lsp-id           uint16
│  │                 │        +--rw source-address   inet:ipv4-address
│  │                 │        +--rw group-address    inet:ipv4-address-no-zone
│  │                 +--rw roots-ipv6
│  │                    +--rw root* [root-address]
│  │                       +--rw root-address      inet:ipv6-address
│  │                       +--rw lsp* [lsp-id source-address group-address]
│  │                          +--rw lsp-id           uint16
│  │                          +--rw source-address   inet:ipv6-address
│  │                          +--rw group-address    inet:ipv6-address-no-zone
│  +--rw forwarding-nexthop {forwarding-nexthop-config}?
│     +--rw interfaces
│        +--rw interface* [interface]
│           +--rw interface          if:interface-ref
│           +--rw address-family* [afi]
│              +--rw afi         address-family
│              +--rw config
│                 +--rw mldp-disable?   boolean {mldp}?
+--rw peers
   +--rw peer* [lsr-id]
         +--rw capability
            +--rw mldp {mldp}?
               +--rw p2mp
               │  +--rw enable?   boolean
               +--rw mp2mp
               │  +--rw enable?   boolean
               +--rw make-before-break
               │  +--rw enable?             boolean
               │  +--rw switchover-delay?   uint16
               │  +--rw timeout?            uint16
               +--rw hub-and-spoke {capability-mldp-hsmp}?
               │  +--rw enable?   boolean
                +--rw node-protection {capability-mldp-node-protection}?
                  +--rw plr?          boolean
                  +--rw merge-point
                     +--rw enable?                          boolean
                     +--rw targeted-session-teardown-delay?   uint16
```

Figure 14

## 4.3.  Operational State

Operational state of mLDP can be queried and obtained from this read-
only container "mldp" which resides under mpls-ldp container.

Please note this state tree refers both the configuration "applied" state as well as the "derived" state related to the mLDP protocol.

Following is a simplified graphical representation of the data model for mLDP operational state:

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
   +--rw mpls-ldp!
      +--rw global
      |  +--ro state
      |  |  +--ro capability
      |  |     +--ro mldp {mldp}?
      |  |        +--ro p2mp
      |  |        |  +--ro enable?   boolean
      |  |        +--ro mp2mp
      |  |        |  +--ro enable?   boolean
      |  |        +--ro make-before-break
      |  |        |  +--ro enable?               boolean
      |  |        |  +--ro switchover-delay?   uint16
      |  |        |  +--ro timeout?            uint16
      |  |        +--ro hub-and-spoke {capability-mldp-hsmp}?
      |  |        |  +--ro enable?   boolean
      |  |        +--ro node-protection {capability-mldp-node-protection}?
      |  |           +--ro plr?          boolean
      |  |           +--ro merge-point
      |  |              +--ro enable?                             boolean
      |  |              +--ro targeted-session-teardown-delay?   uint16
      |  |
      |  +--rw mldp {mldp}?
      |  |  +--ro state
      |  |  |  +--ro enable?   boolean
      |  |  +--rw address-family* [afi]
      |  |     +--rw afi                 ldp-address-family
      |  |     +--ro state
      |  |        +--ro multicast-only-frr {mldp-mofrr}?
      |  |        |  +--ro prefix-list?   prefix-list-ref
      |  |        +--ro recursive-fec
      |  |        |  +--ro prefix-list?   prefix-list-ref
      |  |        +--ro ipv4
      |  |        |  +--ro roots
      |  |        |     +--ro root* [root-address]
      |  |        |        +--ro root-address     inet:ipv4-address
      |  |        |        +--ro is-self?         boolean
      |  |        |        +--ro reachability* [address interface]
      |  |        |           +--ro address     inet:ipv4-address
      |  |        |           +--ro interface   mpls-interface-ref
```

```
   │  │           │      │      +--ro peer?         leafref
   │  │                +--ro bindings
   │  │                   +--ro opaque-type-lspid
   │  │           │      │  +--ro fec-label* [root-address lsp-id recur-root-addr
   ess recur-rd]
   │  │           │      │         +--ro root-address          inet:ipv4-address
   │  │           │      │         +--ro lsp-id                uint32
   │  │           │      │         +--ro recur-root-address    inet:ip-address
   │  │           │      │         +--ro recur-rd              route-distinguisher
   │  │           │      │         +--ro multipoint-type?      multipoint-type
   │  │           │      │         +--ro peer* [direction peer advertisement-type]
   │  │           │      │            +--ro direction           downstream-upstream
   │  │           │      │            +--ro peer                leafref
   │  │           │      │            +--ro advertisement-type  advertised-received
   │  │           │      │            +--ro label?              mpls:mpls-label
   │  │           │      │            +--ro mbb-role?           enumeration
   │  │           │      │            +--ro mofrr-role?         enumeration
   │  │           │      +--ro opaque-type-src
   │  │           │      │  +--ro fec-label* [root-address source-address group-a
   ddress rd recur-root-address recur-rd]
   │  │           │      │         +--ro root-address          inet:ipv4-address
   │  │           │      │         +--ro source-address        inet:ip-address
   │  │           │      │         +--ro group-address         inet:ip-address-no-zon
   e
   │  │           │      │         +--ro rd                    route-distinguisher
   │  │           │      │         +--ro recur-root-address    inet:ip-address
   │  │           │      │         +--ro recur-rd              route-distinguisher
   │  │           │      │         +--ro multipoint-type?      multipoint-type
   │  │           │      │         +--ro peer* [direction peer advertisement-type]
   │  │           │      │            +--ro direction           downstream-upstream
   │  │           │      │            +--ro peer                leafref
   │  │           │      │            +--ro advertisement-type  advertised-received
   │  │           │      │            +--ro label?              mpls:mpls-label
   │  │           │      │            +--ro mbb-role?           enumeration
   │  │           │      │            +--ro mofrr-role?         enumeration
   │  │           │      +--ro opaque-type-bidir
   │  │           │         +--ro fec-label* [root-address rp group-address rd re
   cur-root-address recur-rd]
   │  │           │            +--ro root-address          inet:ipv4-address
   │  │           │            +--ro rp                    inet:ip-address
   │  │           │            +--ro group-address         inet:ip-address-no-zon
   e
   │  │           │            +--ro rd                    route-distinguisher
   │  │           │            +--ro recur-root-address    inet:ip-address
   │  │           │            +--ro recur-rd              route-distinguisher
   │  │           │            +--ro multipoint-type?      multipoint-type
   │  │           │            +--ro peer* [direction peer advertisement-type]
   │  │           │               +--ro direction           downstream-upstream
   │  │           │               +--ro peer                leafref
   │  │           │               +--ro advertisement-type  advertised-received
   │  │           │               +--ro label?              mpls:mpls-label
   │  │           │               +--ro mbb-role?           enumeration
   │  │           │               +--ro mofrr-role?         enumeration
```

```
       │  │                  +--ro ipv6
       │  │                     +--ro roots
       │  │                     │  +--ro root* [root-address]
       │  │                     │     +--ro root-address    inet:ipv6-address
       │  │                     │     +--ro is-self?        boolean
       │  │                     │     +--ro reachability* [address interface]
       │  │                     │        +--ro address      inet:ipv6-address
       │  │                     │        +--ro interface    mpls-interface-ref
       │  │                     │        +--ro peer?        leafref
       │  │                     +--ro bindings
       │  │                        +--ro opaque-type-lspid
       │  │                        │  +--ro fec-label* [root-address lsp-id recur-root-addr
ess recur-rd]
       │  │                        │        +--ro root-address         inet:ipv6-address
       │  │                        │        +--ro lsp-id               uint32
       │  │                        │        +--ro recur-root-address   inet:ip-address
       │  │                        │        +--ro recur-rd             route-distinguisher
       │  │                        │        +--ro multipoint-type?     multipoint-type
       │  │                        │        +--ro peer* [direction peer advertisement-type]
       │  │                        │           +--ro direction           downstream-upstream
       │  │                        │           +--ro peer                leafref
       │  │                        │           +--ro advertisement-type  advertised-received
       │  │                        │           +--ro label?              mpls:mpls-label
       │  │                        │           +--ro mbb-role?            enumeration
       │  │                        │           +--ro mofrr-role?          enumeration
       │  │                        +--ro opaque-type-src
       │  │                        │  +--ro fec-label* [root-address source-address group-a
ddress rd recur-root-address recur-rd]
       │  │                        │        +--ro root-address         inet:ipv6-address
       │  │                        │        +--ro source-address       inet:ip-address
       │  │                        │        +--ro group-address        inet:ip-address-no-zon
e      │  │                        │        +--ro rd                   route-distinguisher
       │  │                        │        +--ro recur-root-address   inet:ip-address
       │  │                        │        +--ro recur-rd             route-distinguisher
       │  │                        │        +--ro multipoint-type?     multipoint-type
       │  │                        │        +--ro peer* [direction peer advertisement-type]
       │  │                        │           +--ro direction           downstream-upstream
       │  │                        │           +--ro peer                leafref
       │  │                        │           +--ro advertisement-type  advertised-received
       │  │                        │           +--ro label?              mpls:mpls-label
       │  │                        │           +--ro mbb-role?            enumeration
       │  │                        │           +--ro mofrr-role?          enumeration
       │  │                        +--ro opaque-type-bidir
       │  │                           +--ro fec-label* [root-address rp group-address rd re
cur-root-address recur-rd]
       │  │                                 +--ro root-address         inet:ipv6-address
       │  │                                 +--ro rp                   inet:ip-address
       │  │                                 +--ro group-address        inet:ip-address-no-zon
e      │  │                                 +--ro rd                   route-distinguisher
       │  │                                 +--ro recur-root-address   inet:ip-address
       │  │                                 +--ro recur-rd             route-distinguisher
```

```
│  │                            +--ro multipoint-type?       multipoint-type
│  │                            +--ro peer* [direction peer advertisement-type]
│  │                               +--ro direction             downstream-upstream
│  │                               +--ro peer                  leafref
│  │                               +--ro advertisement-type    advertised-received
│  │                               +--ro label?                mpls:mpls-label
│  │                               +--ro mbb-role?             enumeration
│  │                               +--ro mofrr-role?           enumeration
│  +--rw forwarding-nexthop {forwarding-nexthop-config}?
│     +--rw interfaces
│        +--rw interface* [interface]
│           +--rw address-family* [afi]
│              +--ro state
│                 +--ro mldp-disable?   boolean {mldp}?
+--rw peers
   +--rw peer* [lsr-id]
      +--ro state
         +--ro capability
         │  +--ro mldp {mldp}?
         │     +--ro p2mp
         │     │  +--ro enable?   boolean
         │     +--ro mp2mp
         │     │  +--ro enable?   boolean
         │     +--ro make-before-break
         │     │  +--ro enable?            boolean
         │     │  +--ro switchover-delay?  uint16
         │     │  +--ro timeout?           uint16
         │     +--ro hub-and-spoke {capability-mldp-hsmp}?
         │     │  +--ro enable?   boolean
         │     +--ro node-protection {capability-mldp-node-protection}?
         │        +--ro plr?         boolean
         │        +--ro merge-point
         │           +--ro enable?                            boolean
         │           +--ro targeted-session-teardown-delay?   uint16
         +--ro received-peer-state
            +--ro capability
               +--ro mldp {mldp}?
                  +--ro p2mp
                  │  +--ro enable?   boolean
                  +--ro mp2mp
                  │  +--ro enable?   boolean
                  +--ro make-before-break
                  │  +--ro enable?   boolean
                  +--ro hub-and-spoke
                  │  +--ro enable?   boolean
                  +--ro node-protection
                     +--ro plr?           boolean
                     +--ro merge-point?   boolean
```

                           Figure 15

4.3.1.  Derived states

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

   o  Root

   o  Bindings (FEC-label)

   o  Capabilities

4.3.1.1.  Root state

   Root address is a fundamental construct for MP FEC bindings and LSPs.
   The root state provides information on all the known roots in a given
   address-familty, and their information on the root reachability (as
   learnt from RIB).  In case of multi-path reachability to a root, the
   selection of upstream path is done on per-LSP basis at the time of
   LSP setup.  Similarly, when protection mechanisms like MBB or MoFRR
   are in place, the path designation as active/standby or primary/
   backup is also done on per LSP basis.  It is to be noted that a given
   root can be shared amongst multiple P2MP and/or MP2MP LSPs.
   Moreover, an LSP can be signaled to more than one root for RNR
   purposes.

   The following diagram illustrates a root database on a branch/transit
   LSR:

```
        root 1.1.1.1:
            path1:
                RIB: GigEthernet 1/0, 12.1.0.2;
                LDP: peer 192.168.0.1:0
            path2:
                RIB: GigEthernet 2/0, 12.2.0.2;
                LDP: peer 192.168.0.3:0

        root 2.2.2.2:
            path1:
                RIB: 3.3.3.3;             (NOTE: This is a recursive path)
                LDP: peer 192.168.0.3:0   (NOTE: T-mLDP peer)

         root 9.9.9.9:
            . . . .
```

Figure 16

   A root entry on a root LSR itself will be presented as follows:

```
        root 9.9.9.9:
            is-self
```

Figure 17

4.3.1.2.  Bindings state

   Binding state provides information on mLDP FEC-label bindings for
   both P2MP and MP2MP FEC types.  Like LDP, the FEC-label binding
   derived state is presented in a FEC-centric view per address-family,
   and provides information on both inbound (received) and outbound
   (advertised) bindings.  The FEC is presented as (root-address,
   opaque-type-data) and the direction (upstream or downstream) is
   picked with respect to root reachability.  In case of MBB or/and
   MoFRR, the role of a given peer binding is also provided with respect
   to MBB (active or standby) or/and MoFRR (primary or backup).

   This document covers following type of opaque values with their keys
   in the operational model of mLDP bindings:

```
+------------------------+------------------+------------+
| Opaque Type            | Key              | RFC        |
+------------------------+------------------+------------+
| Generic LSP Identifier | LSP Id           | [RFC6388]  |
| Transit IPv4 Source    | Source, Group    | [RFC6826]  |
| Transit IPv6 Source    | Source, Group    | [RFC6826]  |
| Transit IPv4 Bidir     | RP, Group        | [RFC6826]  |
| Transit IPv6 Bidir     | RP, Group        | [RFC6826]  |
| Transit VPNv4 Source   | Source, Group, RD| [RFC7246]  |
| Transit VPNv6 Source   | Source, Group, RD| [RFC7246]  |
| Transit VPNv4 Bidir    | RP, Group, RD    | [RFC7246]  |
| Transit VPNv6 Bidir    | RP, Group, RD    | [RFC7246]  |
| Recursive Opaque       | Root             | [RFC6512]  |
| VPN-Recursive Opaque   | Root, RD         | [RFC6512]  |
+------------------------+------------------+------------+
```

                   Table 1: MP Opaque Types and keys

   It is to be noted that there are three basic types (LSP Id, Source,
   and Bidir) and then there are variants (VPN, recursive, VPN-
   recursive) on top of these basic types.

   Following captures high level tree hierarchy for mLDP bindings state:

```
+--rw mpls-ldp!
    +--rw mldp
      +--rw address-family* [afi]
          +--rw afi        address-family
          +--ro state
             +--ro ipv4 (or ipv6)
                +--ro bindings
                   +--ro opaque-type-xxx [root-address, type-specific-key]
                      +--ro root-address
                      +--ro ...
                      +--ro recur-root-address    inet:ipv4-address
                      +--ro recur-rd              route-distinguisher
                      +--ro multipoint-type?      multipoint-type
                      +--ro peer* [direction peer advertisement-type]
                         +--ro direction            downstream-upstream
                         +--ro peer                 leafref
                         +--ro advertisement-type   advertised-received
                         +--ro label?               mpls:mpls-label
                         +--ro mbb-role?            enumeration
                         +--ro mofrr-role?          enumeration
```

                            Figure 18

In the above tree, the type-specific-key varies with the base type as listed in earlier Table 1.  For example, if the opaque type is Generic LSP Identifier, then the type-specific-key will be a uint32 value corresponding to the LSP.  Please see the complete model for all other types.

Moreover, the binding tree defines only three types of sub-trees (i.e. lspid, src, and bidir) which is able to map the respective variants (vpn, recursive, and vpn-recusrive) accordingly.  For example, the key for opaque-type-src is [R, S, G, rd, recur-R, recur-RD], where basic type will specify (R, S,G,-, -, -), VPN type will specify (R, S,G, rd, -, -), recursive type will specify [R, S,G, -, recur-R, -] and VPN-recursive type will specify [R, S,G, -, recur-R, recur-rd].

It is important to take note of the following:

o  The address-family ipv4/ipv4 applies to "root" address in the mLDP binding tree.  The other addresses (source, group, RP etc) do not have to be of the same address family type as the root.

o  The "recur-root-address" field applies to Recursive opaque type, and (recur-root-address, recur-rd) fields applies to VPN-Recursive opaque types as defined in [RFC6512]

o  In case of a recursive FEC, the address-family of the recur-root-address could be different than the address-family of the root address of original encapsulated MP FEC

The following diagram illustrates the FEC-label binding information structure for a P2MP (Transit IPv4 Source type) LSP on a branch/transit LSR:

```
        FEC (root 2.2.2.2, S=192.168.1.1, G=224.1.1.1):
          type: p2mp
          upstream:
            advertised:
              peer 192.168.0.1:0, label 16000 (local)
          downstream:
            received:
              peer 192.168.0.2:0, label 17000 (remote)
              peer 192.168.0.3:0, label 18000  (remote)
```

Figure 19

The following diagram illustrates the FEC-label binding information
structure for a similar MP2MP LSP on a branch/transit LSR:

```
        FEC (root 2.2.2.2, RP=192.168.9.9, G=224.1.1.1):
          type: mp2mp
          upstream:
            advertised:
              peer 192.168.0.1:0, label 16000 (local)
            received:
              peer 192.168.0.1:0, label 17000 (remote)
          downstream:
            advertised:
              peer 192.168.0.2:0, label 16001 (local), MBB role=active
              peer 192.168.0.3:0, label 16002 (local), MBB role=standby
            received:
              peer 192.168.0.2:0, label 17001 (remote)
              peer 192.168.0.3:0, label 18001 (remote)
```

Figure 20

4.3.1.3.  Capabilities state

Like LDP, mLDP capabilities state comprise two types of information -
global information and per-peer information.

4.4.  Notifications

mLDP notification module consists of notification related to changes
in the operational state of an mLDP FEC.  Following is a simplified
graphical representation of the data model for mLDP notifications:

```
   notifications:
     +---n mpls-mldp-fec-event
        +--ro event-type?        oper-status-event-type
        +--ro tree-type?         multipoint-type
        +--ro root?              inet:ip-address
        +--ro (lsp-key-type)?
           +--:(lsp-id-based)
           │  +--ro lsp-id?           uint16
           +--:(source-group-based)
              +--ro source-address?   inet:ip-address
              +--ro group-address?    inet:ip-address
```

Figure 21

4.5.  Actions

   Currently, no RPCs/actions are defined for mLDP.

5.  Open Items

   Following is a list of open items that are to be discussed and
   addressed in future revisions of this document:

   o  Close on augmentation off "mpls" list in "ietf-mpls" defined in
      [I-D.saad-mpls-base-yang]

   o  Align operational state modeling with other routing procols and
      [I-D.openconfig-netmod-opstate]

   o  Complete the section on Protocol-centric implementations and all-
      vrfs

   o  Specify default values for configuration parameters

   o  Revisit and cut down on the scope of the document and number of
      features it is trying to cover

   o  Split the model into a base and extended items

   o  Add statistics for mLDP root LSPs and bindings

   o  Extend the "Configured Leaf LSPs" for various type of opaque-types

   o  Extend mLDP notifications for other types of opaque values as well

   o  Close on single vs separate document for mLDP Yang

6.  YANG Specification

   Following are actual YANG definition for LDP and mLDP constructs
   defined earlier in the document.


 <CODE BEGINS> file "ietf-mpls-ldp@2016-07-08.yang" -->

 module ietf-mpls-ldp {
    namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp";
    // replace with IANA namespace when assigned
    prefix ldp;

    import ietf-inet-types {

```
     prefix "inet";
   }

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

   import ietf-interfaces {
     prefix "if";
   }

   import ietf-ip {
     prefix "ip";
   }

   import ietf-routing {
     prefix "rt";
   }

   import ietf-mpls {
     prefix "mpls";
   }

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

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

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

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

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

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

      Editor:   Xufeng Liu
                <mailto:xliu@kuatrotech.com>

      Editor:   Santosh Esale
```

```
               <mailto:sesale@juniper.net>

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

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

   description
     "This YANG module defines the essential components for the
      management of Multi-Protocol Label Switching (MPLS) Label
      Distribution Protocol (LDP) and Multipoint LDP (mLDP).";

   revision 2016-07-08 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: YANG Data Model for MPLS LDP and mLDP.";
   }

   /*
    * Features
    */

   feature admin-down-config {
     description
       "This feature indicates that the system allows to configure
        administrative down on a VRF instance and a peer.";
   }

   feature all-af-policy-config {
     description
       "This feature indicates that the system allows to configure
        policies that are applied to all address families.";
   }

   feature capability-end-of-lib {
     description
       "This feature indicates that the system allows to configure
        LDP end-of-lib capability.";
   }

   feature capability-mldp-hsmp {
     description
       "This feature indicates that the system allows to configure
        mLDP hub-and-spoke-multipoint capability.";
   }
```

```
   feature capability-mldp-node-protection {
     description
       "This feature indicates that the system allows to configure
        mLDP node-protection capability.";
   }

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

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

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

   feature global-session-authentication {
     description
       "This feature indicates that the system allows to configure
        authentication at global level.";
   }

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

   feature mldp {
     description
       "This feature indicates that the system supports Multicast
        LDP (mLDP).";
   }

   feature mldp-mofrr {
     description
       "This feature indicates that the system supports mLDP
        Multicast only FRR (MoFRR).";
   }
```

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

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

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

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

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

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

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

   /*
    * Typedefs
    */
   typedef ldp-address-family {
     type identityref {
```

```
        base rt:address-family;
      }
      description
        "LDP address family type.";
    }

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

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

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

    typedef label-adv-mode {
      type enumeration {
```

```
      enum downstream-unsolicited {
        description "Downstream Unsolicited.";
      }
      enum downstream-on-demand {
        description "Downstream on Demand.";
      }
    }
    description
      "Label Advertisement Mode.";
  }

  typedef mpls-interface-ref {
    type leafref {
      path "/rt:routing/mpls:mpls/mpls:interface/mpls:name";
    }
    description
      "This type is used by data models that need to reference
       mpls interfaces.";
  }

  typedef multipoint-type {
    type enumeration {
      enum p2mp {
        description "Point to multipoint.";
      }
      enum mp2mp {
        description "Multipoint to multipoint.";
      }
    }
    description
      "p2mp or mp2mp.";
  }

  typedef neighbor-list-ref {
    type string;
    description
      "A type for a reference to a neighbor list.";
  }

  typedef peer-list-ref {
    type string;
    description
      "A type for a reference to a peer list.";
  }

  typedef prefix-list-ref {
    type string;
    description
```

```
      "A type for a reference to a prefix list.";
  }

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

  typedef route-distinguisher {
    type string {
    }
    description
      "Type definition for route distinguisher.";
    reference
      "RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs).";
  }

  /*
   * Identities
   */
  identity adjacency-flag-base {
    description "Base type for adjacency flags.";
  }

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

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

  /*
```

```
 * Groupings
 */

grouping adjacency-state-attributes {
  description
    "Adjacency state attributes.";

  leaf-list flag {
    type identityref {
      base "adjacency-flag-base";
    }
    description "Adjacency flags.";
  }
  container hello-holdtime {
    description "Hello holdtime state.";
    leaf adjacent {
      type uint16;
      units seconds;
      description "Peer holdtime.";
    }
    leaf negotiated {
      type uint16;
      units seconds;
      description "Negotiated holdtime.";
    }
    leaf remaining {
      type uint16;
      units seconds;
      description "Remaining holdtime.";
    }
  }

  leaf next-hello {
    type uint16;
    units seconds;
    description "Time to send the next hello message.";
  }

  container statistics {
    description
      "Statistics objects.";

    leaf discontinuity-time {
      type yang:date-and-time;
      mandatory true;
      description
        "The time on the most recent occasion at which any one or
         more of this interface's counters suffered a
```

```
            discontinuity.  If no such discontinuities have occurred
            since the last re-initialization of the local management
            subsystem, then this node contains the time the local
            management subsystem re-initialized itself.";
        }

        leaf hello-received {
          type yang:counter64;
          description
            "The number of hello messages received.";
        }
        leaf hello-dropped {
          type yang:counter64;
          description
            "The number of hello messages received.";
        }
      } // statistics
    } // adjacency-state-attributes

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

    grouping binding-address-state-attributes {
      description
        "Address binding attributes";
      leaf advertisement-type {
        type advertised-received;
```

```
          description
            "Received or advertised.";
        }
        leaf peer {
          type leafref {
            path "../../../../../../peers/peer/lsr-id";
          }
          must "../advertisement-type = 'received'" {
            description
              "Applicable for received address.";
          }
          description
            "LDP peer from which this address is received.";
        } // peer
      } // binding-address-state-attributes

      grouping binding-label-state-attributes {
        description
          "Label binding attributes";
        list peer {
          key "peer advertisement-type";
          description
            "List of advertised and received peers.";
          leaf peer {
            type leafref {
              path "../../../../../../../peers/peer/lsr-id";
            }
            description
              "LDP peer from which this binding is received,
               or to which this binding is advertised.";
          }
          leaf advertisement-type {
            type advertised-received;
            description
              "Received or advertised.";
          }
          leaf label {
            type mpls:mpls-label;
            description
              "Advertised (outbound) or received (inbound)
               label.";
          }
          leaf used-in-forwarding {
            type boolean;
            description
              "'true' if the lable is used in forwarding.";
          }
        } // peer
```

```
    } // binding-label-state-attributes

    grouping extended-discovery-policy-attributes {
      description
        "LDP policy to control the acceptance of extended neighbor
         discovery hello messages.";
      container hello-accept {
        if-feature policy-extended-discovery-config;
        description
          "Extended discovery acceptance policies.";

        leaf enable {
          type boolean;
          description
            "'true' to accept; 'false' to deny.";
        }
        leaf neighbor-list {
          type neighbor-list-ref;
          description

            "The name of a peer ACL.";
        }
      } // hello-accept
    } // extended-discovery-policy-attributes

    grouping extended-discovery-timers {
      description
        "Extended discovery timer attributes.";
      leaf hello-holdtime {
        type uint16 {
          range 15..3600;
        }
        units seconds;
        description
          "The time interval for which LDP targeted Hello adjacency

           is maintained in the absence of targeted Hello messages
           from an LDP neighbor.";
      }
      leaf hello-interval {
        type uint16 {
          range 5..3600;
        }
        units seconds;
        description
          "The interval between consecutive LDP targeted Hello
           messages used in extended LDP discovery.";
      }
```

```
    } // extended-discovery-timers

    grouping global-attributes {
      description "Configuration attributes at global level.";

      uses instance-attributes;
    } // global-attributes

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

```
        range 30..3600;
      }
      units seconds;
      description
        "Specifies the time interval, in seconds, before the
         termination of the recovery phase.";
    }
  } // graceful-restart
} // graceful-restart-attributes

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

grouping instance-attributes {
  description "Configuration attributes at instance level.";
```

```
   container capability {
     description "Configure capability.";
     container end-of-lib {
       if-feature capability-end-of-lib;
       description
         "Configure end-of-lib capability.";
       leaf enable {
         type boolean;
         description
           "Enable end-of-lib capability.";
       }
     }
     container typed-wildcard-fec {
       if-feature capability-typed-wildcard-fec;
       description
         "Configure typed-wildcard-fec capability.";
       leaf enable {
         type boolean;
         description
           "Enable typed-wildcard-fec capability.";
       }
     }
     container upstream-label-assignment {
       if-feature capability-upstream-label-assignment;
       description
         "Configure upstream label assignment capability.";
       leaf enable {
         type boolean;
         description
           "Enable upstream label assignment.";
       }
     }
     container mldp {
       if-feature mldp;

       description
         "Multipoint capabilities.";
       uses mldp-capabilities;
     }
   } // capability

   uses graceful-restart-attributes;

   leaf igp-synchronization-delay {
     type uint16 {
       range 3..60;
     }
     units seconds;
```

```
        description
          "Sets the interval that the LDP waits before notifying the
           Interior Gateway Protocol (IGP) that label exchange is
           completed so that IGP can start advertising the normal
           metric for the link.";
      }
    leaf lsr-id {
      type yang:dotted-quad;
      description "Router ID.";
    }
  } // instance-attributes

  grouping ldp-adjacency-ref {
    description
      "An absolute reference to an LDP adjacency.";
    choice hello-adjacency-type {
      description
        "Interface or targeted adjacency.";
      case targeted {
        container targeted {
          description "Targeted adjacency.";
          leaf target-address {
            type inet:ip-address;
            description
              "The target address.";
          }
        } // targeted
      }
      case link {
        container link {
          description "Link adjacency.";
          leaf next-hop-interface {
            type mpls-interface-ref;
            description
              "Interface connecting to next-hop.";
          }
          leaf next-hop-address {
            type inet:ip-address;
            must "../next-hop-interface" {
              description
                "Applicable when interface is specified.";

            }
            description
              "IP address of next-hop.";
          }
        } // link
      }
```

```
      }
    } // ldp-adjacency-ref

    grouping ldp-fec-event {
      description
        "A LDP FEC event.";
      leaf prefix {
        type inet:ip-prefix;
        description
          "FEC.";
      }
    } // ldp-fec-event

    grouping ldp-peer-ref {
      description
        "An absolute reference to an LDP peer.";
      leaf peer-ref {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/mpls-ldp/"
            + "peers/peer/lsr-id";
        }
        description
          "Reference to an LDP peer.";
      }
    } // ldp-peer-ref

    grouping mldp-capabilities {
      description
        "mLDP capabilities.";
      container p2mp {
        description
          "Configure point-to-multipoint capability.";
        leaf enable {
          type boolean;
          description
            "Enable point-to-multipoint.";
        }
      }
      container mp2mp {
        description
          "Configure multipoint-to-multipoint capability.";
        leaf enable {
          type boolean;
          description
            "Enable multipoint-to-multipoint.";
        }
      }
      container make-before-break {
```

```
        description
          "Configure make-before-break capability.";
        leaf enable {
          type boolean;
          description
            "Enable make-before-break.";
        }
        leaf switchover-delay {
          type uint16;
          units seconds;
          description
            "Switchover delay in seconds.";
        }
        leaf timeout {
          type uint16;
          units seconds;
          description
            "Timeout in seconds.";
        }
      }
      container hub-and-spoke {
        if-feature capability-mldp-hsmp;
        description
          "Configure hub-and-spoke-multipoint capability.";
        reference
          "RFC7140: LDP Extensions for Hub and Spoke Multipoint
           Label Switched Path";
        leaf enable {
          type boolean;
          description
            "Enable hub-and-spoke-multipoint.";
        }
      }
      container node-protection {
        if-feature capability-mldp-node-protection;
        description
          "Configure node-protection capability.";
        reference
          "RFC7715: mLDP Node Protection.";
        leaf plr {
          type boolean;
          description
            "Point of Local Repair capable for MP LSP node
             protection.";
        }
        container merge-point {
          description
            "Merge Point capable for MP LSP node protection.";
```

```
        leaf enable {
          type boolean;
          description
            "Enable merge point capability.";
        }
        leaf targeted-session-teardown-delay {
          type uint16;
          units seconds;
          description
            "Targeted session teardown delay.";
        }
      } // merge-point
    }
  } // mldp-capabilities

  grouping  mldp-configured-lsp-roots {
    description
      "mLDP roots containers.";

    container roots-ipv4 {

      when "../../../af = 'ipv4'" {
        description
          "Only for IPv4.";
      }
      description
        "Configured IPv4 multicast LSPs.";
      list root {
        key "root-address";
        description
          "List of roots for configured multicast LSPs.";

        leaf root-address {
          type inet:ipv4-address;
          description
            "Root address.";
        }

        list lsp {
          must "(lsp-id = 0 and source-address != '0.0.0.0' and "
            + "group-address != '0.0.0.0') or "
            + "(lsp-id != 0 and source-address = '0.0.0.0' and "
            + "group-address = '0.0.0.0')" {
            description
              "A LSP can be identified by either <lsp-id> or
               <source-address, group-address>.";
          }
          key "lsp-id source-address group-address";
```

```
            description
              "List of LSPs.";
            leaf lsp-id {
              type uint16;
              description "ID to identify the LSP.";
            }
            leaf source-address {
              type inet:ipv4-address;
              description
                "Source address.";
            }
            leaf group-address {
              type inet:ipv4-address-no-zone;
              description
                "Group address.";
            }
          } // list lsp
        } // list root
      } // roots-ipv4

      container roots-ipv6 {

        when "../../../af = 'ipv6'" {
          description
            "Only for IPv6.";
        }
        description
          "Configured IPv6 multicast LSPs.";

        list root {
          key "root-address";
          description
            "List of roots for configured multicast LSPs.";

          leaf root-address {
            type inet:ipv6-address;
            description
              "Root address.";
          }

          list lsp {
            must "(lsp-id = 0 and source-address != '::' and "
              + "group-address != '::') or "
              + "(lsp-id != 0 and source-address = '::' and "
              + "group-address = '::')" {
              description
                "A LSP can be identified by either <lsp-id> or
                 <source-address, group-address>.";
```

```
            }
            key "lsp-id source-address group-address";
            description
              "List of LSPs.";
            leaf lsp-id {
              type uint16;
              description "ID to identify the LSP.";
            }
            leaf source-address {
              type inet:ipv6-address;
              description
                "Source address.";
            }
            leaf group-address {
              type inet:ipv6-address-no-zone;
              description
                "Group address.";
            }
          } // list lsp
        } // list root
      } // roots-ipv6
    } // mldp-configured-lsp-roots

    grouping mldp-fec-event {
      description
        "A mLDP FEC event.";
      leaf tree-type {
        type multipoint-type;
        description
          "p2mp or mp2mp.";
      }
      leaf root {
        type inet:ip-address;
        description
          "Root address.";
      }
      choice lsp-key-type {
        description
          "LSP ID based or source-group based .";
        case lsp-id-based {
          leaf lsp-id {
            type uint16;
            description
              "ID to identify the LSP.";
          }
        }
        case source-group-based {
          leaf source-address {
```

```
            type inet:ip-address;
            description

              "LSP source address.";
          }
          leaf group-address {
            type inet:ip-address;
            description
              "Multicast group address.";
          }
        } // case source-group-based
      }
    } // mldp-fec-event

    grouping  mldp-binding-label-state-attributes {
      description
        "mLDP label binding attributes.";

      leaf multipoint-type {
        type multipoint-type;
        description
          "The type of mutipoint, p2mp or mp2mp.";
      }
      list peer {
        key "direction peer advertisement-type";
        description
          "List of advertised and received peers.";
        leaf direction {
          type downstream-upstream;
          description
            "Downstream or upstream.";
        }
        leaf peer {
          type leafref {
            path
              "../../../../../../../../../peers/peer/lsr-id";
          }
          description
            "LDP peer from which this binding is received,
             or to which this binding is advertised.";
        }
        leaf advertisement-type {
          type advertised-received;
          description
            "Advertised or received.";
        }
        leaf label {
          type mpls:mpls-label;
```

```
           description
             "Advertised (outbound) or received (inbound) label.";
         }
         leaf mbb-role {
           when "../direction = 'upstream'" {
             description
               "For upstream.";
           }
           type enumeration {
             enum none {
               description "MBB is not enabled.";
             }
             enum active {
               description "This LSP is active.";
             }
             enum inactive {
               description "This LSP is inactive.";
             }
           }
           description
             "The MBB status of this LSP.";
         }
         leaf mofrr-role {
           when "../direction = 'upstream'" {
             description
               "For upstream.";
           }
           type enumeration {
             enum none {
               description "MOFRR is not enabled.";
             }
             enum primary {
               description "This LSP is primary.";
             }
             enum backup {
               description "This LSP is backup.";
             }
           }
           description
             "The MOFRR status of this LSP.";
         }
       } // peer
     } // mldp-binding-label-state-attributes

   grouping peer-af-policy-container {
     description
       "LDP policy attribute container under peer address-family.";
     container label-policy {
```

```
        description
          "Label policy attributes.";
        container advertise {
          description
            "Label advertising policies.";
          leaf prefix-list {
            type prefix-list-ref;
              description
                "Applies the prefix list to outgoing label
                 advertisements.";
          }
        }
        container accept {
          description
            "Label advertisement acceptance policies.";
          leaf prefix-list {
            type prefix-list-ref;
            description
              "Applies the prefix list to incoming label
               advertisements.";
          }
        } // accept
      } // label-policy
    } // peer-af-policy-container

  grouping peer-attributes {
    description "Peer configuration attributes.";

    leaf session-ka-holdtime {
      type uint16 {
        range 45..3600;
      }
      units seconds;
      description
        "The time interval after which an inactive LDP session
         terminates and the corresponding TCP session closes.
         Inactivity is defined as not receiving LDP packets from the
         peer.";
    }
    leaf session-ka-interval {
      type uint16 {
        range 15..1200;
      }
      units seconds;
      description
        "The interval between successive transmissions of keepalive
         packets. Keepalive packets are only sent in the absence of
         other LDP packets transmitted over the LDP session.";
```

```
      }
    } // peer-attributes

    grouping peer-authentication {
      description
        "Peer authentication attributes.";
      leaf session-authentication-md5-password {
        type string {
          length "1..80";
        }
        description
          "Assigns an encrypted MD5 password to an LDP
           peer";
      } // md5-password
    } // peer-authentication

    grouping peer-state-derived {
      description "Peer derived state attributes.";

      container label-advertisement-mode {
        description "Label advertisement mode state.";
        leaf local {
          type label-adv-mode;
          description
            "Local Label Advertisement Mode.";
        }
        leaf peer {
          type label-adv-mode;
          description
            "Peer Label Advertisement Mode.";
        }
        leaf negotiated {
          type label-adv-mode;
          description
            "Negotiated Label Advertisement Mode.";
        }
      }
      leaf next-keep-alive {
        type uint16;
        units seconds;
        description "Time to send the next KeepAlive message.";
      }

      leaf peer-ldp-id {
        type yang:dotted-quad;
        description "Peer LDP ID.";
      }
```

```
      container received-peer-state {
        description "Peer features.";

        uses graceful-restart-attributes-per-peer;

        container capability {
          description "Configure capability.";
          container end-of-lib {
            description
              "Configure end-of-lib capability.";
            leaf enable {
              type boolean;
              description
                "Enable end-of-lib capability.";
            }
          }
          container typed-wildcard-fec {
            description
              "Configure typed-wildcard-fec capability.";
            leaf enable {
              type boolean;
              description
                "Enable typed-wildcard-fec capability.";
            }
          }
          container upstream-label-assignment {
            description
              "Configure upstream label assignment capability.";
            leaf enable {
              type boolean;
              description
                "Enable upstream label assignment.";
            }
          }
          container mldp {
            if-feature mldp;
            description
              "Multipoint capabilities.";

            container p2mp {
              description
                "Configure point-to-multipoint capability.";
              leaf enable {
                type boolean;
                description
                  "Enable point-to-multipoint.";
              }
            }
```

```
            container mp2mp {
              description
                "Configure multipoint-to-multipoint capability.";
              leaf enable {
                type boolean;
                description
                  "Enable multipoint-to-multipoint.";
              }
            }
            container make-before-break {
              description
                "Configure make-before-break capability.";
              leaf enable {
                type boolean;
                description
                  "Enable make-before-break.";
              }
            }
            container hub-and-spoke {
              description
                "Configure hub-and-spoke-multipoint capability.";
              reference
                "RFC7140: LDP Extensions for Hub and Spoke Multipoint
                 Label Switched Path";
              leaf enable {
                type boolean;
                description
                  "Enable hub-and-spoke-multipoint.";
              }
            }
            container node-protection {
              description
                "Configure node-protection capability.";
              reference
                "RFC7715: mLDP Node Protection.";
              leaf plr {
                type boolean;
                description
                  "Point of Local Repair capable for MP LSP node
                   protection.";
              }
              leaf merge-point {
                type boolean;
                description
                  "Merge Point capable for MP LSP node protection.";
              } // merge-point
            } // node-protection
          } // mldp
```

```
        } // capability
      } // received-peer-state

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

      leaf session-state {
        type enumeration {
          enum non-existent {
            description "NON EXISTENT state. Transport disconnected.";
          }
          enum initialized {
            description "INITIALIZED state.";
          }
          enum openrec {
            description "OPENREC state.";
          }
          enum opensent {
            description "OPENSENT state.";
          }
          enum operational {
            description "OPERATIONAL state.";
          }
        }
        description
          "Representing the operational status.";
      }

      container tcp-connection {
        description "TCP connection state.";
        leaf local-address {
          type inet:ip-address;
```

```
        description "Local address.";
      }
      leaf local-port {
        type inet:port-number;
        description "Local port.";
      }
      leaf remote-address {
        type inet:ip-address;
        description "Remote address.";
      }
      leaf remote-port {
        type inet:port-number;
        description "Remote port.";
      }
    } // tcp-connection

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

    container statistics {
      description
        "Statistics objects.";

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

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

      leaf total-addresses {
        type uint32;
```

```
            description
              "The number of learned addresses.";
          }
        leaf total-labels {
          type uint32;
          description
            "The number of learned labels.";
        }
        leaf total-fec-label-bindings {
          type uint32;
          description
            "The number of learned label-address bindings.";
        }
    } // statistics
  } // peer-state-derived

  grouping policy-container {
    description
      "LDP policy attributes.";
    container label-policy {
      description
        "Label policy attributes.";
      container independent-mode {
        description
          "Independent label policy attributes.";
        container assign {

          if-feature policy-label-assignment-config;
          description
            "Label assignment policies";
          choice prefix-option {
            description
              "Use either prefix-list or host-routes-only.";
            case prefix-list {
              leaf prefix-list {
                type prefix-list-ref;
                description
                  "Assign labels according to certain prefixes.";
              }
            }
            case host-routes-only {
              leaf host-routes-only {
                type boolean;
                description
                  "'true' to apply host routes only.";
              }
            }
          } // prefix-option
```

```
            }
          container advertise {
            description
              "Label advertising policies.";
            container explicit-null {
              description
                "Enables an egress router to advertise an
                 explicit null label (value 0) in place of an
                 implicit null label (value 3) to the
                 penultimate hop router.";
              leaf enable {
                type boolean;
                description
                  "'true' to enable explicit null.";
              }
              leaf prefix-list {
                type prefix-list-ref;
                description
                  "Prefix list name. Applies the filters in the
                   specified prefix list to label
                   advertisements.
                   If the prefix list is not specified, explicit
                   null label advertisement is enabled for all
                   directly connected prefixes.";
              }
            }
            leaf prefix-list {
              type prefix-list-ref;
              description
                "Applies the prefix list to outgoing label
                 advertisements.";

            }
          }
          container accept {
            description
              "Label advertisement acceptance policies.";
            leaf prefix-list {
              type prefix-list-ref;
              description
                "Applies the prefix list to incoming label
                 advertisements.";
            }
          }
        } // independent-mode
        container ordered-mode {
          if-feature policy-ordered-label-config;
          description
```

```
            "Ordered label policy attributes.";
          container egress-lsr {
            description
              "Egress LSR label assignment policies";
            leaf prefix-list {
              type prefix-list-ref;
              description
                "Assign labels according to certain prefixes.";
            }
          }
          container advertise {
            description
              "Label advertising policies.";
            leaf prefix-list {
              type prefix-list-ref;
              description
                "Applies the prefix list to outgoing label
                 advertisements.";
            }
          }
          container accept {
            description
              "Label advertisement acceptance policies.";
            leaf prefix-list {
              type prefix-list-ref;
              description
                "Applies the prefix list to incoming label
                 advertisements.";
            }
          }
        } // ordered-mode
      } // label-policy
    } // policy-container

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

```
      type yang:counter64;
      description
        "The number of address messages sent or received.";
    }
    leaf address-withdraw {
      type yang:counter64;
      description
        "The number of address-withdraw messages sent or received.";
    }
    leaf initialization {
      type yang:counter64;
      description
        "The number of initialization messages sent or received.";
    }
    leaf keepalive {
      type yang:counter64;
      description
        "The number of keepalive messages sent or received.";
    }
    leaf label-abort-request {
      type yang:counter64;
      description
        "The number of label-abort-request messages sent or
         received.";
    }
    leaf label-mapping {
      type yang:counter64;
      description
        "The number of label-mapping messages sent or received.";
    }
    leaf label-release {
      type yang:counter64;
      description
        "The number of label-release messages sent or received.";
    }
    leaf label-request {
      type yang:counter64;
      description
        "The number of label-request messages sent or received.";
    }
    leaf label-withdraw {
      type yang:counter64;
      description
        "The number of label-withdraw messages sent or received.";
    }
    leaf notification {
      type yang:counter64;
      description
```

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

    /*
     * Configuration data nodes
     */

    augment "/rt:routing/rt:control-plane-protocols" {
      description "LDP augmentation.";

      container mpls-ldp {
        presence "Container for LDP protocol.";
        description
          "Container for LDP protocol.";

        container global {
          description
            "Global attributes for LDP.";
          container config {
            description
              "Configuration data.";
            uses global-attributes;
          }
          container state {
            config false;
            description
              "Operational state data.";
            uses global-attributes;
          }

          container mldp {
            if-feature mldp;
            description
              "mLDP attributes at per instance level. Defining
               attributes here does not enable any MP capabilities.
               MP capabilities need to be explicitly enabled under
               container capability.";

            container config {
              description
                "Configuration data.";
              leaf enable {
                type boolean;
                description
                  "Enable mLDP.";
              }
            }
```

```
            container state {
              config false;
              description

                "Operational state data.";
              leaf enable {
                type boolean;
                description
                  "Enable mLDP.";
              }
            }

            list address-family {
              key "afi";
              description
                "Per-af params.";
              leaf afi {
                type ldp-address-family;
                description
                  "Address family type value.";
              }

              container config {
                description
                  "Configuration data.";
                container multicast-only-frr {
                  if-feature mldp-mofrr;
                  description
                    "Multicast only FRR (MoFRR) policy.";
                  leaf prefix-list {
                    type prefix-list-ref;
                    description
                      "Enables MoFRR for the specified access list.";
                  }
                } // multicast-only-frr
                container recursive-fec {
                  description
                    "Recursive FEC policy.";
                  leaf prefix-list {
                    type prefix-list-ref;
                    description
                      "Enables recursive FEC for the specified access
                       list.";
                  }
                } // recursive-for
              }
              container state {
                config false;
```

```
                description
                  "Operational state data.";
                container multicast-only-frr {
                  if-feature mldp-mofrr;

                  description
                    "Multicast only FRR (MoFRR) policy.";
                  leaf prefix-list {
                    type prefix-list-ref;
                    description
                      "Enables MoFRR for the specified access list.";
                  }
                } // multicast-only-frr
                container recursive-fec {
                  description
                    "Recursive FEC policy.";
                  leaf prefix-list {
                    type prefix-list-ref;
                    description
                      "Enables recursive FEC for the specified access
                       list.";
                  }
                } // recursive-fec

                container ipv4 {
                  when "../../afi = 'ipv4'" {
                    description
                      "Only for IPv4.";
                  }
                  description
                    "IPv4 state information.";
                  container roots {
                    description
                      "IPv4 multicast LSP roots.";
                    list root {
                      key "root-address";
                      description
                        "List of roots for configured multicast LSPs.";

                      leaf root-address {
                        type inet:ipv4-address;
                        description
                          "Root address.";
                      }

                      leaf is-self {
                        type boolean;
                        description
```

```
                          "This is the root.";
                      }

                      list reachability {
                        key "address interface";
                        description
                          "A next hop for reachability to root,
                           as a RIB view.";
                        leaf address {
                          type inet:ipv4-address;
                          description
                            "The next hop address to reach root.";
                        }
                        leaf interface {
                          type mpls-interface-ref;
                          description
                            "Interface connecting to next-hop.";
                        }
                        leaf peer {
                          type leafref {
                            path
                              "../../../../../../../../peers/peer/"
                              + "lsr-id";
                          }
                          description
                            "LDP peer from which this next hop can be
                             reached.";
                        }
                      }
                    } // list root
                  } // roots
                  container bindings {
                    description
                      "mLDP FEC to label bindings.";
                    container opaque-type-lspid {
                      description
                        "The type of opaque value element is
                         the generic LSP identifier";
                      reference
                        "RFC6388: Label Distribution Protocol
                         Extensions for Point-to-Multipoint and
                         Multipoint-to-Multipoint Label Switched
                         Paths.";
                      list fec-label {
                        key
                          "root-address lsp-id "
                          + "recur-root-address recur-rd";
                        description
```

```
                          "List of FEC to label bindings.";
                        leaf root-address {
                          type inet:ipv4-address;
                          description
                            "Root address.";
                        }
                        leaf lsp-id {
                          type uint32;
                          description "ID to identify the LSP.";
                        }
                        leaf recur-root-address {
                          type inet:ip-address;
                          description
                            "Recursive root address.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        leaf recur-rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher in the VPN-Recursive
                             Opaque Value.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        uses mldp-binding-label-state-attributes;
                      } // fec-label
                    } // opaque-type-lspid

                  container opaque-type-src {
                    description
                      "The type of opaque value element is
                       the transit source TLV";
                    reference
                      "RFC6826: Multipoint LDP In-Band Signaling for
                       Point-to-Multipoint and
                       Multipoint-to-Multipoint Label Switched
                       Paths.";
                    list fec-label {
                      key
                        "root-address source-address group-address "
                        + "rd recur-root-address recur-rd";
                      description
                        "List of FEC to label bindings.";
                      leaf root-address {
                        type inet:ipv4-address;
```

```
                        description
                          "Root address.";
                      }
                      leaf source-address {
                        type inet:ip-address;
                        description
                          "Source address.";
                      }
                      leaf group-address {
                        type inet:ip-address-no-zone;
                        description
                          "Group address.";
                      }
                      leaf rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher.";
                        reference
                          "RFC7246: Multipoint Label Distribution
                           Protocol In-Band Signaling in a Virtual
                           Routing and Forwarding (VRF) Table
                           Context.";
                      }
                      leaf recur-root-address {
                        type inet:ip-address;
                        description
                          "Recursive root address.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      leaf recur-rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher in the VPN-Recursive
                           Opaque Value.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      uses mldp-binding-label-state-attributes;
                    } // fec-label
                  } // opaque-type-src

                  container opaque-type-bidir {
                    description
                      "The type of opaque value element is
                       the generic LSP identifier";
```

```
                        reference
                          "RFC6826: Multipoint LDP In-Band Signaling for
                           Point-to-Multipoint and
                           Multipoint-to-Multipoint Label Switched
                           Paths.";
                        list fec-label {
                          key
                            "root-address rp group-address "
                            + "rd recur-root-address recur-rd";
                          description
                            "List of FEC to label bindings.";
                          leaf root-address {
                            type inet:ipv4-address;
                            description
                              "Root address.";
                          }
                          leaf rp {
                            type inet:ip-address;
                            description
                              "RP address.";
                          }
                          leaf group-address {
                            type inet:ip-address-no-zone;
                            description
                              "Group address.";
                          }
                          leaf rd {
                            type route-distinguisher;
                            description
                              "Route Distinguisher.";
                            reference
                              "RFC7246: Multipoint Label Distribution
                               Protocol In-Band Signaling in a Virtual
                               Routing and Forwarding (VRF) Table
                               Context.";
                          }
                          leaf recur-root-address {
                            type inet:ip-address;
                            description
                              "Recursive root address.";
                            reference
                              "RFC6512: Using Multipoint LDP When the
                               Backbone Has No Route to the Root";
                          }
                          leaf recur-rd {
                            type route-distinguisher;
                            description
                              "Route Distinguisher in the VPN-Recursive
```

```
                       Opaque Value.";
                   reference
                     "RFC6512: Using Multipoint LDP When the
                      Backbone Has No Route to the Root";
                 }
               uses mldp-binding-label-state-attributes;
             } // fec-label
           } // opaque-type-bidir
         } // bindings
       } // ipv4

       container ipv6 {
         when "../../afi = 'ipv6'" {
           description
             "Only for IPv6.";
         }
         description
           "IPv6 state information.";
         container roots {
           description
             "IPv6 multicast LSP roots.";
           list root {
             key "root-address";
             description
               "List of roots for configured multicast LSPs.";

             leaf root-address {
               type inet:ipv6-address;
               description
                 "Root address.";
             }

             leaf is-self {
               type boolean;
               description
                 "This is the root.";
             }

             list reachability {
               key "address interface";
               description
                 "A next hop for reachability to root,
                  as a RIB view.";
               leaf address {
                 type inet:ipv6-address;
                 description
                   "The next hop address to reach root.";
               }
```

```
                    leaf interface {
                      type mpls-interface-ref;
                      description
                        "Interface connecting to next-hop.";
                    }
                    leaf peer {
                      type leafref {
                        path
                          "../../../../../../../../../peers/peer/"
                          + "lsr-id";
                      }
                      description
                        "LDP peer from which this next hop can be
                         reached.";
                    }
                  }
                } // list root
              } // roots
              container bindings {
                description
                  "mLDP FEC to label bindings.";
                container opaque-type-lspid {
                  description
                    "The type of opaque value element is
                     the generic LSP identifier";
                  reference
                    "RFC6388: Label Distribution Protocol
                     Extensions for Point-to-Multipoint and
                     Multipoint-to-Multipoint Label Switched
                     Paths.";
                  list fec-label {
                    key
                      "root-address lsp-id "
                      + "recur-root-address recur-rd";
                    description
                      "List of FEC to label bindings.";
                    leaf root-address {
                      type inet:ipv6-address;
                      description
                        "Root address.";
                    }
                    leaf lsp-id {
                      type uint32;
                      description "ID to identify the LSP.";
                    }
                    leaf recur-root-address {
                      type inet:ip-address;
                      description
```

```
                                  "Recursive root address.";
                                reference
                                  "RFC6512: Using Multipoint LDP When the
                                   Backbone Has No Route to the Root";
                              }
                            leaf recur-rd {
                              type route-distinguisher;
                              description
                                "Route Distinguisher in the VPN-Recursive
                                 Opaque Value.";
                              reference
                                "RFC6512: Using Multipoint LDP When the
                                 Backbone Has No Route to the Root";
                            }
                            uses mldp-binding-label-state-attributes;
                          } // fec-label
                        } // opaque-type-lspid

                        container opaque-type-src {
                          description
                            "The type of opaque value element is
                             the transit Source TLV";
                          reference
                            "RFC6826: Multipoint LDP In-Band Signaling for
                             Point-to-Multipoint and
                             Multipoint-to-Multipoint Label Switched
                             Paths.";
                          list fec-label {
                            key
                              "root-address source-address group-address "
                              + "rd recur-root-address recur-rd";
                            description
                              "List of FEC to label bindings.";
                            leaf root-address {
                              type inet:ipv6-address;
                              description
                                "Root address.";
                            }
                            leaf source-address {
                              type inet:ip-address;
                              description
                                "Source address.";
                            }
                            leaf group-address {
                              type inet:ip-address-no-zone;
                              description
                                "Group address.";
                            }
```

```
                        leaf rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher.";
                          reference
                            "RFC7246: Multipoint Label Distribution
                             Protocol In-Band Signaling in a Virtual
                             Routing and Forwarding (VRF) Table
                             Context.";
                        }
                        leaf recur-root-address {
                          type inet:ip-address;
                          description
                            "Recursive root address.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        leaf recur-rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher in the VPN-Recursive
                             Opaque Value.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        uses mldp-binding-label-state-attributes;
                      } // fec-label
                    } // opaque-type-src

                    container opaque-type-bidir {
                      description
                        "The type of opaque value element is
                         the generic LSP identifier";
                      reference
                        "RFC6826: Multipoint LDP In-Band Signaling for
                         Point-to-Multipoint and
                         Multipoint-to-Multipoint Label Switched
                         Paths.";
                      list fec-label {
                        key
                          "root-address rp group-address "
                          + "rd recur-root-address recur-rd";
                        description
                          "List of FEC to label bindings.";
                        leaf root-address {
                          type inet:ipv6-address;
```

```
                          description
                            "Root address.";
                        }
                        leaf rp {
                          type inet:ip-address;
                          description
                            "RP address.";
                        }
                        leaf group-address {
                          type inet:ip-address-no-zone;
                          description
                            "Group address.";
                        }
                        leaf rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher.";
                          reference
                            "RFC7246: Multipoint Label Distribution
                             Protocol In-Band Signaling in a Virtual
                             Routing and Forwarding (VRF) Table
                             Context.";
                        }
                        leaf recur-root-address {
                          type inet:ip-address;
                          description
                            "Recursive root address.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        leaf recur-rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher in the VPN-Recursive
                             Opaque Value.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        uses mldp-binding-label-state-attributes;
                      } // fec-label
                    } // opaque-type-bidir
                  } // bindings
                } // ipv6
              } // state

              container configured-leaf-lsps {
```

```
                  description
                    "Configured multicast LSPs.";

                  container p2mp {
                    description
                      "Configured point-to-multipoint LSPs.";
                    uses mldp-configured-lsp-roots;
                  }
                  container mp2mp {
                    description
                      "Configured multipoint-to-multipoint LSPs.";
                    uses mldp-configured-lsp-roots;
                  }
                } // configured-leaf-lsps
              } // list address-family
            } // mldp

            list address-family {
              key "afi";
              description
                "Per-vrf per-af params.";
              leaf afi {
                type ldp-address-family;
                description
                  "Address family type value.";
              }

              container config {
                description
                  "Configuration data.";
                leaf enable {
                  type boolean;
                  description
                    "'true' to enable the address family.";
                }
                uses policy-container;

                container ipv4 {
                  when "../../afi = 'ipv4'" {
                    description
                      "Only for IPv4.";
                  }
                  description
                    "IPv4 address family.";
                  leaf transport-address {
                    type inet:ipv4-address;
                    description
                      "The transport address advertised in LDP Hello
```

```
                    messages.";
                }
              } // ipv4
              container ipv6 {
                when "../../afi = 'ipv6'" {
                  description
                    "Only for IPv6.";
                }
                description
                  "IPv6 address family.";
                leaf transport-address {
                  type inet:ipv6-address;
                  description
                    "The transport address advertised in LDP Hello
                    messages.";
                }
              } // ipv6
            }
            container state {
              config false;
              description
                "Operational state data.";
              leaf enable {
                type boolean;
                description
                  "'true' to enable the address family.";
              }

              uses policy-container;

              container ipv4 {
                when "../../afi = 'ipv4'" {
                  description
                    "Only for IPv4.";
                }
                description
                  "IPv4 address family.";
                leaf transport-address {
                  type inet:ipv4-address;
                  description
                    "The transport address advertised in LDP Hello
                    messages.";
                }

                container bindings {
                  description
                    "LDP address and label binding information.";
                  list address {
```

```
                    key "address";
                    description
                      "List of address bindings.";
                    leaf address {
                      type inet:ipv4-address;
                      description
                        "Binding address.";
                    }
                    uses binding-address-state-attributes;
                  } // binding-address

                  list fec-label {
                    key "fec";
                    description
                      "List of label bindings.";
                    leaf fec {
                      type inet:ipv4-prefix;
                      description
                        "Prefix FEC.";
                    }
                    uses binding-label-state-attributes;
                  } // fec-label
                } // binding
              } // ipv4
              container ipv6 {
                when "../../afi = 'ipv6'" {
                  description
                    "Only for IPv6.";
                }
                description
                  "IPv6 address family.";
                leaf transport-address {
                  type inet:ipv6-address;
                  description
                    "The transport address advertised in LDP Hello
                     messages.";
                }

                container binding {
                  description
                    "LDP address and label binding information.";
                  list address {
                    key "address";
                    description
                      "List of address bindings.";
                    leaf address {
                      type inet:ipv6-address;
                      description
```

```
                        "Binding address.";
                  }
                  uses binding-address-state-attributes;
              } // binding-address

              list fec-label {
                key "fec";
                description
                  "List of label bindings.";
                leaf fec {
                  type inet:ipv6-prefix;
                  description
                    "Prefix FEC.";
                }
                uses binding-label-state-attributes;
              } // fec-label
            } // binding
          } // ipv6
        } // state
      } // address-family

      container discovery {
        description
          "Neibgbor discovery configuration.";

        container interfaces {
          description
            "A list of interfaces for basic descovery.";
          container config {
            description
              "Configuration data.";
            uses basic-discovery-timers;
          }
          container state {
            config false;
            description

              "Operational state data.";
            uses basic-discovery-timers;
          }

          list interface {
            key "interface";
            description
              "List of LDP interfaces.";
            leaf interface {
              type mpls-interface-ref;
              description
```

```
                    "Interface.";
                }
                container config {
                  description
                    "Configuration data.";
                  uses basic-discovery-timers {
                    if-feature per-interface-timer-config;
                  }
                  leaf igp-synchronization-delay {
                    if-feature per-interface-timer-config;
                    type uint16 {
                      range 3..60;
                    }
                    units seconds;
                    description
                      "Sets the interval that the LDP waits before
                       notifying the Interior Gateway Protocol (IGP)
                       that label exchange is completed so that IGP
                       can start advertising the normal metric for
                       the link.";
                  }
                }
                container state {
                  config false;
                  description
                    "Operational state data.";
                  uses basic-discovery-timers {
                    if-feature per-interface-timer-config;
                  }
                  leaf igp-synchronization-delay {
                    if-feature per-interface-timer-config;
                    type uint16 {
                      range 3..60;
                    }
                    units seconds;
                    description
                      "Sets the interval that the LDP waits before
                       notifying the Interior Gateway Protocol (IGP)
                       that label exchange is completed so that IGP
                       can start advertising the normal metric for
                       the link.";
                  }
                  leaf next-hello {
                    type uint16;
                    units seconds;
                    description "Time to send the next hello message.";
                  }
                } // state
```

```
list address-family {
  key "afi";
  description
    "Per-vrf per-af params.";
  leaf afi {
    type ldp-address-family;
    description
      "Address family type value.";
  }
  container config {
    description
      "Configuration data.";
    leaf enable {
      type boolean;
      description
        "Enable the address family on the interface.";
    }

    container ipv4 {
      must "/if:interfaces/if:interface"
        + "[name = current()/../../../interface]/"
        + "ip:ipv4" {
        description
          "Only if IPv4 is enabled on the interface.";
      }
      description
        "IPv4 address family.";
      leaf transport-address {
        type union {
          type enumeration {
            enum "use-interface-address" {
              description
              "Use interface address as the transport
               address.";
            }
          }
          type inet:ipv4-address;
        }
        description
          "IP address to be advertised as the LDP
           transport address.";
      }
    }

    container ipv6 {
      must "/if:interfaces/if:interface"
        + "[name = current()/../../../interface]/"
        + "ip:ipv6" {
```

```
                      description
                        "Only if IPv6 is enabled on the interface.";
                    }
                    description
                      "IPv6 address family.";
                    leaf transport-address {
                      type union {
                        type enumeration {
                          enum "use-interface-address" {
                            description
                              "Use interface address as the transport
                               address.";
                          }
                        }
                        type inet:ipv4-address;
                      }
                      description
                        "IP address to be advertised as the LDP
                         transport address.";
                    }
                  } // ipv6
                }
                container state {
                  config false;
                  description
                    "Operational state data.";
                  leaf enable {
                    type boolean;
                    description
                      "Enable the address family on the interface.";
                  }

                  container ipv4 {
                    must "/if:interfaces/if:interface"
                      + "[name = current()/../../../interface]/"
                      + "ip:ipv4" {
                      description
                        "Only if IPv4 is enabled on the interface.";
                    }
                    description
                      "IPv4 address family.";
                    leaf transport-address {
                      type union {
                        type enumeration {

                          enum "use-interface-address" {
                            description
                              "Use interface address as the transport
```

```
                             address.";
                    }
                  }
                  type inet:ipv4-address;
                }
                description
                  "IP address to be advertised as the LDP
                   transport address.";
              }

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

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

                uses adjacency-state-attributes;

                leaf peer {
                  type leafref {
                    path "../../../../../../../../peers/peer/"
                      + "lsr-id";
                  }
                  description
                    "LDP peer from this adjacency.";
                }
              } // hello-adjacencies
            }
            container ipv6 {
              must "/if:interfaces/if:interface"
                + "[name = current()/../../../interface]/"
                + "ip:ipv6" {
                description
                  "Only if IPv6 is enabled on the interface.";
              }
              description
                "IPv6 address family.";
              leaf transport-address {
                type union {
                  type enumeration {
                    enum "use-interface-address" {
                      description
                      "Use interface address as the transport
                       address.";
```

```
                    }
                  }
                  type inet:ipv4-address;
                }
                description
                  "IP address to be advertised as the LDP
                   transport address.";
              }

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

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

                uses adjacency-state-attributes;

                leaf peer {
                  type leafref {
                    path "../../../../../../../../peers/peer/"
                      + "lsr-id";
                  }
                  description
                    "LDP peer from this adjacency.";
                }
              } // hello-adjacencies
            } // ipv6
          }
        } // address-family
      } // list interface
    } // interfaces

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

        description
          "Configuration data.";
        uses extended-discovery-timers;
        uses extended-discovery-policy-attributes;
      }
      container state {
```

```
            config false;
            description
              "Operational state data.";
            uses extended-discovery-timers;
            uses extended-discovery-policy-attributes;
          }

          list address-family {
            key "afi";
            description
              "Per-af params.";
            leaf afi {
              type ldp-address-family;
              description
                "Address family type value.";
            }

            container state {
              config false;
              description
                "Operational state data.";

              container ipv4 {
                when "../../afi = 'ipv4'" {
                  description
                    "For IPv4.";
                }
                description
                  "IPv4 address family.";
                list hello-adjacencies {
                  key "local-address adjacent-address";
                  description "List of hello adjacencies.";

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

                  uses adjacency-state-attributes;

                  leaf peer {
                    type leafref {
```

```
                            path "../../../../../../../peers/peer/"
                              + "lsr-id";
                        }
                        description
                          "LDP peer from this adjacency.";
                      }
                    } // hello-adjacencies
                  } // ipv4

                  container ipv6 {
                    when "../../afi = 'ipv6'" {
                      description
                        "For IPv6.";
                    }
                    description
                      "IPv6 address family.";
                    list hello-adjacencies {
                      key "local-address adjacent-address";
                      description "List of hello adjacencies.";

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

                      uses adjacency-state-attributes;

                      leaf peer {
                        type leafref {
                          path "../../../../../../../peers/peer/"
                            + "lsr-id";
                        }
                        description
                          "LDP peer from this adjacency.";
                      }
                    } // hello-adjacencies
                  } // ipv6
                } // state

                container ipv4 {
                  when "../afi = 'ipv4'" {
                    description
```

```
                    "For IPv4.";
                }
              description
                "IPv4 address family.";
              list target {
                key "adjacent-address";
                description
                  "Targeted discovery params.";

                leaf adjacent-address {
                  type inet:ipv4-address;
                  description
                    "Configures a remote LDP neighbor and enables
                     extended LDP discovery of the specified
                     neighbor.";
                }
                container config {
                  description
                    "Configuration data.";
                  leaf enable {
                    type boolean;
                    description
                      "Enable the target.";
                  }
                  leaf local-address {
                    type inet:ipv4-address;
                    description
                      "The local address.";
                  }
                }
                container state {
                  config false;
                  description
                    "Operational state data.";
                  leaf enable {
                    type boolean;
                    description
                      "Enable the target.";
                  }
                  leaf local-address {
                    type inet:ipv4-address;
                    description
                      "The local address.";
                  }
                } // state
              }
            } // ipv4
            container ipv6 {
```

```
                when "../afi = 'ipv6'" {
                  description
                    "For IPv6.";
                }
                description
                  "IPv6 address family.";
                list target {
                  key "adjacent-address";
                  description
                    "Targeted discovery params.";

                  leaf adjacent-address {
                    type inet:ipv6-address;
                    description
                      "Configures a remote LDP neighbor and enables
                       extended LDP discovery of the specified
                       neighbor.";
                  }
                  container config {
                    description
                      "Configuration data.";
                    leaf enable {
                      type boolean;
                      description
                        "Enable the target.";
                    }
                    leaf local-address {
                      type inet:ipv6-address;
                      description
                        "The local address.";
                    }
                  }
                  container state {
                    config false;
                    description
                      "Operational state data.";
                    leaf enable {
                      type boolean;
                      description
                        "Enable the target.";
                    }
                    leaf local-address {
                      type inet:ipv6-address;
                      description
                        "The local address.";
                    }
                  } // state
                }
```

```
              } // ipv6
            } // address-family
          } // targeted
        } // discovery

      container forwarding-nexthop {
        if-feature forwarding-nexthop-config;
        description
          "Configuration for forwarding nexthop.";

        container interfaces {
          description
            "A list of interfaces on which forwarding is
             disabled.";

          list interface {
            key "interface";
            description
              "List of LDP interfaces.";
            leaf interface {
              type mpls-interface-ref;
              description
                "Interface.";
            }
            list address-family {
              key "afi";
              description
                "Per-vrf per-af params.";
              leaf afi {
                type ldp-address-family;
                description
                  "Address family type value.";
              }
              container config {
                description
                  "Configuration data.";
                leaf ldp-disable {
                  type boolean;
                  description
                    "Disable LDP forwarding on the interface.";
                }
                leaf mldp-disable {
                  if-feature mldp;
                  type boolean;
                  description
                    "Disable mLDP forwarding on the interface.";
                }
              }
```

```
                container state {
                  config false;
                  description
                    "Operational state data.";
                  leaf ldp-disable {
                    type boolean;
                    description
                      "Disable LDP forwarding on the interface.";
                  }
                  leaf mldp-disable {
                    if-feature mldp;

                    type boolean;
                    description
                      "Disable mLDP forwarding on the interface.";
                  }
                }
              } // address-family
            } // list interface
          } // interfaces
        } // forwarding-nexthop
        uses policy-container {
          if-feature all-af-policy-config;
        }
      } // global

      container peers {
        description
          "Peers configuration attributes.";

        container config {
          description
            "Configuration data.";
          uses peer-authentication {
            if-feature global-session-authentication;
          }
          uses peer-attributes;

          container session-downstream-on-demand {
            if-feature session-downstream-on-demand-config;
            description
              "Session downstream-on-demand attributes.";
            leaf enable {
              type boolean;
              description
                "'true' if session downstream-on-demand is enabled.";
            }
            leaf peer-list {
```

```
                  type peer-list-ref;
                  description
                    "The name of a peer ACL.";
                }
              }
            }
            container state {
              config false;
              description
                "Operational state data.";
              uses peer-authentication {
                if-feature global-session-authentication;
              }
              uses peer-attributes;

              container session-downstream-on-demand {
                if-feature session-downstream-on-demand-config;
                description
                  "Session downstream-on-demand attributes.";
                leaf enable {
                  type boolean;
                  description
                    "'true' if session downstream-on-demand is enabled.";
                }
                leaf peer-list {
                  type peer-list-ref;
                  description
                    "The name of a peer ACL.";
                }
              }
            }

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

              leaf lsr-id {
                type yang:dotted-quad;
                description "LSR ID.";
              }

              container config {
                description
                  "Configuration data.";
                leaf admin-down {
                  type boolean;
                  default false;
```

```
                description
                  "'true' to disable the peer.";
              }

              container capability {
                description
                  "Per peer capability";
                container mldp {
                  if-feature mldp;
                  description
                    "mLDP capabilities.";
                  uses mldp-capabilities;
                }
              }

              uses peer-af-policy-container {
                if-feature all-af-policy-config;
              }

              uses peer-authentication;

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

              uses peer-attributes {
                if-feature per-peer-session-attributes-config;
              }

              container address-family {
                description
                  "Per-vrf per-af params.";
                container ipv4 {
                  description
                    "IPv4 address family.";
                  uses peer-af-policy-container;
                }
                container ipv6 {
                  description
                    "IPv6 address family.";
                  uses peer-af-policy-container;
                } // ipv6
              } // address-family
            }
            container state {
              config false;
              description
                "Operational state data.";
```

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

            container capability {
              description
                "Per peer capability";
              container mldp {
                if-feature mldp;
                description
                  "mLDP capabilities.";
                uses mldp-capabilities;
              }
            }

            uses peer-af-policy-container {
              if-feature all-af-policy-config;
            }

            uses peer-authentication;

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

            uses peer-attributes {
              if-feature per-peer-session-attributes-config;
            }

            container address-family {
              description
                "Per-vrf per-af params.";
              container ipv4 {
                description
                  "IPv4 address family.";
                uses peer-af-policy-container;

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

                  leaf local-address {
                    type inet:ipv4-address;
                    description
                      "Local address of the hello adjacency.";
```

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

              uses adjacency-state-attributes;

              leaf interface {
                type mpls-interface-ref;
                description "Interface for this adjacency.";
              }
            } // hello-adjacencies
          } // ipv4
          container ipv6 {
            description
              "IPv6 address family.";
            uses peer-af-policy-container;

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

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

              uses adjacency-state-attributes;

              leaf interface {
                type mpls-interface-ref;
                description "Interface for this adjacency.";
              }
            } // hello-adjacencies
          } // ipv6
        } // address-family

        uses peer-state-derived;
      } // state
    } // list peer
```

```
        } // peers
      } // container mpls-ldp
    }

    /*
     * RPCs
     */
    rpc mpls-ldp-clear-peer {
      description
        "Clears the session to the peer.";
      input {
        leaf lsr-id {
          type union {
            type yang:dotted-quad;
            type uint32;
          }
          description
            "LSR ID of peer to be cleared. If this is not provided
             then all peers are cleared";
        }
      }
    }

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

```
                  presence "Present to clear link adjacencies.";
                  description
                    "Clear link adjacencies.";
                  leaf next-hop-interface {
                    type mpls-interface-ref;
                    description

                      "Interface connecting to next-hop. If this is not
                       provided then all link adjacencies are cleared.";
                  }
                  leaf next-hop-address {
                    type inet:ip-address;
                    must "../next-hop-interface" {
                      description
                        "Applicable when interface is specified.";
                    }
                    description
                      "IP address of next-hop. If this is not provided
                       then adjacencies to all next-hops on the given
                       interface are cleared.";
                  } // next-hop-address
                } // link
              }
            }
          }
        }
      }

    rpc mpls-ldp-clear-peer-statistics {
      description
        "Clears protocol statistics (e.g. sent and received
         counters).";
      input {
        leaf lsr-id {
          type union {
            type yang:dotted-quad;
            type uint32;
          }
          description
            "LSR ID of peer whose statistic are to be cleared.
             If this is not provided then all peers statistics are
             cleared";
        }
      }
    }

    /*
     * Notifications
```

```
   */
   notification mpls-ldp-peer-event {

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

   notification mpls-ldp-hello-adjacency-event {
     description
       "Notification event for a change of LDP adjacency operational
        status.";
     leaf event-type {
       type oper-status-event-type;
       description "Event type.";
     }
     uses ldp-adjacency-ref;
   }

   notification mpls-ldp-fec-event {
     description
       "Notification event for a change of FEC status.";
     leaf event-type {
       type oper-status-event-type;
       description "Event type.";
     }
     uses ldp-fec-event;
   }

   notification mpls-mldp-fec-event {
     description
       "Notification event for a change of FEC status.";
     leaf event-type {
       type oper-status-event-type;
       description "Event type.";
     }
     uses mldp-fec-event;
   }
 }

 <CODE ENDS>
```

Figure 22

7.  Security Considerations

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

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

8.  IANA Considerations

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

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

9.  Acknowledgments

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

10.  References

10.1.  Normative References

   [I-D.ietf-netmod-routing-cfg]
            Lhotka, L. and A. Lindem, "A YANG Data Model for Routing
            Management", draft-ietf-netmod-routing-cfg-22 (work in
            progress), July 2016.

   [I-D.rtgyangdt-rtgwg-ni-model]
            Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic,
            "Network Instance Model", draft-rtgyangdt-rtgwg-ni-
            model-00 (work in progress), May 2016.

   [I-D.saad-mpls-base-yang]
            Raza, K., Gandhi, R., Liu, X., Beeram, V., Saad, T.,
            Bryskin, I., Chen, X., Jones, R., and B. Wen, "A YANG Data
            Model for MPLS Base", draft-saad-mpls-base-yang-00 (work
            in progress), May 2016.

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

   [RFC3478]  Leelanivas, M., Rekhter, Y., and R. Aggarwal, "Graceful
              Restart Mechanism for Label Distribution Protocol",
              RFC 3478, DOI 10.17487/RFC3478, February 2003,
              <http://www.rfc-editor.org/info/rfc3478>.

   [RFC5036]  Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed.,
              "LDP Specification", RFC 5036, DOI 10.17487/RFC5036,
              October 2007, <http://www.rfc-editor.org/info/rfc5036>.

   [RFC5331]  Aggarwal, R., Rekhter, Y., and E. Rosen, "MPLS Upstream
              Label Assignment and Context-Specific Label Space",
              RFC 5331, DOI 10.17487/RFC5331, August 2008,
              <http://www.rfc-editor.org/info/rfc5331>.

   [RFC5561]  Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL.
              Le Roux, "LDP Capabilities", RFC 5561,
              DOI 10.17487/RFC5561, July 2009,
              <http://www.rfc-editor.org/info/rfc5561>.

   [RFC5918]  Asati, R., Minei, I., and B. Thomas, "Label Distribution
              Protocol (LDP) 'Typed Wildcard' Forward Equivalence Class
              (FEC)", RFC 5918, DOI 10.17487/RFC5918, August 2010,
              <http://www.rfc-editor.org/info/rfc5918>.

   [RFC5919]  Asati, R., Mohapatra, P., Chen, E., and B. Thomas,
              "Signaling LDP Label Advertisement Completion", RFC 5919,
              DOI 10.17487/RFC5919, August 2010,
              <http://www.rfc-editor.org/info/rfc5919>.

   [RFC6020]  Bjorklund, M., Ed., "YANG - A Data Modeling Language for
              the Network Configuration Protocol (NETCONF)", RFC 6020,
              DOI 10.17487/RFC6020, October 2010,
              <http://www.rfc-editor.org/info/rfc6020>.

   [RFC6241]  Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
              and A. Bierman, Ed., "Network Configuration Protocol
              (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
              <http://www.rfc-editor.org/info/rfc6241>.

   [RFC6388]  Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B.
              Thomas, "Label Distribution Protocol Extensions for Point-
              to-Multipoint and Multipoint-to-Multipoint Label Switched
              Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011,
              <http://www.rfc-editor.org/info/rfc6388>.

   [RFC6389]  Aggarwal, R. and JL. Le Roux, "MPLS Upstream Label
              Assignment for LDP", RFC 6389, DOI 10.17487/RFC6389,
              November 2011, <http://www.rfc-editor.org/info/rfc6389>.

   [RFC6512]  Wijnands, IJ., Rosen, E., Napierala, M., and N. Leymann,
              "Using Multipoint LDP When the Backbone Has No Route to
              the Root", RFC 6512, DOI 10.17487/RFC6512, February 2012,
              <http://www.rfc-editor.org/info/rfc6512>.

   [RFC6536]  Bierman, A. and M. Bjorklund, "Network Configuration
              Protocol (NETCONF) Access Control Model", RFC 6536,
              DOI 10.17487/RFC6536, March 2012,
              <http://www.rfc-editor.org/info/rfc6536>.

   [RFC6826]  Wijnands, IJ., Ed., Eckert, T., Leymann, N., and M.
              Napierala, "Multipoint LDP In-Band Signaling for Point-to-
              Multipoint and Multipoint-to-Multipoint Label Switched
              Paths", RFC 6826, DOI 10.17487/RFC6826, January 2013,
              <http://www.rfc-editor.org/info/rfc6826>.

   [RFC7060]  Napierala, M., Rosen, E., and IJ. Wijnands, "Using LDP
              Multipoint Extensions on Targeted LDP Sessions", RFC 7060,
              DOI 10.17487/RFC7060, November 2013,
              <http://www.rfc-editor.org/info/rfc7060>.

   [RFC7140]  Jin, L., Jounay, F., Wijnands, IJ., and N. Leymann, "LDP
              Extensions for Hub and Spoke Multipoint Label Switched
              Path", RFC 7140, DOI 10.17487/RFC7140, March 2014,
              <http://www.rfc-editor.org/info/rfc7140>.

   [RFC7246]  Wijnands, IJ., Ed., Hitchen, P., Leymann, N., Henderickx,
              W., Gulko, A., and J. Tantsura, "Multipoint Label
              Distribution Protocol In-Band Signaling in a Virtual
              Routing and Forwarding (VRF) Table Context", RFC 7246,
              DOI 10.17487/RFC7246, June 2014,
              <http://www.rfc-editor.org/info/rfc7246>.

   [RFC7438]  Wijnands, IJ., Ed., Rosen, E., Gulko, A., Joorde, U., and
              J. Tantsura, "Multipoint LDP (mLDP) In-Band Signaling with
              Wildcards", RFC 7438, DOI 10.17487/RFC7438, January 2015,
              <http://www.rfc-editor.org/info/rfc7438>.

   [RFC7552]  Asati, R., Pignataro, C., Raza, K., Manral, V., and R.
              Papneja, "Updates to LDP for IPv6", RFC 7552,
              DOI 10.17487/RFC7552, June 2015,
              <http://www.rfc-editor.org/info/rfc7552>.

   [RFC7715]  Wijnands, IJ., Ed., Raza, K., Atlas, A., Tantsura, J., and
              Q. Zhao, "Multipoint LDP (mLDP) Node Protection",
              RFC 7715, DOI 10.17487/RFC7715, January 2016,
              <http://www.rfc-editor.org/info/rfc7715>.

10.2.  Informative References

   [I-D.ietf-rtgwg-policy-model]
              Shaikh, A., Shakir, R., D'Souza, K., and C. Chase,
              "Routing Policy Configuration Model for Service Provider
              Networks", draft-ietf-rtgwg-policy-model-01 (work in
              progress), April 2016.

   [I-D.iwijnand-mpls-mldp-multi-topology]
              Wijnands, I. and K. Raza, "mLDP Extensions for Multi
              Topology Routing", draft-iwijnand-mpls-mldp-multi-
              topology-03 (work in progress), June 2013.

   [I-D.openconfig-netmod-opstate]
              Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling
              of Operational State Data in YANG", draft-openconfig-
              netmod-opstate-01 (work in progress), July 2015.

   [RFC4364]  Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private
              Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February
              2006, <http://www.rfc-editor.org/info/rfc4364>.

   [RFC7307]  Zhao, Q., Raza, K., Zhou, C., Fang, L., Li, L., and D.
              King, "LDP Extensions for Multi-Topology", RFC 7307,
              DOI 10.17487/RFC7307, July 2014,
              <http://www.rfc-editor.org/info/rfc7307>.

Appendix A.  Additional Contributors

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

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

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

Authors' Addresses

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


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


Sowmya Krishnaswamy
Cisco Systems, Inc.
Email: sowkrish@cisco.com


Xufeng Liu
Ericsson
Email: xliu@kuatrotech.com


Jeff Tantsura
Ericsson
Email: jeff.tantsura@ericsson.com


Santosh Esale
Juniper Networks
Email: sesale@juniper.net


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

Loa Andersson
Huawei Technologies
Email: loa@pi.nu


Himanshu Shah
Ciena Corporation
Email: hshah@ciena.com


Matthew Bocci
Alcatel-Lucent
Email: matthew.bocci@alcatel-lucent.com

MPLS WG                                                 K. Kompella
Internet-Draft                                  Juniper Networks, Inc.
Intended status: Standards Track                       L. Contreras
Expires: January 8, 2017                                  Telefonica
                                                        July 7, 2016

                          Resilient MPLS Rings
                          draft-ietf-mpls-rmr-02

   Abstract

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

   Requirements Language

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

   Status of This Memo

   Copyright Notice

Table of Contents

1.  Introduction

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

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

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

## 1.1.  Definitions

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

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

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

Figure 1: Ring with 8 nodes

The following terminology is used for ring LSPs:

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

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

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

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

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

Ring links:  Links that connnect ring neighbors.

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

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

    UN: 00  undefined link

    CW: 01  clockwise ring link

    AC: 10  anticlockwise ring link

    EX: 11  express link

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

The following notation is used for ring LSPs:

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

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

CL_jk:  A label allocated by R_j for RL_k in the CW direction.

AL_jk:  A label allocated by R_j for RL_k in the AC direction.

P_jk (Q_jk):  A Path (Resv) message sent by R_j for RL_k.

2.  Motivation

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

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

3.  Theory of Operation

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

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

3.1.  Provisioning

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

3.2.  Ring Nodes

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

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

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

3.3.  Ring Links and Directions

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

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

3.3.1.  Express Links

   Express links are discovered once ring nodes, ring links and
   directions have been established.  As defined earlier, express links
   are links joining non-neighboring ring nodes; often, this may be the

result of optically bypassing ring nodes.  The use of express links
will be described in a future version of this document.

## 3.4.  Ring LSPs

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

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

## 3.5.  Installing Primary LFIB Entries

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

Clearly, R_k does not act as ingress for its own LSPs.  However, if
these LSPs use UHP, then R_k installs LFIB entries to pop CL_k,k for
packets received from R_k-1 and to pop AL_k,k for packets received
from R_k+1.

## 3.6.  Installing FRR LFIB Entries

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

3.7.  Protection

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

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

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

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

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

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

   2.  A ring node sends protected traffic (i.e., traffic switched from
       CW to AC or vice versa) with TTL just large enough to reach the
       egress.

   3.  A ring node sends protected traffic with a special purpose label
       below the ring LSP label.  A protecting node first checks for the
       presence of this label; if present, it means that the traffic is
       looping and MUST be dropped.

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

4.  Autodiscovery

4.1.  Overview

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

```
              S1
             /   \
            |     R0 . . . R1        R0 has MV = 11
            |   .   \          .     R1 has MV = 10
            R7        _____ R2   All other nodes have MV = 00
Anti-       |   .                .   |
clockwise   |   .       Ring      .  | Clockwise
         v  .         RID = 17    .  v
            R6                    R3
              .                 .
              R5 . . . R4
                \       /
                 \     /
                   An
```
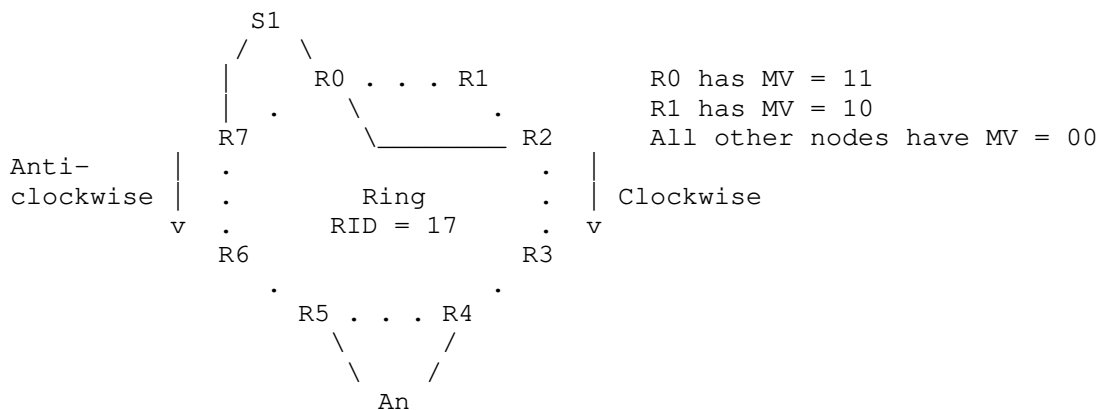
               Figure 2: Ring with non-ring nodes and links

   In what follows, we refer to a ring node and a rink link Type-Length-
   Value (TLV).  These are new TLVs that contain RIDs and associated
   flags.  A ring node TLV is a TLV that contains information for each
   ring that this node participates in.  A ring link TLV identifies a
   link and contains information about every ring that that link is part
   of.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type (TBD)  |  Length = 6*N |      Ring ID 1 (4 octets) ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    ... (RID continued)        |        Flags (2 octets)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Ring ID 2 (4 octets)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Flags (2 octets)        |     Ring ID 2 (4 octets) ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    ... (RID continued)        |        Flags (2 octets)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ... etc.                                                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Ring Node TLV Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type (TBD)  | Length=8+6*N  |     My Interface Index ...    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ... (continued, 4 octets)     |   Remote Interface Index ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ... (continued, 4 octets)     |     Ring ID 1 (4 octets) ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    ... (RID continued)        |        Flags (2 octets)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Ring ID 2 (4 octets)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Flags (2 octets)        |     Ring ID 2 (4 octets) ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    ... (RID continued)        |        Flags (2 octets)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ... etc.                                                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Ring Link TLV Format

```
   0                       1
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |MV |SS | SO  |G|   MBZ   |SU |M|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  MV: Mastership Value
  SS: Supported Signaling Protocols (10 = RSVP-TE; 01 = LDP)
  SO: Supported OAM Protocols (100 = BFD; 010 = CFM; 001 = EFM)
  G:  Node is a Grandmaster Clock (1 = True, 0 = False)
  SU: Signaling Protocol to Use  (00 = none; 01 = LDP; 10 = RSVP-TE)
  M : Elected Master (0 = no, 1 = yes)
```

                  Flags for a Ring Node TLV

```
   0                       1
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |RD |OAM|         MBZ         |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  RD:  Ring Direction
  OAM: OAM Protocols (00 = none; 01 = BFD; 10 = CFM; 11 = EFM)
```

                  Flags for a Ring Link TLV

## 4.2.  Ring Announcement Phase

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

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

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

## 4.3.  Mastership Phase

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

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

4.4.  Ring Identification Phase

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

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

4.5.  Ring Changes

The main changes to a ring are:

    ring link addition;

    ring link deletion;

    ring node addition; and

    ring node deletion.

The main goal of handling ring changes is (as much as possible) not
to perturb existing ring operation.  Thus, if the ring master hasn't
changed, all of the above changes should be local to the point of
change.  Link adds just update the IGP; signaling should take
advantage of the new capacity as soon as it learns.  Link deletions
in the case of parallel links also show up as a change in capacity
(until the last link in the bundle is removed.)

The removal of the last ring link between two nodes, or the removal
of a ring node is an event that triggers protection switching.  In a
simple ring, the result is a broken ring.  However, if a ring has
express links, then it may be able to converge to a smaller ring with
protection.  Details of this process will be given in a future
version.

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

## 5.  Ring Signaling

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

## 6.  Ring OAM

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

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

## 7.  Security Considerations

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

## 8.  Acknowledgments

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

## 9.  IANA Considerations

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

10.  References

10.1.  Normative References

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

10.2.  Informative References

   [RFC3209]  Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V.,
              and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP
              Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001,
              <http://www.rfc-editor.org/info/rfc3209>.

   [RFC3630]  Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering
              (TE) Extensions to OSPF Version 2", RFC 3630,
              DOI 10.17487/RFC3630, September 2003,
              <http://www.rfc-editor.org/info/rfc3630>.

   [RFC5036]  Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed.,
              "LDP Specification", RFC 5036, DOI 10.17487/RFC5036,
              October 2007, <http://www.rfc-editor.org/info/rfc5036>.

   [RFC5305]  Li, T. and H. Smit, "IS-IS Extensions for Traffic
              Engineering", RFC 5305, DOI 10.17487/RFC5305, October
              2008, <http://www.rfc-editor.org/info/rfc5305>.

Authors' Addresses

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

   Email: kireeti.kompella@gmail.com

   Luis M. Contreras
   Telefonica
   Ronda de la Comunicacion
   Sur-3 building, 3rd floor
   Madrid  28050
   Spain

   Email: luismiguel.contrerasmurillo@telefonica.com
   URI:   http://people.tid.es/LuisM.Contreras/

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

Abstract

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

Requirements Language

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119][RFC8174].

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

Copyright Notice

Table of Contents

1.  Introduction

   Rings are a very common topology either at infrastructure level
   (e.g., physical ring fiber deployments in Layer 1 networks) or node
   interconnection structures (e.g., loops created in bridged
   interconnected infrastructures [IEEE.802.1D_2004]).  A ring is the
   simplest topology offering link and node resilience.  Rings are
   nearly ubiquitous in access and aggregation networks.  As MPLS
   increases its presence in such networks, and takes on a greater role,
   it is imperative that MPLS handles rings well; this is not the case
   today.

   This document describes the special nature of rings, and the special
   needs of MPLS on rings.  It then shows how these needs can be met in
   several ways, some of which involve extensions to protocols such as
   IS-IS [RFC5305], OSPF[RFC3630], RSVP-TE [RFC3209] and LDP [RFC5036].
   RMR LSPs can also be signaled with IGP [RFC8402]; that will be
   described in a future document.

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

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

1.1.  Definitions

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

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

"clockwise" (CW) and the reverse direction as "anticlockwise" (AC).
As there may be several rings in a graph, we number each ring with a
distinct ring ID RID.

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

Figure 1: Ring with 8 nodes

The following terminology is used for ring LSPs:

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

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

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

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

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

Ring links:  Links that connect ring neighbors.

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

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

   UN: 00  undefined link

   CW: 01  clockwise ring link

        AC: 10   anticlockwise ring link

        EX: 11   express link

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

   The following notation is used for ring LSPs:

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

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

   CL_jk:  A label allocated by R_j for RL_k in the CW direction.

   AL_jk:  A label allocated by R_j for RL_k in the AC direction.

1.2.  Changes from -12 in response to reviews

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

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

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

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

      Sec 1: rephrased last sentence [TAR]

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

      Sec 3.1: typo fixed [Gen]

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

      Sec 3.5: updated reference [OPS]

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

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

      Sec 4.1: clean up [Gen]

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

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

      Sec 4.5: removed "and" [Gen]

      Sec 5: updated text on timers [TAR]

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

      Sec Cons: added text on compromised nodes [SAD]

2.  Motivation

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

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

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

3.  Theory of Operation

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

   Once ring identification is complete, each node signals one or more
   ring LSPs RL_i.  RL_i, anchored on node R_i, consists of two counter-
   rotating unicast LSPs that start and end at R_i.  A ring LSP is
   "multipoint": any node R_j can use RL_i to send traffic to R_i; this
   can be in either the CW or AC directions, or both (i.e., load

balanced).  Both of these counter-rotating LSPs are "active"; the
choice of direction to send traffic to R_i is determined by policy at
the node where traffic is injected into the ring.  The default policy
is to send traffic along the shortest path.  Bidirectional
connectivity between nodes R_i and R_j is achieved by using two
different ring LSPs: R_i uses RL_j to reach R_j, and R_j uses RL_i to
reach R_i.

## 3.1.  Provisioning

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

## 3.2.  Ring Nodes

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

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

```
                         R0 . . . R1
                       .             .
                    R7                 R2
          Anti-     │ .       Ring       . │
          Clockwise │ .                  . │ Clockwise
                  v   .     RID = 17      .   v
                    R6                 R3
                       .             .
                        R5 . . . R4
                       .             .
                    R13                R8
          Anti-     │ .       Ring       . │
          Clockwise │ .                  . │ Clockwise
                  v   .     RID = 18      .   v
                    R12                R9
                       .             .
                        R11 . . . R10

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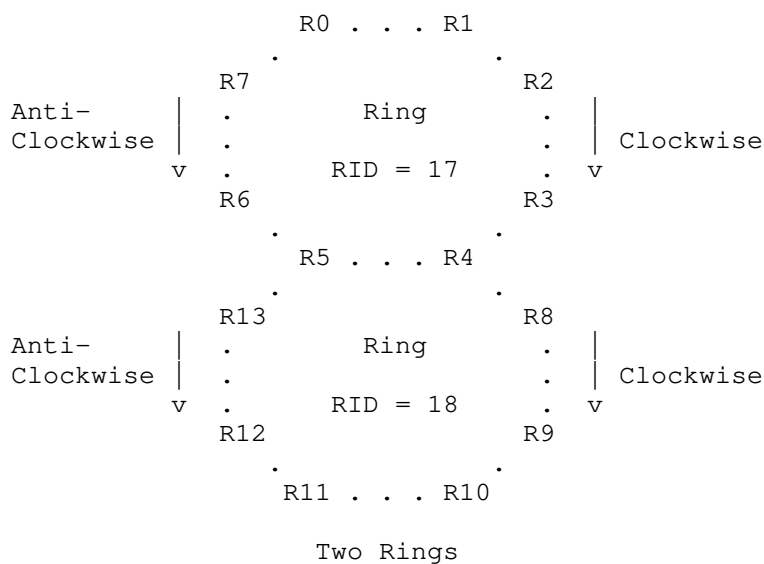
   If R3 and R6 are configured with RID 17, R8 and R13 with RID 18, and
   all other nodes with RID 0, this will end up as two rings with R4 and
   R5 in both.  However, other permutations of RID configurations could
   easily end up with all nodes being in both rings 17 and 18, whereupon
   the maximal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

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

3.3.  Ring Links and Directions

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

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

### 3.3.1.  Express Links

Express links are discovered once ring nodes, ring links and
directions have been established.  As defined earlier, express links
are links joining non-neighboring ring nodes; often, this may be the
result of optically bypassing ring nodes.

### 3.4.  Ring LSPs

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

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

### 3.5.  Installing Primary LFIB Entries

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

Clearly, R_k does not act as ingress for its own LSPs.  However, R_k
can send OAM messages, for example, an MPLS ping or traceroute
([RFC8029]), using labels CL_k,k+1 and AL_k-1,k, to test the entire

ring LSP anchored at R_k in both directions.  Furthermore, if these
LSPs use Ultimate Hop Popping, then R_k installs LFIB entries to pop
CL_k,k for packets received from R_k-1 and to pop AL_k,k for packets
received from R_k+1.

3.6.  Protection

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

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

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

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

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

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

2.  A ring node sends protected traffic (i.e., traffic switched from
    CW to AC or vice versa) with TTL just large enough to reach the
    egress.

   3.  A ring node sends protected traffic with a special purpose label
       below the ring LSP label.  A protecting node first checks for the
       presence of this label; if present, it means that the traffic is
       looping and MUST be dropped.

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

3.7.  Installing FRR LFIB Entries

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

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

```
        +------+--------+-----------+--------+-----------+
        | Dest | CW/NH  | CW FRR/NH | AC/NH  | AC FRR/NH |
        +------+--------+-----------+--------+-----------+
        | ...  |        |           |        |           |
        | R4   | L42/R2 |  L40/R0   | L40/R0 |  L42/R2   |
        | R5   | L52/R2 |  L50/R0   | L50/R0 |  L52/R2   |
        | ...  |        |           |        |           |
        +------+--------+-----------+--------+-----------+
```

                            R1's LFIB

4.  Autodiscovery

4.1.  Overview

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

```
              S1
            /   \
           |     R0 . . . R1        R0 has MV = 11
           |  .    \       .        R1 has MV = 10
           R7        _____ R2   All other nodes have MV = 00
  Anti-    |  .                . |
  clockwise|  .      Ring      . | Clockwise
         v .        RID = 17   . v
           R6                  R3
              .              .
              R5 . . . R4
                \       /
                 \     /
                    An
```
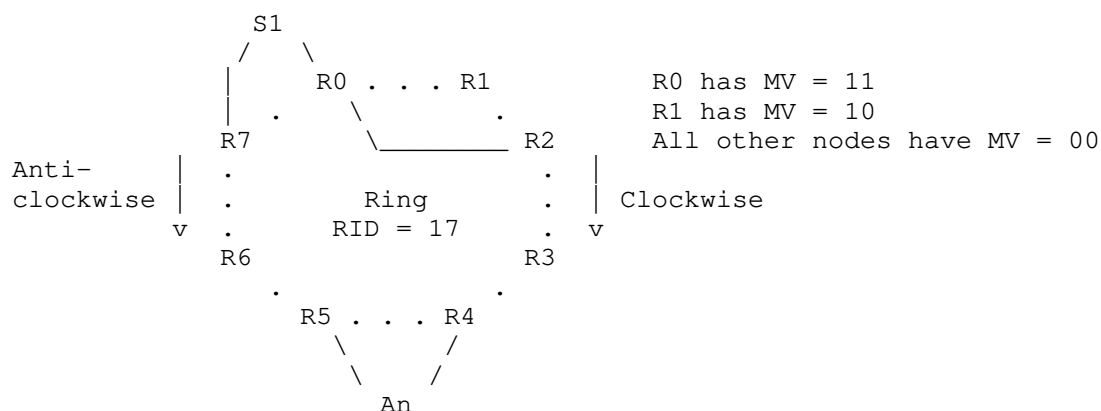
                 Figure 2: Ring with non-ring nodes and links

   We use three concepts below:

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

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

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

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

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

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

                         Ring Node TLV Format

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

                          Ring Link Sub-TLV Format

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

                        Flags for a Ring Node TLV

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

                        Flags for a Ring Link TLV

4.2.  Ring Announcement Phase

   Each node participating in an MPLS ring is assigned an RID; in the
   example, RID = 17.  A node is also provisioned with a mastership
   value.  Each node advertises a ring node TLV for each ring it is
   participating in, along with the associated flags.  It then starts
   timer T1; this timer is to allow each node time to hear from all
   other nodes in the ring.  [The settings for timers T1 and T2 (below)
   are particular to the specific IGP used for signaling; they will be
   discussed in the IGP document that defines the ring node/link TLVs.]
   The settings for timers T1 and T2 (below) will be discussed in the
   IGP document that defines the ring node/link TLVs.]

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

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

4.3.  Mastership Phase

   When timer T1 fires, a node enters the mastership phase.  In this
   phase, each ring node N starts timer T2 and checks if it is master as
   follows.  N examines the MV value of all ring nodes and selects those
   with the highest MV vale.  Among these nodes, N finds the node with
   the lowest loopback address.  If that node is N, N declares itself
   master to the entire ring by readvertising its ring node TLV with the
   M bit set.

   When timer T2 fires, each node examines the ring node TLVs from all
   other nodes in the ring to identify the ring master.  There should be
   exactly one; if not, each node restarts timer T2 and tries again.

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

4.4.  Ring Identification Phase

```
                    R0 . . . R1   <------ 3. Anti-clockwise neighbor
                 .          .
             R7 --------------  R2 <--- 0. Ring Master
             .        Ring    / .
             .               /  .        1. Maximal ring includes R3
             .              /   .
             R6            /   R3 <--- 2. Clockwise neighbor
               .          /   .
                R5 . . . R4   <------ 4. R4 is express neighbor
                              R7 is also an express neighbor of R2
```
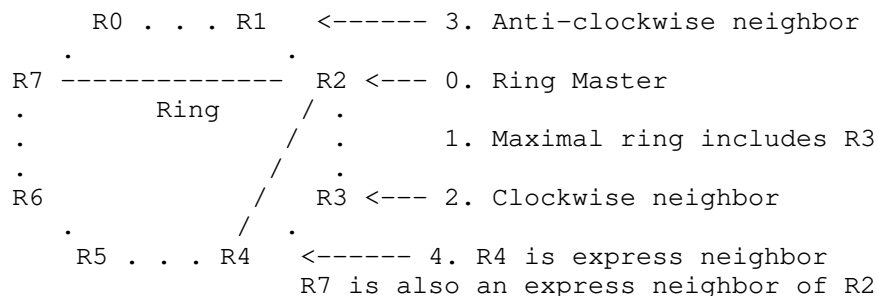
                     Figure 3: Ring Identification

   When there is exactly one ring master M (here, R2), M enters the Ring
   Identification Phase.  M indicates that it has successfully completed
   this phase by advertising ring link TLVs.  This is the trigger for

M's CW neighbor to enter the Ring Identification Phase.  This phase
passes CW until all ring nodes have completed ring identification.

The Ring Identification Phase proceeds as follows:

1.  M identifies all ring nodes for ring RID, i.e., those that have
    announced ring node TLVs with the ring ID = RID.

2.  M computes a maximal ring among these nodes.

3.  Based on that, M picks a CW neighbor and an AC neighbor.

4.  M then inserts ring link TLVs with ring direction CW for each
    link to its CW neighbor; M also inserts a ring link TLV with
    direction AC for each link to its AC neighbor.  (Note that there
    may be multiple links from M to each of its neighbors.)

5.  Finally, M determines its express links.  These are links to IGP
    neighbors that are ring nodes but neither the CW or AC neighbor.
    M advertises ring link TLVs for express links by setting the link
    direction to "express link".

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

1.  Each node Y listens for ring link TLVs.  The set of nodes S
    consists of those that have announced ring link TLVs.

2.  If a node Z announces a ring link TLV with Y as the CW neighbor,
    then Y is next.

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

1.  when X computes a maximal ring, it MUST include all nodes in S.

2.  X knows its AC neighbor (Z above), and doesn't have to pick it.

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

4.5.  Ring Changes

The main changes to a ring are:

    ring link addition;

ring link deletion;

ring node addition;

ring node deletion.

The main goal of handling ring changes is (as much as possible) not
to perturb existing ring operation.  Thus, if the ring master hasn't
changed, all of the above changes should be local to the point of
change.  Link adds just update the IGP; signaling should take
advantage of the new capacity as soon as it learns.  Link deletions
in the case of parallel links also show up as a change in capacity
(until the last link in the bundle is removed.)

The removal of the last ring link between two nodes, or the removal
of a ring node is an event that triggers protection switching.  In a
simple ring, the result is a broken ring.  However, if a ring has
express links, then it may be able to converge to a smaller ring with
protection.

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

5.  Ring OAM

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

Each ring node also sends OAM messages over each direction of its
ring LSP.  This is a multi-hop OAM to check LSP liveness; typically,
BFD would be used for this.  Each node chooses the hello interval,
the choice of which should be based on the size of the ring (as each
node would have to send out twice that many hello messages every
interval) and the desired failure detection time.

6.  Advanced Topics

6.1.  Beyond the Ring

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

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

```
         N0 ___     R0 . . . R1
              \\   .             .
               R7                 R2
               .       Ring        .
               .        17         .
               .                   .
               R6                 R3 ----- N1
                 .               .
                  R5 . . . R4
```
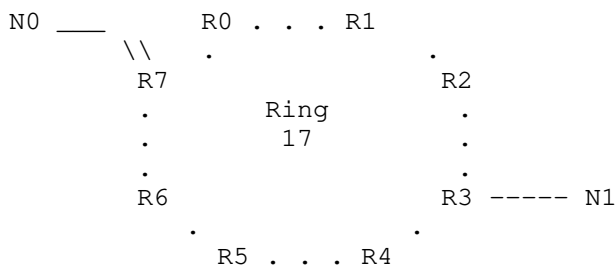
Figure 4: Beyond the Ring

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

1.  N1 sends an "egress label" L0 for its loopback N1 to R3 and
    inserts a "pop L0 and forward" entry in its LFIB.

2.  R3 sends a label L1 for N1 to R7 over the targeted LDP session
    and inserts a "swap L1 with L0" in its LFIB.

3.  R7 sends label L2 for N1 to N0 and inserts a "swap L2 with L1"
    entry in its LFIB.

4.  N0 inserts a "push L2" entry in its LFIB for traffic destined to
    N1.

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

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

6.2.  Half-rings

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

6.3.  Hub Node Resilience

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

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

7.  Security Considerations

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

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

8.  Acknowledgments

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

9.  IANA Considerations

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

10.  References

10.1.  Normative References

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

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

10.2.  Informative References

   [IEEE.802.1D_2004]
              IEEE, "IEEE Standard for Local and metropolitan area
              networks: Media Access Control (MAC) Bridges", IEEE
              802.1D-2004, DOI 10.1109/ieeestd.2004.94569, July 2004,
              <http://ieeexplore.ieee.org/servlet/opac?punumber=9155>.

   [RFC3209]  Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V.,
              and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP
              Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001,
              <https://www.rfc-editor.org/info/rfc3209>.

   [RFC3630]  Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering
              (TE) Extensions to OSPF Version 2", RFC 3630,
              DOI 10.17487/RFC3630, September 2003,
              <https://www.rfc-editor.org/info/rfc3630>.

   [RFC5036]  Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed.,
              "LDP Specification", RFC 5036, DOI 10.17487/RFC5036,
              October 2007, <https://www.rfc-editor.org/info/rfc5036>.

   [RFC5305]  Li, T. and H. Smit, "IS-IS Extensions for Traffic
              Engineering", RFC 5305, DOI 10.17487/RFC5305, October
              2008, <https://www.rfc-editor.org/info/rfc5305>.

   [RFC8029]  Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N.,
              Aldrin, S., and M. Chen, "Detecting Multiprotocol Label
              Switched (MPLS) Data-Plane Failures", RFC 8029,
              DOI 10.17487/RFC8029, March 2017,
              <https://www.rfc-editor.org/info/rfc8029>.

   [RFC8402]  Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
              Decraene, B., Litkowski, S., and R. Shakir, "Segment
              Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
              July 2018, <https://www.rfc-editor.org/info/rfc8402>.

Authors' Addresses

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

   Email: kireeti.ietf@gmail.com


   Luis M. Contreras
   Telefonica
   Ronda de la Comunicacion
   Sur-3 building, 3rd floor
   Madrid  28050
   Spain

   Email: luismiguel.contrerasmurillo@telefonica.com
   URI:   http://lmcontreras.com

PCE Working Group                                    D. Dhody, Ed.
Internet-Draft                                   Huawei Technologies
Intended status: Standards Track                       J. Hardwick
Expires: April 30, 2017                                 Metaswitch
                                                        V. Beeram
                                                  Juniper Networks
                                                       J. Tantsura
                                                  October 27, 2016

        A YANG Data Model for Path Computation Element Communications Protocol
                                (PCEP)
                      draft-ietf-pce-pcep-yang-01

Abstract

   This document defines a YANG data model for the management of Path
   Computation Element communications Protocol (PCEP) for communications
   between a Path Computation Client (PCC) and a Path Computation
   Element (PCE), or between two PCEs.  The data model includes
   configuration data and state data (status information and counters
   for the collection of statistics).

Status of This Memo

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

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

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

   This Internet-Draft will expire on April 30, 2017.

Table of Contents

1.  Introduction

   The Path Computation Element (PCE) defined in [RFC4655] is an entity
   that is capable of computing a network path or route based on a
   network graph, and applying computational constraints.  A Path

Computation Client (PCC) may make requests to a PCE for paths to be computed.

PCEP is the communication protocol between a PCC and PCE and is defined in [RFC5440].  PCEP interactions include path computation requests and path computation replies as well as notifications of specific states related to the use of a PCE in the context of Multiprotocol Label Switching (MPLS) and Generalized MPLS (GMPLS) Traffic Engineering (TE).  [I-D.ietf-pce-stateful-pce] specifies extensions to PCEP to enable stateful control of MPLS TE LSPs.

This document defines a YANG [RFC6020] data model for the management of PCEP speakers.  It is important to establish a common data model for how PCEP speakers are identified, configured, and monitored.  The data model includes configuration data and state data (status information and counters for the collection of statistics).

This document contains a specification of the PCEP YANG module, "ietf-pcep" which provides the PCEP [RFC5440] data model.

2.  Requirements Language

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

3.  Terminology and Notation

This document uses the terminology defined in [RFC4655] and [RFC5440].  In particular, it uses the following acronyms.

o  Path Computation Request message (PCReq).

o  Path Computation Reply message (PCRep).

o  Notification message (PCNtf).

o  Error message (PCErr).

o  Request Parameters object (RP).

o  Synchronization Vector object (SVEC).

o  Explicit Route object (ERO).

This document also uses the following terms defined in [RFC7420]:

o  PCEP entity: a local PCEP speaker.

   o  PCEP peer: to refer to a remote PCEP speaker.

   o  PCEP speaker: where it is not necessary to distinguish between
      local and remote.

   Further, this document also uses the following terms defined in
   [I-D.ietf-pce-stateful-pce] :

   o  Stateful PCE, Passive Stateful PCE, Active Stateful PCE

   o  Delegation, Revocation, Redelegation

   o  LSP State Report, Path Computation Report message (PCRpt).

   o  LSP State Update, Path Computation Update message (PCUpd).

   [I-D.ietf-pce-pce-initiated-lsp] :

   o  PCE-initiated LSP, Path Computation LSP Initiate Message
      (PCInitiate).

   [I-D.ietf-pce-lsp-setup-type] :

   o  Path Setup Type (PST).

   [I-D.ietf-pce-segment-routing] :

   o  Segment Routing (SR).

3.1.  Tree Diagrams

   A graphical representation of the complete data tree is presented in
   Section 5.  The meaning of the symbols in these diagrams is as
   follows and as per [I-D.ietf-netmod-rfc6087bis]:

   o  Brackets "[" and "]" enclose list keys.

   o  Abbreviations before data node names: "rw" means configuration
      (read-write) and "ro" state data (read-only).

   o  Symbols after data node names: "?" means an optional node, "!"
      means a presence container, and "*" denotes a list and leaf-list.

   o  Parentheses enclose choice and case nodes, and case nodes are also
      marked with a colon (":").

   o  Ellipsis ("...") stands for contents of subtrees that are not
      shown.

3.2.  Prefixes in Data Node Names

   In this document, names of data nodes and other data model objects
   are often used without a prefix, as long as it is clear from the
   context in which YANG module each name is defined.  Otherwise, names
   are prefixed using the standard prefix associated with the
   corresponding YANG module, as shown in Table 1.

```
+-----------+----------------+--------------------------------+
| Prefix    | YANG module    | Reference                      |
+-----------+----------------+--------------------------------+
| yang      | ietf-yang-types | [RFC6991]                     |
| inet      | ietf-inet-types | [RFC6991]                     |
| te        | ietf-te        | [I-D.ietf-teas-yang-te]        |
| te-types  | ietf-te-types  | [I-D.ietf-teas-yang-te]        |
| key-chain | ietf-key-chain | [I-D.ietf-rtgwg-yang-key-chain] |
+-----------+----------------+--------------------------------+
```

            Table 1: Prefixes and corresponding YANG modules

4.  Objectives

   This section describes some of the design objectives for the model:

   o  In case of existing implementations, it needs to map the data
      model defined in this document to their proprietary native data
      model.  To facilitate such mappings, the data model should be
      simple.

   o  The data model should be suitable for new implementations to use
      as is.

   o  Mapping to the PCEP MIB Module should be clear.

   o  The data model should allow for static configurations of peers.

   o  The data model should include read-only counters in order to
      gather statistics for sent and received PCEP messages, received
      messages with errors, and messages that could not be sent due to
      errors.

   o  It should be fairly straightforward to augment the base data model
      for advanced PCE features.

5.  The Design of PCEP Data Model

   The module, "ietf-pcep", defines the basic components of a PCE
   speaker.

```
module: ietf-pcep
   +--rw pcep!
   │  +--rw entity
   │     +--rw addr                         inet:ip-address
   │     +--rw enabled?                     boolean
   │     +--rw role                         pcep-role
   │     +--rw description?                 string
   │     +--rw speaker-entity-id?           string {stateful-sync-opt}?
   │     +--rw domain
   │     │  +--rw domain* [domain-type domain]
   │     │     +--rw domain-type    domain-type
   │     │     +--rw domain         domain
   │     +--rw capability
   │     │  +--rw gmpls?                 boolean {gmpls}?
   │     │  +--rw bi-dir?                boolean
   │     │  +--rw diverse?               boolean
   │     │  +--rw load-balance?          boolean
   │     │  +--rw synchronize?           boolean {svec}?
   │     │  +--rw objective-function?    boolean {objective-function}?
   │     │  +--rw add-path-constraint?   boolean
   │     │  +--rw prioritization?        boolean
   │     │  +--rw multi-request?         boolean
   │     │  +--rw gco?                   boolean {gco}?
   │     │  +--rw p2mp?                  boolean {p2mp}?
   │     │  +--rw stateful {stateful}?
   │     │  │  +--rw enabled?               boolean
   │     │  │  +--rw active?                boolean
   │     │  │  +--rw pce-initiated?         boolean {pce-initiated}?
   │     │  │  +--rw include-db-ver?        boolean {stateful-sync-opt}?
   │     │  │  +--rw trigger-resync?        boolean {stateful-sync-opt}?
   │     │  │  +--rw trigger-initial-sync?  boolean {stateful-sync-opt}?
   │     │  │  +--rw incremental-sync?      boolean {stateful-sync-opt}?
   │     │  +--rw sr {sr}?
   │     │     +--rw enabled?   boolean
   │     +--rw pce-info
   │     │  +--rw scope
   │     │  │  +--rw intra-area-scope?           boolean
   │     │  │  +--rw intra-area-pref?            uint8
   │     │  │  +--rw inter-area-scope?           boolean
   │     │  │  +--rw inter-area-scope-default?   boolean
   │     │  │  +--rw inter-area-pref?            uint8
   │     │  │  +--rw inter-as-scope?             boolean
   │     │  │  +--rw inter-as-scope-default?     boolean
```

```
│  │  │  +--rw inter-as-pref?              uint8
│  │  │  +--rw inter-layer-scope?         boolean
│  │  │  +--rw inter-layer-pref?          uint8
│  │  +--rw neigh-domains
│  │  │  +--rw domain* [domain-type domain]
│  │  │     +--rw domain-type   domain-type
│  │  │     +--rw domain        domain
│  │  +--rw path-key {path-key}?
│  │     +--rw enabled?        boolean
│  │     +--rw discard-timer?  uint32
│  │     +--rw reuse-time?     uint32
│  │     +--rw pce-id?         inet:ip-address
│  +--rw (auth-type-selection)?
│  │  +--:(auth-key-chain)
│  │  │  +--rw key-chain?                         key-chain:key-chain-ref
│  │  +--:(auth-key)
│  │  │  +--rw key?                               string
│  │  │  +--rw crypto-algorithm
│  │  │     +--rw (algorithm)?
│  │  │        +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
│  │  │        │  +--rw hmac-sha1-12?         empty
│  │  │        +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
│  │  │        │  +--rw aes-cmac-prf-128?     empty
│  │  │        +--:(md5)
│  │  │        │  +--rw md5?                  empty
│  │  │        +--:(sha-1)
│  │  │        │  +--rw sha-1?                empty
│  │  │        +--:(hmac-sha-1)
│  │  │        │  +--rw hmac-sha-1?           empty
│  │  │        +--:(hmac-sha-256)
│  │  │        │  +--rw hmac-sha-256?         empty
│  │  │        +--:(hmac-sha-384)
│  │  │        │  +--rw hmac-sha-384?         empty
│  │  │        +--:(hmac-sha-512)
│  │  │        │  +--rw hmac-sha-512?         empty
│  │  │        +--:(clear-text) {clear-text}?
│  │  │        │  +--rw clear-text?           empty
│  │  │        +--:(replay-protection-only) {replay-protection-only}?
│  │  │           +--rw replay-protection-only?   empty
│  │  +--:(auth-tls) {tls}?
│  │     +--rw tls
│  +--rw connect-timer?              uint32
│  +--rw connect-max-retry?          uint32
│  +--rw init-backoff-timer?         uint32
│  +--rw max-backoff-timer?          uint32
│  +--rw open-wait-timer?            uint32
│  +--rw keep-wait-timer?            uint32
│  +--rw keep-alive-timer?           uint32
```

```
│        +--rw dead-timer?                 uint32
│        +--rw allow-negotiation?          boolean
│        +--rw max-keep-alive-timer?       uint32
│        +--rw max-dead-timer?             uint32
│        +--rw min-keep-alive-timer?       uint32
│        +--rw min-dead-timer?             uint32
│        +--rw sync-timer?                 uint32 {svec}?
│        +--rw request-timer?              uint32
│        +--rw max-sessions?               uint32
│        +--rw max-unknown-reqs?           uint32
│        +--rw max-unknown-msgs?           uint32
│        +--rw pcep-notification-max-rate  uint32
│        +--rw stateful-parameter {stateful}?
│        │  +--rw state-timeout?        uint32
│        │  +--rw redelegation-timeout? uint32
│        │  +--rw rpt-non-pcep-lsp?     boolean
│        +--rw of-list {objective-function}?
│        │  +--rw objective-function* [of]
│        │     +--rw of    objective-function
│        +--rw peers
│           +--rw peer* [addr]
│              +--rw addr               inet:ip-address
│              +--rw description?       string
│              +--rw domain
│              │  +--rw domain* [domain-type domain]
│              │     +--rw domain-type    domain-type
│              │     +--rw domain         domain
│              +--rw capability
│              │  +--rw gmpls?               boolean {gmpls}?
│              │  +--rw bi-dir?              boolean
│              │  +--rw diverse?             boolean
│              │  +--rw load-balance?        boolean
│              │  +--rw synchronize?         boolean {svec}?
│              │  +--rw objective-function?  boolean {objective-function}?
│              │  +--rw add-path-constraint? boolean
│              │  +--rw prioritization?      boolean
│              │  +--rw multi-request?       boolean
│              │  +--rw gco?                 boolean {gco}?
│              │  +--rw p2mp?                boolean {p2mp}?
│              │  +--rw stateful {stateful}?
│              │  │  +--rw enabled?             boolean
│              │  │  +--rw active?              boolean
│              │  │  +--rw pce-initiated?       boolean {pce-initiated}?
│              │  │  +--rw include-db-ver?      boolean {stateful-sync-opt}?
│              │  │  +--rw trigger-resync?      boolean {stateful-sync-opt}?
│              │  │  +--rw trigger-initial-sync? boolean {stateful-sync-opt}?
│              │  │  +--rw incremental-sync?    boolean {stateful-sync-opt}?
│              │  +--rw sr {sr}?
```

```
   │                │    │  +--rw enabled?    boolean
   │                +--rw scope
   │                │  +--rw intra-area-scope?           boolean
   │                │  +--rw intra-area-pref?            uint8
   │                │  +--rw inter-area-scope?           boolean
   │                │  +--rw inter-area-scope-default?   boolean
   │                │  +--rw inter-area-pref?            uint8
   │                │  +--rw inter-as-scope?             boolean
   │                │  +--rw inter-as-scope-default?     boolean
   │                │  +--rw inter-as-pref?              uint8
   │                │  +--rw inter-layer-scope?          boolean
   │                │  +--rw inter-layer-pref?           uint8
   │                +--rw neigh-domains
   │                │  +--rw domain* [domain-type domain]
   │                │     +--rw domain-type    domain-type
   │                │     +--rw domain         domain
   │                +--rw delegation-pref?    uint8 {stateful}?
   │                +--rw (auth-type-selection)?
   │                │  +--:(auth-key-chain)
   │                │  │  +--rw key-chain?             key-chain:key-chain-ref
   │                │  +--:(auth-key)
   │                │  │  +--rw key?                  string
   │                │  │  +--rw crypto-algorithm
   │                │  │     +--rw (algorithm)?
   │                │  │        +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
   │                │  │        │  +--rw hmac-sha1-12?            empty
   │                │  │        +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
   │                │  │        │  +--rw aes-cmac-prf-128?        empty
   │                │  │        +--:(md5)
   │                │  │        │  +--rw md5?                     empty
   │                │  │        +--:(sha-1)
   │                │  │        │  +--rw sha-1?                   empty
   │                │  │        +--:(hmac-sha-1)
   │                │  │        │  +--rw hmac-sha-1?              empty
   │                │  │        +--:(hmac-sha-256)
   │                │  │        │  +--rw hmac-sha-256?            empty
   │                │  │        +--:(hmac-sha-384)
   │                │  │        │  +--rw hmac-sha-384?            empty
   │                │  │        +--:(hmac-sha-512)
   │                │  │        │  +--rw hmac-sha-512?            empty
   │                │  │        +--:(clear-text) {clear-text}?
   │                │  │        │  +--rw clear-text?              empty
   │                │  │        +--:(replay-protection-only) {replay-protection-only}
?
   │                │  │              +--rw replay-protection-only?    empty
   │                │  +--:(auth-tls) {tls}?
   │                │     +--rw tls
   │                +--rw of-list {objective-function}?
   │                   +--rw objective-function* [of]
```

```
   |                  +--rw of     objective-function
 +--ro pcep-state
    +--ro entity
       +--ro addr?                   inet:ip-address
       +--ro index?                  uint32
       +--ro admin-status?           pcep-admin-status
       +--ro oper-status?            pcep-admin-status
       +--ro role?                   pcep-role
       +--ro description?            string
       +--ro speaker-entity-id?      string {stateful-sync-opt}?
       +--ro domain
       |  +--ro domain* [domain-type domain]
       |     +--ro domain-type    domain-type
       |     +--ro domain         domain
       +--ro capability
       |  +--ro gmpls?                  boolean {gmpls}?
       |  +--ro bi-dir?                 boolean
       |  +--ro diverse?                boolean
       |  +--ro load-balance?           boolean
       |  +--ro synchronize?            boolean {svec}?
       |  +--ro objective-function?     boolean {objective-function}?
       |  +--ro add-path-constraint?    boolean
       |  +--ro prioritization?         boolean
       |  +--ro multi-request?          boolean
       |  +--ro gco?                    boolean {gco}?
       |  +--ro p2mp?                   boolean {p2mp}?
       |  +--ro stateful {stateful}?
       |  |  +--ro enabled?               boolean
       |  |  +--ro active?                boolean
       |  |  +--ro pce-initiated?         boolean {pce-initiated}?
       |  |  +--ro include-db-ver?        boolean {stateful-sync-opt}?
       |  |  +--ro trigger-resync?        boolean {stateful-sync-opt}?
       |  |  +--ro trigger-initial-sync?  boolean {stateful-sync-opt}?
       |  |  +--ro incremental-sync?      boolean {stateful-sync-opt}?
       |  +--ro sr {sr}?
       |     +--ro enabled?    boolean
       +--ro pce-info
       |  +--ro scope
       |  |  +--ro intra-area-scope?         boolean
       |  |  +--ro intra-area-pref?          uint8
       |  |  +--ro inter-area-scope?         boolean
       |  |  +--ro inter-area-scope-default? boolean
       |  |  +--ro inter-area-pref?          uint8
       |  |  +--ro inter-as-scope?           boolean
       |  |  +--ro inter-as-scope-default?   boolean
       |  |  +--ro inter-as-pref?            uint8
       |  |  +--ro inter-layer-scope?        boolean
       |  |  +--ro inter-layer-pref?         uint8
```

```
          | +--ro neigh-domains
          |   +--ro domain* [domain-type domain]
          |     +--ro domain-type    domain-type
          |     +--ro domain         domain
          +--ro path-key {path-key}?
          |   +--ro enabled?        boolean
          |   +--ro discard-timer?  uint32
          |   +--ro reuse-time?     uint32
          |   +--ro pce-id?         inet:ip-address
        +--ro (auth-type-selection)?
        |  +--:(auth-key-chain)
        |  |  +--ro key-chain?              key-chain:key-chain-ref
        |  +--:(auth-key)
        |  |  +--ro key?                    string
        |  |  +--ro crypto-algorithm
        |  |     +--ro (algorithm)?
        |  |        +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
        |  |        |  +--ro hmac-sha1-12?          empty
        |  |        +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
        |  |        |  +--ro aes-cmac-prf-128?      empty
        |  |        +--:(md5)
        |  |        |  +--ro md5?                   empty
        |  |        +--:(sha-1)
        |  |        |  +--ro sha-1?                 empty
        |  |        +--:(hmac-sha-1)
        |  |        |  +--ro hmac-sha-1?            empty
        |  |        +--:(hmac-sha-256)
        |  |        |  +--ro hmac-sha-256?          empty
        |  |        +--:(hmac-sha-384)
        |  |        |  +--ro hmac-sha-384?          empty
        |  |        +--:(hmac-sha-512)
        |  |        |  +--ro hmac-sha-512?          empty
        |  |        +--:(clear-text) {clear-text}?
        |  |        |  +--ro clear-text?            empty
        |  |        +--:(replay-protection-only) {replay-protection-only}?
        |  |           +--ro replay-protection-only?  empty
        |  +--:(auth-tls) {tls}?
        |     +--ro tls
        +--ro connect-timer?        uint32
        +--ro connect-max-retry?    uint32
        +--ro init-backoff-timer?   uint32
        +--ro max-backoff-timer?    uint32
        +--ro open-wait-timer?      uint32
        +--ro keep-wait-timer?      uint32
        +--ro keep-alive-timer?     uint32
        +--ro dead-timer?           uint32
        +--ro allow-negotiation?    boolean
        +--ro max-keep-alive-timer? uint32
```

```
         +--ro max-dead-timer?        uint32
         +--ro min-keep-alive-timer?  uint32
         +--ro min-dead-timer?        uint32
         +--ro sync-timer?            uint32 {svec}?
         +--ro request-timer?         uint32
         +--ro max-sessions?          uint32
         +--ro max-unknown-reqs?      uint32
         +--ro max-unknown-msgs?      uint32
         +--ro stateful-parameter {stateful}?
         |  +--ro state-timeout?         uint32
         |  +--ro redelegation-timeout?  uint32
         |  +--ro rpt-non-pcep-lsp?      boolean
         +--ro lsp-db {stateful}?
         |  +--ro db-ver?             uint64 {stateful-sync-opt}?
         |  +--ro association-list* [id source global-source extended-id]
         |  |  +--ro type?          assoc-type
         |  |  +--ro id             uint16
         |  |  +--ro source         inet:ip-address
         |  |  +--ro global-source  uint32
         |  |  +--ro extended-id    string
         |  |  +--ro lsp* [plsp-id pcc-id]
         |  |     +--ro plsp-id   -> /pcep-state/entity/lsp-db/lsp/plsp-id
         |  |     +--ro pcc-id    -> /pcep-state/entity/lsp-db/lsp/pcc-id
         |  +--ro lsp* [plsp-id pcc-id]
         |     +--ro plsp-id                uint32
         |     +--ro pcc-id                 inet:ip-address
         |     +--ro lsp-ref
         |     |  +--ro source?                 -> /te:te/lsps-state/lsp/source
         |     |  +--ro destination?            -> /te:te/lsps-state/lsp/destinati
on
         |     |  +--ro tunnel-id?              -> /te:te/lsps-state/lsp/tunnel-id
         |     |  +--ro lsp-id?                 -> /te:te/lsps-state/lsp/lsp-id
         |     |  +--ro extended-tunnel-id?  -> /te:te/lsps-state/lsp/extended-
tunnel-id
         |     |  +--ro type?                   -> /te:te/lsps-state/lsp/type
         |     +--ro admin-state?        boolean
         |     +--ro operational-state?  operational-state
         |     +--ro delegated
         |     |  +--ro enabled?   boolean
         |     |  +--ro pce?       -> /pcep-state/entity/peers/peer/addr
         |     |  +--ro srp-id?    uint32
         |     +--ro initiation {pce-initiated}?
         |     |  +--ro enabled?   boolean
         |     |  +--ro pce?       -> /pcep-state/entity/peers/peer/addr
         |     +--ro symbolic-path-name?  string
         |     +--ro last-error?          lsp-error
         |     +--ro pst?                 pst
         |     +--ro association-list* [id source global-source extended-id]
         |        +--ro id                -> /pcep-state/entity/lsp-db/associatio
n-list/id
         |        +--ro source            -> /pcep-state/entity/lsp-db/associatio
n-list/source
```

```
      |              +--ro global-source    -> /pcep-state/entity/lsp-db/associatio
   n-list/global-source
      |              +--ro extended-id      -> /pcep-state/entity/lsp-db/associatio
   n-list/extended-id
         +--ro path-keys {path-key}?
         |  +--ro path-keys* [path-key]
         |     +--ro path-key        uint16
         |     +--ro cps
         |     |  +--ro explicit-route-objects* [index]
         |     |     +--ro index                  uint8
         |     |     +--ro explicit-route-usage?   identityref
         |     |     +--ro (type)?
         |     |        +--:(ipv4-address)
         |     |        |  +--ro v4-address?            inet:ipv4-address
         |     |        |  +--ro v4-prefix-length?      uint8
         |     |        |  +--ro v4-loose?              boolean
         |     |        +--:(ipv6-address)
         |     |        |  +--ro v6-address?            inet:ipv6-address
         |     |        |  +--ro v6-prefix-length?      uint8
         |     |        |  +--ro v6-loose?              boolean
         |     |        +--:(as-number)
         |     |        |  +--ro as-number?             uint16
         |     |        +--:(unnumbered-link)
         |     |        |  +--ro router-id?             inet:ip-address
         |     |        |  +--ro interface-id?          uint32
         |     |        +--:(label)
         |     |           +--ro value?                 uint32
         |     +--ro pcc-original?   -> /pcep-state/entity/peers/peer/addr
         |     +--ro req-id?         uint32
         |     +--ro retrieved?      boolean
         |     +--ro pcc-retrieved?  -> /pcep-state/entity/peers/peer/addr
         |     +--ro creation-time?  yang:timestamp
         |     +--ro discard-time?   uint32
         |     +--ro reuse-time?     uint32
         +--ro of-list {objective-function}?
         |  +--ro objective-function* [of]
         |     +--ro of    objective-function
         +--ro peers
            +--ro peer* [addr]
               +--ro addr                   inet:ip-address
               +--ro role?                  pcep-role
               +--ro domain
               |  +--ro domain* [domain-type domain]
               |     +--ro domain-type    domain-type
               |     +--ro domain         domain
               +--ro capability
               |  +--ro gmpls?                 boolean {gmpls}?
               |  +--ro bi-dir?                boolean
               |  +--ro diverse?               boolean
               |  +--ro load-balance?          boolean
```

```
         │    +--ro synchronize?          boolean {svec}?
         │    +--ro objective-function?   boolean {objective-function}?
         │    +--ro add-path-constraint?  boolean
         │    +--ro prioritization?       boolean
         │    +--ro multi-request?        boolean
         │    +--ro gco?                  boolean {gco}?
         │    +--ro p2mp?                 boolean {p2mp}?
         │    +--ro stateful {stateful}?
         │    │  +--ro enabled?                boolean
         │    │  +--ro active?                 boolean
         │    │  +--ro pce-initiated?          boolean {pce-initiated}?
         │    │  +--ro include-db-ver?         boolean {stateful-sync-opt}?
         │    │  +--ro trigger-resync?         boolean {stateful-sync-opt}?
         │    │  +--ro trigger-initial-sync?   boolean {stateful-sync-opt}?
         │    │  +--ro incremental-sync?       boolean {stateful-sync-opt}?
         │    +--ro sr {sr}?
         │       +--ro enabled?    boolean
         +--ro pce-info
         │  +--ro scope
         │  │  +--ro intra-area-scope?         boolean
         │  │  +--ro intra-area-pref?          uint8
         │  │  +--ro inter-area-scope?         boolean
         │  │  +--ro inter-area-scope-default? boolean
         │  │  +--ro inter-area-pref?          uint8
         │  │  +--ro inter-as-scope?           boolean
         │  │  +--ro inter-as-scope-default?   boolean
         │  │  +--ro inter-as-pref?            uint8
         │  │  +--ro inter-layer-scope?        boolean
         │  │  +--ro inter-layer-pref?         uint8
         │  +--ro neigh-domains
         │     +--ro domain* [domain-type domain]
         │        +--ro domain-type    domain-type
         │        +--ro domain         domain
         +--ro delegation-pref?      uint8 {stateful}?
         +--ro (auth-type-selection)?
         │  +--:(auth-key-chain)
         │  │  +--ro key-chain?              key-chain:key-chain-ref
         │  +--:(auth-key)
         │  │  +--ro key?                  string
         │  │  +--ro crypto-algorithm
         │  │     +--ro (algorithm)?
         │  │        +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
         │  │        │  +--ro hmac-sha1-12?          empty
         │  │        +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
         │  │        │  +--ro aes-cmac-prf-128?      empty
         │  │        +--:(md5)
         │  │        │  +--ro md5?                   empty
         │  │        +--:(sha-1)
```

```
      |  |            | +--ro sha-1?                   empty
      |  |            +--:(hmac-sha-1)
      |  |            | +--ro hmac-sha-1?              empty
      |  |            +--:(hmac-sha-256)
      |  |            | +--ro hmac-sha-256?            empty
      |  |            +--:(hmac-sha-384)
      |  |            | +--ro hmac-sha-384?            empty
      |  |            +--:(hmac-sha-512)
      |  |            | +--ro hmac-sha-512?            empty
      |  |            +--:(clear-text) {clear-text}?
      |  |            | +--ro clear-text?              empty
      |  |            +--:(replay-protection-only) {replay-protection-only}
?
      |  |               +--ro replay-protection-only? empty
      |  +--:(auth-tls) {tls}?
      |     +--ro tls
      +--ro of-list {objective-function}?
      |  +--ro objective-function* [of]
      |     +--ro of    objective-function
      +--ro discontinuity-time?     yang:timestamp
      +--ro initiate-session?       boolean
      +--ro session-exists?         boolean
      +--ro num-sess-setup-ok?      yang:counter32
      +--ro num-sess-setup-fail?    yang:counter32
      +--ro session-up-time?        yang:timestamp
      +--ro session-fail-time?      yang:timestamp
      +--ro session-fail-up-time?   yang:timestamp
      +--ro pcep-stats
         +--ro avg-rsp-time?              uint32
         +--ro lwm-rsp-time?              uint32
         +--ro hwm-rsp-time?              uint32
         +--ro num-pcreq-sent?            yang:counter32
         +--ro num-pcreq-rcvd?            yang:counter32
         +--ro num-pcrep-sent?            yang:counter32
         +--ro num-pcrep-rcvd?            yang:counter32
         +--ro num-pcerr-sent?            yang:counter32
         +--ro num-pcerr-rcvd?            yang:counter32
         +--ro num-pcntf-sent?            yang:counter32
         +--ro num-pcntf-rcvd?            yang:counter32
         +--ro num-keepalive-sent?        yang:counter32
         +--ro num-keepalive-rcvd?        yang:counter32
         +--ro num-unknown-rcvd?          yang:counter32
         +--ro num-corrupt-rcvd?          yang:counter32
         +--ro num-req-sent?              yang:counter32
         +--ro num-req-sent-pend-rep?     yang:counter32
         +--ro num-req-sent-ero-rcvd?     yang:counter32
         +--ro num-req-sent-nopath-rcvd?  yang:counter32
         +--ro num-req-sent-cancel-rcvd?  yang:counter32
         +--ro num-req-sent-error-rcvd?   yang:counter32
```

```
        │  +--ro num-req-sent-timeout?        yang:counter32
        │  +--ro num-req-sent-cancel-sent?    yang:counter32
        │  +--ro num-req-rcvd?                yang:counter32
        │  +--ro num-req-rcvd-pend-rep?       yang:counter32
        │  +--ro num-req-rcvd-ero-sent?       yang:counter32
        │  +--ro num-req-rcvd-nopath-sent?    yang:counter32
        │  +--ro num-req-rcvd-cancel-sent?    yang:counter32
        │  +--ro num-req-rcvd-error-sent?     yang:counter32
        │  +--ro num-req-rcvd-cancel-rcvd?    yang:counter32
        │  +--ro num-rep-rcvd-unknown?        yang:counter32
        │  +--ro num-req-rcvd-unknown?        yang:counter32
        │  +--ro svec {svec}?
        │  │  +--ro num-svec-sent?        yang:counter32
        │  │  +--ro num-svec-req-sent?    yang:counter32
        │  │  +--ro num-svec-rcvd?        yang:counter32
        │  │  +--ro num-svec-req-rcvd?    yang:counter32
        │  +--ro stateful {stateful}?
        │  │  +--ro num-pcrpt-sent?               yang:counter32
        │  │  +--ro num-pcrpt-rcvd?               yang:counter32
        │  │  +--ro num-pcupd-sent?               yang:counter32
        │  │  +--ro num-pcupd-rcvd?               yang:counter32
        │  │  +--ro num-rpt-sent?                 yang:counter32
        │  │  +--ro num-rpt-rcvd?                 yang:counter32
        │  │  +--ro num-rpt-rcvd-error-sent?      yang:counter32
        │  │  +--ro num-upd-sent?                 yang:counter32
        │  │  +--ro num-upd-rcvd?                 yang:counter32
        │  │  +--ro num-upd-rcvd-unknown?         yang:counter32
        │  │  +--ro num-upd-rcvd-undelegated?     yang:counter32
        │  │  +--ro num-upd-rcvd-error-sent?      yang:counter32
        │  │  +--ro initiation {pce-initiated}?
        │  │     +--ro num-pcinitiate-sent?            yang:counter32
        │  │     +--ro num-pcinitiate-rcvd?            yang:counter32
        │  │     +--ro num-initiate-sent?              yang:counter32
        │  │     +--ro num-initiate-rcvd?              yang:counter32
        │  │     +--ro num-initiate-rcvd-error-sent?   yang:counter32
        │  +--ro path-key {path-key}?
        │  │  +--ro num-unknown-path-key?      yang:counter32
        │  │  +--ro num-exp-path-key?          yang:counter32
        │  │  +--ro num-dup-path-key?          yang:counter32
        │  │  +--ro num-path-key-no-attempt?   yang:counter32
        │  +--ro num-req-sent-closed?        yang:counter32
        │  +--ro num-req-rcvd-closed?        yang:counter32
        +--ro sessions
           +--ro session* [initiator]
              +--ro initiator                 pcep-initiator
              +--ro state-last-change?        yang:timestamp
              +--ro state?                    pcep-sess-state
              +--ro session-creation?         yang:timestamp
```

```
                    +--ro connect-retry?          yang:counter32
                    +--ro local-id?               uint32
                    +--ro remote-id?              uint32
                    +--ro keepalive-timer?        uint32
                    +--ro peer-keepalive-timer?   uint32
                    +--ro dead-timer?             uint32
                    +--ro peer-dead-timer?        uint32
                    +--ro ka-hold-time-rem?       uint32
                    +--ro overloaded?             boolean
                    +--ro overload-time?          uint32
                    +--ro peer-overloaded?        boolean
                    +--ro peer-overload-time?     uint32
                    +--ro lspdb-sync?             sync-state {stateful}?
                    +--ro recv-db-ver?            uint64 {stateful,stateful-syn
c-opt}?
                    +--ro of-list {objective-function}?
                    |  +--ro objective-function* [of]
                    |     +--ro of    objective-function
                    +--ro speaker-entity-id?      string {stateful-sync-opt}?
                    +--ro discontinuity-time?     yang:timestamp
                    +--ro pcep-stats
                       +--ro avg-rsp-time?                uint32
                       +--ro lwm-rsp-time?                uint32
                       +--ro hwm-rsp-time?                uint32
                       +--ro num-pcreq-sent?              yang:counter32
                       +--ro num-pcreq-rcvd?              yang:counter32
                       +--ro num-pcrep-sent?              yang:counter32
                       +--ro num-pcrep-rcvd?              yang:counter32
                       +--ro num-pcerr-sent?              yang:counter32
                       +--ro num-pcerr-rcvd?              yang:counter32
                       +--ro num-pcntf-sent?              yang:counter32
                       +--ro num-pcntf-rcvd?              yang:counter32
                       +--ro num-keepalive-sent?          yang:counter32
                       +--ro num-keepalive-rcvd?          yang:counter32
                       +--ro num-unknown-rcvd?            yang:counter32
                       +--ro num-corrupt-rcvd?            yang:counter32
                       +--ro num-req-sent?                yang:counter32
                       +--ro num-req-sent-pend-rep?       yang:counter32
                       +--ro num-req-sent-ero-rcvd?       yang:counter32
                       +--ro num-req-sent-nopath-rcvd?    yang:counter32
                       +--ro num-req-sent-cancel-rcvd?    yang:counter32
                       +--ro num-req-sent-error-rcvd?     yang:counter32
                       +--ro num-req-sent-timeout?        yang:counter32
                       +--ro num-req-sent-cancel-sent?    yang:counter32
                       +--ro num-req-rcvd?                yang:counter32
                       +--ro num-req-rcvd-pend-rep?       yang:counter32
                       +--ro num-req-rcvd-ero-sent?       yang:counter32
                       +--ro num-req-rcvd-nopath-sent?    yang:counter32
                       +--ro num-req-rcvd-cancel-sent?    yang:counter32
```

```
                            +--ro num-req-rcvd-error-sent?    yang:counter32
                            +--ro num-req-rcvd-cancel-rcvd?   yang:counter32
                            +--ro num-rep-rcvd-unknown?       yang:counter32
                            +--ro num-req-rcvd-unknown?       yang:counter32
                            +--ro svec {svec}?
                            │  +--ro num-svec-sent?       yang:counter32
                            │  +--ro num-svec-req-sent?   yang:counter32
                            │  +--ro num-svec-rcvd?       yang:counter32
                            │  +--ro num-svec-req-rcvd?   yang:counter32
                            +--ro stateful {stateful}?
                            │  +--ro num-pcrpt-sent?               yang:counter32
                            │  +--ro num-pcrpt-rcvd?               yang:counter32
                            │  +--ro num-pcupd-sent?               yang:counter32
                            │  +--ro num-pcupd-rcvd?               yang:counter32
                            │  +--ro num-rpt-sent?                 yang:counter32
                            │  +--ro num-rpt-rcvd?                 yang:counter32
                            │  +--ro num-rpt-rcvd-error-sent?      yang:counter32
                            │  +--ro num-upd-sent?                 yang:counter32
                            │  +--ro num-upd-rcvd?                 yang:counter32
                            │  +--ro num-upd-rcvd-unknown?         yang:counter32
                            │  +--ro num-upd-rcvd-undelegated?     yang:counter32
                            │  +--ro num-upd-rcvd-error-sent?      yang:counter32
                            │  +--ro initiation {pce-initiated}?
                            │     +--ro num-pcinitiate-sent?            yang:counter
32
                            │     +--ro num-pcinitiate-rcvd?            yang:counter
32
                            │     +--ro num-initiate-sent?              yang:counter
32
                            │     +--ro num-initiate-rcvd?              yang:counter
32
                            │     +--ro num-initiate-rcvd-error-sent?   yang:counter
32
                            +--ro path-key {path-key}?
                               +--ro num-unknown-path-key?    yang:counter32
                               +--ro num-exp-path-key?        yang:counter32
                               +--ro num-dup-path-key?        yang:counter32
                               +--ro num-path-key-no-attempt? yang:counter32
notifications:
   +---n pcep-session-up
   │  +--ro peer-addr?           -> /pcep-state/entity/peers/peer/addr
   │  +--ro session-initiator?   -> /pcep-state/entity/peers/peer/sessions/sessi
on/initiator
   │  +--ro state-last-change?   yang:timestamp
   │  +--ro state?               pcep-sess-state
   +---n pcep-session-down
   │  +--ro peer-addr?           -> /pcep-state/entity/peers/peer/addr
   │  +--ro session-initiator?   pcep-initiator
   │  +--ro state-last-change?   yang:timestamp
   │  +--ro state?               pcep-sess-state
   +---n pcep-session-local-overload
   │  +--ro peer-addr?           -> /pcep-state/entity/peers/peer/addr
   │  +--ro session-initiator?   -> /pcep-state/entity/peers/peer/sessions/sessi
on/initiator
   │  +--ro overloaded?          boolean
```

```
    |  +--ro overload-time?        uint32
    +---n pcep-session-local-overload-clear
    |  +--ro peer-addr?    -> /pcep-state/entity/peers/peer/addr
    |  +--ro overloaded?   boolean
    +---n pcep-session-peer-overload
    |  +--ro peer-addr?                  -> /pcep-state/entity/peers/peer/addr
    |  +--ro session-initiator?    -> /pcep-state/entity/peers/peer/sessions/sess
ion/initiator
    |  +--ro peer-overloaded?       boolean
    |  +--ro peer-overload-time?    uint32
    +---n pcep-session-peer-overload-clear
       +--ro peer-addr?            -> /pcep-state/entity/peers/peer/addr
       +--ro peer-overloaded?   boolean
```

## 5.1.  The Entity

The PCEP yang module may contain status information for the local
PCEP entity.

The entity has an IP address (using ietf-inet-types [RFC6991]) and a
"role" leaf (the local entity PCEP role) as mandatory.

Note that, the PCEP MIB module [RFC7420] uses an entity list and a
system generated entity index as a primary index to the read only
entity table.  If the device implements the PCEP MIB, the "index"
leaf MUST contain the value of the corresponding pcePcepEntityIndex
and only one entity is assumed.

## 5.2.  The Peer Lists

The peer list contains peer(s) that the local PCEP entity knows
about.  A PCEP speaker is identified by its IP address.  If there is
a PCEP speaker in the network that uses multiple IP addresses then it
looks like multiple distinct peers to the other PCEP speakers in the
network.

Since PCEP sessions can be ephemeral, the peer list tracks a peer
even when no PCEP session currently exists to that peer.  The
statistics contained are an aggregate of the statistics for all
successive sessions to that peer.

To limit the quantity of information that is stored, an
implementation MAY choose to discard this information if and only if
no PCEP session exists to the corresponding peer.

The data model for PCEP peer presented in this document uses a flat
list of peers.  Each peer in the list is identified by its IP address
(addr-type, addr).

There is one list for static peer configuration ("/pcep/entity/
peers"), and a separate list for the operational state of all peers
(i.e.  static as well as discovered)("/pcep-state/entity/peers").
The former is used to enable remote PCE configuration at PCC (or PCE)
while the latter has the operational state of these peers as well as
the remote PCE peer which were discovered and PCC peers that have
initiated session.

## 5.3.  The Session Lists

The session list contains PCEP session that the PCEP entity (PCE or
PCC) is currently participating in.  The statistics in session are
semantically different from those in peer since the former applies to
the current session only, whereas the latter is the aggregate for all
sessions that have existed to that peer.

Although [RFC5440] forbids more than one active PCEP session between
a given pair of PCEP entities at any given time, there is a window
during session establishment where two sessions may exist for a given
pair, one representing a session initiated by the local PCEP entity
and the other representing a session initiated by the peer.  If
either of these sessions reaches active state first, then the other
is discarded.

The data model for PCEP session presented in this document uses a
flat list of sessions.  Each session in the list is identified by its
initiator.  This index allows two sessions to exist transiently for a
given peer, as discussed above.

There is only one list for the operational state of all sessions
("/pcep-state/entity/peers/peer/sessions/session").

## 5.4.  Notifications

This YANG model defines a list of notifications to inform client of
important events detected during the protocol operation.  The
notifications defined cover the PCEP MIB notifications.

## 6.  Advanced PCE Features

This document contains a specification of the base PCEP YANG module,
"ietf-pcep" which provides the basic PCEP [RFC5440] data model.

This document further handles advanced PCE features like -

o  Capability and Scope

o  Domain information (local/neighbour)

o  Path-Key

o  OF

o  GCO

o  P2MP

o  GMPLS

o  Inter-Layer

o  Stateful PCE

o  Segement Routing

o  Authentication including PCEPS (TLS)

[Editor's Note - Some of them would be added in a future revision.]

6.1.  Stateful PCE's LSP-DB

In the operational state of PCEP which supports stateful PCE mode, the list of LSP state are maintained in LSP-DB.  The key is the PLSP-ID and the PCC IP address.

The PCEP data model contains the operational state of LSPs (/pcep-state/entity/lsp-db/lsp/) with PCEP specific attributes.  The generic TE attributes of the LSP are defined in [I-D.ietf-teas-yang-te].  A reference to LSP state in TE model is maintained.

7.  Open Issues and Next Step

This section is added so that open issues can be tracked.  This section would be removed when the document is ready for publication.

7.1.  The PCE-Initiated LSP

The TE Model at [I-D.ietf-teas-yang-te] should support creationg of tunnels at the controller (PCE) and marking them as PCE-Initiated. The LSP-DB in the PCEP Yang (/pcep-state/entity/lsp-db/lsp/ initiation) also marks the LSPs which are PCE-initiated.

7.2.  PCEP over TLS (PCEPS)

A future version of this document would add TLS related configurations.

8.  PCEP YANG Module

    RFC Ed.: In this section, replace all occurrences of 'XXXX' with the
    actual RFC number and all occurrences of the revision date below with
    the date of RFC publication (and remove this note).


```
<CODE BEGINS> file "ietf-pcep@2016-10-27.yang"
module ietf-pcep {
    namespace "urn:ietf:params:xml:ns:yang:ietf-pcep";
    prefix pcep;

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

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

    import ietf-te {
        prefix "te";
    }

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

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


    organization
        "IETF PCE (Path Computation Element) Working Group";

    contact
        "WG Web:   <http://tools.ietf.org/wg/pce/>
         WG List:  <mailto:pce@ietf.org>
         WG Chair: JP Vasseur
                   <mailto:jpv@cisco.com>
         WG Chair: Julien Meuric
                   <mailto:julien.meuric@orange.com>
         WG Chair: Jonathan Hardwick
                   <mailto:Jonathan.Hardwick@metaswitch.com>
         Editor:   Dhruv Dhody
                   <mailto:dhruv.ietf@gmail.com>";
```

```
    description
        "The YANG module defines a generic configuration and
         operational model for PCEP common across all of the
         vendor implementations.";


    revision 2016-10-27 {
        description "Initial revision.";
        reference
            "RFC XXXX:  A YANG Data Model for Path Computation
                        Element Communications Protocol
                        (PCEP)";
    }

    /*
     * Identities
     */

    identity pcep {
        description "Identity for the PCEP protocol.";
    }

    /*
     * Typedefs
     */
    typedef pcep-role {
        type enumeration {
            enum unknown {
                value "0";
                description
                "An unknown role";
            }
            enum pcc {
                value "1";
                description
                "The role of a Path Computation Client";
            }
            enum pce {
                value "2";
                description
                "The role of Path Computation Element";
            }
            enum pcc-and-pce {
                value "3";
                description
                "The role of both Path Computation Client and
                 Path Computation Element";
            }
```

```
        }

        description
            "The role of a PCEP speaker.
             Takes one of the following values
             - unknown(0): the role is not known.
             - pcc(1): the role is of a Path Computation
               Client (PCC).
             - pce(2): the role is of a Path Computation
               Server (PCE).
             - pccAndPce(3): the role is of both a PCC and
               a PCE.";

    }

    typedef pcep-admin-status {
        type enumeration {
                enum admin-status-up {
                value "1";
                description
                "Admin Status is Up";
            }
            enum admin-status-down {
                value "2";
                description
                "Admin Status is Down";
            }
        }

        description
        "The Admin Status of the PCEP entity.
         Takes one of the following values
            - admin-status-up(1): Admin Status is Up.
            - admin-status-down(2): Admin Status is Down";
    }

    typedef pcep-oper-status {
        type enumeration {
            enum oper-status-up {
                value "1";
                description
                "The PCEP entity is active";
            }
            enum oper-status-down {
                value "2";
                description
                "The PCEP entity is inactive";
            }
```

```
            enum oper-status-going-up {
                value "3";
                description
                "The PCEP entity is activating";
            }
            enum oper-status-going-down {
                value "4";
                description
                "The PCEP entity is deactivating";
            }
            enum oper-status-failed {
                value "5";
                description
                "The PCEP entity has failed and will recover
                 when possible.";
            }
            enum oper-status-failed-perm {
                value "6";
                description
                "The PCEP entity has failed and will not recover
                 without operator intervention";
            }
        }
        description
        "The operational status of the PCEP entity.
         Takes one of the following values
             - oper-status-up(1): Active
             - oper-status-down(2): Inactive
             - oper-status-going-up(3): Activating
             - oper-status-going-down(4): Deactivating
             - oper-status-failed(5): Failed
             - oper-status-failed-perm(6): Failed Permanantly";
    }

    typedef pcep-initiator {
        type enumeration {
            enum local {
                value "1";
                description
                "The local PCEP entity initiated the session";
            }

            enum remote {
                value "2";
                description
                "The remote PCEP peer initiated the session";
            }
        }
```

```
        description
        "The initiator of the session, that is, whether the TCP
         connection was initiated by the local PCEP entity or
         the remote peer.
         Takes one of the following values
             - local(1): Initiated locally
             - remote(2): Initiated remotely";
    }

    typedef pcep-sess-state {
        type enumeration {
            enum tcp-pending {
                value "1";
                description
                    "The tcp-pending state of PCEP session.";
            }

            enum open-wait {
                value "2";
                description
                    "The open-wait state of PCEP session.";
            }

            enum keep-wait {
                value "3";
                description
                    "The keep-wait state of PCEP session.";
            }

            enum session-up {
                value "4";
                description
                    "The session-up state of PCEP session.";
            }
        }
        description
            "The current state of the session.
             The set of possible states excludes the idle state
             since entries do not exist in the idle state.
             Takes one of the following values
                 - tcp-pending(1): PCEP TCP Pending state
                 - open-wait(2): PCEP Open Wait state
                 - keep-wait(3): PCEP Keep Wait state
                 - session-up(4): PCEP Session Up state";
    }

    typedef domain-type {
        type enumeration {
```

```
            enum ospf-area {
                value "1";
                description
                    "The OSPF area.";
            }
            enum isis-area {
                value "2";
                description
                    "The IS-IS area.";
            }
            enum as {
                value "3";
                description
                    "The Autonomous System (AS).";
            }
        }
        description
            "The PCE Domain Type";
    }

    typedef domain-ospf-area {
        type union {
            type uint32;
            type yang:dotted-quad;
        }
        description
            "OSPF Area ID.";
    }

    typedef domain-isis-area {
        type string {
            pattern '[0-9A-Fa-f]{2}\.([0-9A-Fa-f]{4}\.){0,3}';
        }
        description
            "IS-IS Area ID.";
    }

    typedef domain-as {
        type uint32;
        description
            "Autonomous System number.";

    }

    typedef domain {
        type union {
            type domain-ospf-area;
            type domain-isis-area;
```

```
            type domain-as;
        }
        description
            "The Domain Information";
    }

    typedef operational-state {
        type enumeration {
            enum down {
                value "0";
                description
                    "not active.";
            }
            enum up {
                value "1";
                description
                    "signalled.";
            }
            enum active {
                value "2";
                description
                    "up and carrying traffic.";
            }
            enum going-down {
                value "3";
                description
                    "LSP is being torn down, resources are
                     being released.";
            }
            enum going-up {
                value "4";
                description
                    "LSP is being signalled.";
            }
        }
        description
            "The operational status of the LSP";
    }

    typedef lsp-error {
        type enumeration {
            enum no-error {
                value "0";
                description
                    "No error, LSP is fine.";
            }
            enum unknown {
                value "1";
```

```
                    description
                        "Unknown reason.";
                }
                enum limit {
                    value "2";
                    description
                        "Limit reached for PCE-controlled LSPs.";
                }
                enum pending {
                    value "3";
                    description
                        "Too many pending LSP update requests.";
                }
                enum unacceptable {
                    value "4";
                    description
                        "Unacceptable parameters.";
                }
                enum internal {
                    value "5";
                    description
                        "Internal error.";
                }
                enum admin {
                    value "6";
                    description
                        "LSP administratively brought down.";
                }
                enum preempted {
                    value "7";
                    description
                        "LSP preempted.";
                }
                enum rsvp {
                    value "8";
                    description
                        "RSVP signaling error.";
                }
            }
            description
                "The LSP Error Codes.";
        }

        typedef sync-state {
            type enumeration {
                enum pending {
                    value "0";
                    description
```

```
                            "The state synchronization
                             has not started.";
                    }
                enum ongoing {
                    value "1";
                    description
                        "The state synchronization
                         is ongoing.";
                }
                enum finished {
                    value "2";
                    description
                        "The state synchronization
                         is finished.";
                }
            }
            description
                "The LSP-DB state synchronization operational status.";
        }

        typedef pst{
            type enumeration{
                enum rsvp-te{
                    value "0";
                    description
                        "RSVP-TE signaling protocol";
                }
                enum sr{
                    value "1";
                    description
                        "Segment Routing Traffic Engineering";
                }
            }
            description
                "The Path Setup Type";
        }

        typedef assoc-type{
            type enumeration{
                enum protection{
                    value "1";
                    description
                        "Path Protection Association Type";
                }
            }
            description
                "The PCEP Association Type";
        }
```

```
    typedef objective-function{
        type enumeration{
            enum mcp{
                value "1";
                description
                    "Minimum Cost Path (MCP)";
            }
            enum mlp{
                value "2";
                description
                    "Minimum Load Path (MLP)";
            }
            enum mbp{
                value "3";
                description
                    "Maximum residual Bandwidth Path (MBP)";
            }
            enum mbc{
                value "4";
                description
                    "Minimize aggregate Bandwidth Consumption (MBC)";
            }
            enum mll{
                value "5";
                description
                    "Minimize the Load of the most loaded Link (MLL)";
            }
            enum mcc{
                value "6";
                description
                    "Minimize the Cumulative Cost of a set of paths
                    (MCC)";
            }
            enum spt{
                value "7";
                description
                    "Shortest Path Tree (SPT)";
            }
            enum mct{
                value "8";
                description
                    "Minimum Cost Tree (MCT)";
            }
            enum mplp{
                value "9";
                description
                    "Minimum Packet Loss Path (MPLP)";
            }
```

```
            enum mup{
                value "10";
                description
                    "Maximum Under-Utilized Path (MUP)";
            }
            enum mrup{
                value "11";
                description
                    "Maximum Reserved Under-Utilized Path (MRUP)";
            }
        }
        description
            "The PCEP Objective functions";
    }

    /*
     * Features
     */

    feature svec {
        description
            "Support synchronized path computation.";
    }

    feature gmpls {
        description
            "Support GMPLS.";
    }

    feature objective-function {
        description
            "Support OF as per RFC 5541.";
    }

    feature gco {
        description
            "Support GCO as per RFC 5557.";
    }

    feature path-key {
        description
            "Support path-key as per RFC 5520.";
    }

    feature p2mp {
        description
            "Support P2MP as per RFC 6006.";
    }
```

```
   feature stateful {
       description
           "Support stateful PCE.";
   }

   feature stateful-sync-opt {
       description
           "Support stateful sync optimization";
   }

   feature pce-initiated {
       description
           "Support PCE-Initiated LSP.";
   }

   feature tls {
       description
           "Support PCEP over TLS.";
   }

   feature sr {
       description
           "Support Segment Routing for PCE.";
   }

   /*
    * Groupings
    */



   grouping pcep-entity-info{
       description
           "This grouping defines the attributes for PCEP entity.";
       leaf connect-timer {
           type uint32 {
               range "1..65535";
           }
           units "seconds";
           default 60;
           description
               "The time in seconds that the PCEP entity will wait
                to establish a TCP connection with a peer.  If a
                TCP connection is not established within this time
                then PCEP aborts the session setup attempt.";
           reference
               "RFC 5440: Path Computation Element (PCE)
                          Communication Protocol (PCEP)";
```

```
        }

        leaf connect-max-retry {
            type uint32;
            default 5;
            description
                "The maximum number of times the system tries to
                 establish a TCP connection to a peer before the
                 session with the peer transitions to the idle
                 state.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }

        leaf init-backoff-timer {
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            description
               "The initial back-off time in seconds for retrying
                a failed session setup attempt to a peer.
                The back-off time increases for each failed
                session setup attempt, until a maximum back-off
                time is reached.  The maximum back-off time is
                max-backoff-timer.";
        }

        leaf max-backoff-timer {
            type uint32;
            units "seconds";
            description
               "The maximum back-off time in seconds for retrying
                a failed session setup attempt to a peer.
                The back-off time increases for each failed session
                setup attempt, until this maximum value is reached.
                Session setup attempts then repeat periodically
                without any further increase in back-off time.";
        }

        leaf open-wait-timer {
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            default 60;
            description
```

```
                "The time in seconds that the PCEP entity will wait
                 to receive an Open message from a peer after the
                 TCP connection has come up.
                 If no Open message is received within this time then
                 PCEP terminates the TCP connection and deletes the
                 associated sessions.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                            Communication Protocol (PCEP)";
        }

        leaf keep-wait-timer {
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            default 60;
            description
                "The time in seconds that the PCEP entity will wait
                 to receive a Keepalive or PCErr message from a peer
                 during session initialization after receiving an
                 Open message.  If no Keepalive or PCErr message is
                 received within this time then PCEP terminates the
                 TCP connection and deletes the associated
                 sessions.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                            Communication Protocol (PCEP)";
        }

        leaf keep-alive-timer {
            type uint32 {
                range "0..255";
            }
            units "seconds";
            default 30;
            description
                "The keep alive transmission timer that this PCEP
                 entity will propose in the initial OPEN message of
                 each session it is involved in.  This is the
                 maximum time between two consecutive messages sent
                 to a peer. Zero means that the PCEP entity prefers
                 not to send Keepalives at all.
                 Note that the actual Keepalive transmission
                 intervals, in either direction of an active PCEP
                 session, are determined by negotiation between the
                 peers as specified by RFC 5440, and so may differ
                 from this configured value.";
```

```
        reference
            "RFC 5440: Path Computation Element (PCE)
                    Communication Protocol (PCEP)";
    }

    leaf dead-timer {
        type uint32 {
            range "0..255";
        }
        units "seconds";
        must "(. > ../keep-alive-timer)" {
            error-message "The dead timer must be "
                    + "larger than the keep alive timer";
            description
                "This value MUST be greater than
                 keep-alive-timer.";

        }
        default 120;
        description
            "The dead timer that this PCEP entity will propose
             in the initial OPEN message of each session it is
             involved in. This is the time after which a peer
             should declare a session down if it does not
             receive any PCEP messages. Zero suggests that the
             peer does not run a dead timer at all." ;
        reference
            "RFC 5440: Path Computation Element (PCE)
                    Communication Protocol (PCEP)";
    }

    leaf allow-negotiation{
        type boolean;
        description
            "Whether the PCEP entity will permit negotiation of
             session parameters.";
    }

    leaf max-keep-alive-timer{
        type uint32 {
            range "0..255";
        }
        units "seconds";
        description
            "In PCEP session parameter negotiation in seconds,
             the maximum value that this PCEP entity will
             accept from a peer for the interval between
             Keepalive transmissions. Zero means that the PCEP
```

```
                entity will allow no Keepalive transmission at
                all." ;
        }

        leaf max-dead-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
            description
                "In PCEP session parameter negotiation in seconds,
                 the maximum value that this PCEP entity will accept
                 from a peer for the Dead timer.  Zero means that
                 the PCEP entity will allow not running a Dead
                 timer.";
        }

        leaf min-keep-alive-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
            description
                "In PCEP session parameter negotiation in seconds,
                 the minimum value that this PCEP entity will
                 accept for the interval between Keepalive
                 transmissions. Zero means that the PCEP entity
                 insists on no Keepalive transmission at all.";
        }

        leaf min-dead-timer{
            type uint32 {
                range "0..255";
            }
            units "seconds";
            description
                "In PCEP session parameter negotiation in
                 seconds, the minimum value that this PCEP entity
                 will accept for the Dead timer.  Zero means that
                 the PCEP entity insists on not running a Dead
                 timer.";
        }

        leaf sync-timer{
            if-feature svec;
            type uint32 {
                range "0..65535";
            }
```

```
            units "seconds";
            default 60;
            description
                "The value of SyncTimer in seconds is used in the
                 case of synchronized path computation request
                 using the SVEC object. Consider the case where a
                 PCReq message is received by a PCE that contains
                 the SVEC object referring to M synchronized path
                 computation requests.  If after the expiration of
                 the SyncTimer all the M path computation requests
                 have not been, received a protocol error is
                 triggered and the PCE MUST cancel the whole set
                 of path computation requests.
                 The aim of the SyncTimer is to avoid the storage
                 of unused synchronized requests should one of
                 them get lost for some reasons (for example, a
                 misbehaving PCC).
                 Zero means that the PCEP entity does not use the
                 SyncTimer.";
            reference
                "RFC 5440: Path Computation Element (PCE)
                           Communication Protocol (PCEP)";
        }


        leaf request-timer{
            type uint32 {
                range "1..65535";
            }
            units "seconds";
            description
                "The maximum time that the PCEP entity will wait
                 for a response to a PCReq message.";
        }

        leaf max-sessions{
            type uint32;
            description
              "Maximum number of sessions involving this PCEP
               entity that can exist at any time.";
        }

        leaf max-unknown-reqs{
            type uint32;
            default 5;
            description
              "The maximum number of unrecognized requests and
               replies that any session on this PCEP entity is
```

```
            willing to accept per minute before terminating
            the session.
            A PCRep message contains an unrecognized reply
            if it contains an RP object whose request ID
            does not correspond to any in-progress request
            sent by this PCEP entity.
            A PCReq message contains an unrecognized request
            if it contains an RP object whose request ID is
            zero.";
          reference
             "RFC 5440: Path Computation Element (PCE)
                      Communication Protocol (PCEP)";
      }

      leaf max-unknown-msgs{
          type uint32;
          default 5;
          description
           "The maximum number of unknown messages that any
            session on this PCEP entity is willing to accept
            per minute before terminating the session.";
          reference
             "RFC 5440: Path Computation Element (PCE)
                      Communication Protocol (PCEP)";
      }

  }//pcep-entity-info

  grouping pce-scope{
      description
          "This grouping defines PCE path computation scope
           information which maybe relevant to PCE selection.
           This information corresponds to PCE auto-discovery
           information.";
      reference
          "RFC 5088: OSPF Protocol Extensions for Path
                    Computation Element (PCE)
                    Discovery
           RFC 5089: IS-IS Protocol Extensions for Path
                    Computation Element (PCE)
                    Discovery";
      leaf intra-area-scope{
          type boolean;
          default true;
          description
              "PCE can compute intra-area paths.";
      }
      leaf intra-area-pref{
```

```
            type uint8{
                range "0..7";
            }
            description
              "The PCE's preference for intra-area TE LSP
              computation.";
        }
        leaf inter-area-scope{
            type boolean;
            default false;
            description
                "PCE can compute inter-area paths.";
        }
        leaf inter-area-scope-default{
            type boolean;
            default false;
            description
                "PCE can act as a default PCE for inter-area
                 path computation.";
        }
        leaf inter-area-pref{
            type uint8{
                range "0..7";
            }
            description
              "The PCE's preference for inter-area TE LSP
              computation.";
        }
        leaf inter-as-scope{
            type boolean;
            default false;
            description
                "PCE can compute inter-AS paths.";
        }
        leaf inter-as-scope-default{
            type boolean;
            default false;
            description
                "PCE can act as a default PCE for inter-AS
                 path computation.";
        }
        leaf inter-as-pref{
            type uint8{
                range "0..7";
            }
            description
              "The PCE's preference for inter-AS TE LSP
              computation.";
```

```
        }
        leaf inter-layer-scope{
            type boolean;
            default false;
            description
                "PCE can compute inter-layer paths.";
        }
        leaf inter-layer-pref{
            type uint8{
                range "0..7";
            }
            description
              "The PCE's preference for inter-layer TE LSP
              computation.";
        }
    }//pce-scope

    grouping domain{
        description
            "This grouping specifies a Domain where the
             PCEP speaker has topology visibility.";
        leaf domain-type{
            type domain-type;
            description
              "The domain type.";
        }
        leaf domain{
            type domain;
            description
              "The domain Information.";
        }
    }//domain

    grouping capability{
        description
            "This grouping specifies a capability
             information of local PCEP entity. This maybe
             relevant to PCE selection as well. This
             information corresponds to PCE auto-discovery
             information.";
        reference
            "RFC 5088: OSPF Protocol Extensions for Path
                       Computation Element (PCE)
                       Discovery
             RFC 5089: IS-IS Protocol Extensions for Path
                       Computation Element (PCE)
                       Discovery";
        leaf gmpls{
```

```
        if-feature gmpls;
        type boolean;
        description
          "Path computation with GMPLS link
           constraints.";
    }
    leaf bi-dir{
        type boolean;
        description
          "Bidirectional path computation.";
    }
    leaf diverse{
        type boolean;
        description
          "Diverse path computation.";
    }
    leaf load-balance{
        type boolean;
        description
          "Load-balanced path computation.";
    }
    leaf synchronize{
        if-feature svec;
        type boolean;
        description
          "Synchronized paths computation.";
    }
    leaf objective-function{
        if-feature objective-function;
        type boolean;
        description
          "Support for multiple objective functions.";
    }
    leaf add-path-constraint{
        type boolean;
        description
          "Support for additive path constraints (max
           hop count, etc.).";
    }
    leaf prioritization{
        type boolean;
        description
          "Support for request prioritization.";
    }
    leaf multi-request{
        type boolean;
        description
          "Support for multiple requests per message.";
```

```
        }
        leaf gco{
            if-feature gco;
            type boolean;
            description
              "Support for Global Concurrent Optimization
               (GCO).";
        }
        leaf p2mp{
            if-feature p2mp;
            type boolean;
            description
              "Support for P2MP path computation.";
        }

        container stateful{
            if-feature stateful;
            description
                "If stateful PCE feature is present";
            leaf enabled{
                type boolean;
                description
                    "Enabled or Disabled";
            }
            leaf active{
                type boolean;
                description
                  "Support for active stateful PCE.";
            }
            leaf pce-initiated{
                if-feature pce-initiated;
                type boolean;
                description
                  "Support for PCE-initiated LSP.";
            }
            leaf include-db-ver{
                if-feature stateful-sync-opt;
                type boolean;
                description
                    "Support inclusion of LSP-DB-VERSION
                     in LSP object";
            }
            leaf trigger-resync{
                if-feature stateful-sync-opt;
                type boolean;
                description
                    "Support PCE triggered re-synchronization";
            }
```

```
            leaf trigger-initial-sync{
                if-feature stateful-sync-opt;
                type boolean;
                description
                    "PCE triggered initial synchronization";
            }
            leaf incremental-sync{
                if-feature stateful-sync-opt;
                type boolean;
                description
                    "Support incremental (delta) sync";
            }
        }
        container sr{
            if-feature sr;
            description
                "If segment routing is supported";
            leaf enabled{
                type boolean;
                description
                    "Enabled or Disabled";
            }

        }
    }//capability

    grouping info{
        description
            "This grouping specifies all information which
             maybe relevant to both PCC and PCE.
             This information corresponds to PCE auto-discovery
             information.";
        container domain{
            description
                "The local domain for the PCEP entity";
            list domain{
                key "domain-type domain";
                description
                    "The local domain.";
                uses domain{
                    description
                        "The local domain for the PCEP entity.";
                }
            }
        }
        container capability{
            description
                "The PCEP entity capability";
```

```
            uses capability{
                description
                    "The PCEP entity supported
                    capabilities.";
            }
        }


    }//info

    grouping pce-info{
        description
            "This grouping specifies all PCE information
             which maybe relevant to the PCE selection.
             This information corresponds to PCE auto-discovery
             information.";
        container scope{
            description
                "The path computation scope";
            uses pce-scope;
        }

        container neigh-domains{
            description
                "The list of neighbour PCE-Domain
                 toward which a PCE can compute
                 paths";
            list domain{
                key "domain-type domain";

                description
                    "The neighbour domain.";
                uses domain{
                    description
                        "The PCE neighbour domain.";
                }
            }
        }
    }//pce-info

    grouping pcep-stats{
        description
            "This grouping defines statistics for PCEP. It is used
             for both peer and current session.";
        leaf avg-rsp-time{
            type uint32;
            units "milliseconds";
            must "(/pcep-state/entity/peers/peer/role != 'pcc'" +
```

```
              " or " +
              "(/pcep-state/entity/peers/peer/role = 'pcc'" +
              " and (. = 0)))" {
             error-message
                "Invalid average response time";
             description
                "If role is pcc then this leaf is meaningless
                 and is set to zero.";
        }
        description
          "The average response time.
           If an average response time has not been
           calculated then this leaf has the value zero.";
    }

    leaf lwm-rsp-time{
        type uint32;
        units "milliseconds";
        must "(/pcep-state/entity/peers/peer/role != 'pcc'" +
              " or " +
              "(/pcep-state/entity/peers/peer/role = 'pcc'" +
              " and (. = 0)))" {
             error-message
                "Invalid smallest (low-water mark)
                 response time";
             description
                "If role is pcc then this leaf is meaningless
                 and is set to zero.";
        }
        description
         "The smallest (low-water mark) response time seen.
          If no responses have been received then this
          leaf has the value zero.";
    }

    leaf hwm-rsp-time{
        type uint32;
        units "milliseconds";
        must "(/pcep-state/entity/peers/peer/role != 'pcc'" +
              " or " +
              "(/pcep-state/entity/peers/peer/role = 'pcc'" +
              " and (. = 0)))" {
             error-message
                "Invalid greatest (high-water mark)
                 response time seen";
             description
                "If role is pcc then this field is
                 meaningless and is set to zero.";
```

```
        }
        description
         "The greatest (high-water mark) response time seen.
          If no responses have been received then this object
          has the value zero.";
    }

    leaf num-pcreq-sent{
        type yang:counter32;
        description
         "The number of PCReq messages sent.";
    }

    leaf num-pcreq-rcvd{
        type yang:counter32;
        description
          "The number of PCReq messages received.";
    }

    leaf num-pcrep-sent{
        type yang:counter32;
        description
          "The number of PCRep messages sent.";
    }

    leaf num-pcrep-rcvd{
        type yang:counter32;
        description
          "The number of PCRep messages received.";
    }

    leaf num-pcerr-sent{
        type yang:counter32;
        description
          "The number of PCErr messages sent.";
    }

    leaf num-pcerr-rcvd{
        type yang:counter32;
        description
          "The number of PCErr messages received.";
    }

    leaf num-pcntf-sent{
        type yang:counter32;
        description
          "The number of PCNtf messages sent.";
    }
```

```
        leaf num-pcntf-rcvd{
            type yang:counter32;
            description
              "The number of PCNtf messages received.";
        }

        leaf num-keepalive-sent{
            type yang:counter32;
            description
              "The number of Keepalive messages sent.";
        }

        leaf num-keepalive-rcvd{
            type yang:counter32;
            description
              "The number of Keepalive messages received.";
        }

        leaf num-unknown-rcvd{
            type yang:counter32;
            description
              "The number of unknown messages received.";
        }

        leaf num-corrupt-rcvd{
            type yang:counter32;
            description
              "The number of corrupted PCEP message received.";
        }

        leaf num-req-sent{
            type yang:counter32;
            description
              "The number of requests sent.  A request corresponds
               1:1 with an RP object in a PCReq message. This might
               be greater than num-pcreq-sent because multiple
               requests can be batched into a single PCReq
               message.";
        }

        leaf num-req-sent-pend-rep{
            type yang:counter32;
            description
              "The number of requests that have been sent for
               which a response is still pending.";
        }

        leaf num-req-sent-ero-rcvd{
```

```
        type yang:counter32;
        description
          "The number of requests that have been sent for
           which a response with an ERO object was received.
           Such responses indicate that a path was
           successfully computed by the peer.";
    }

    leaf num-req-sent-nopath-rcvd{
        type yang:counter32;
        description
          "The number of requests that have been sent for
           which a response with a NO-PATH object was
           received. Such responses indicate that the peer
           could not find a path to satisfy the
           request.";
    }

    leaf num-req-sent-cancel-rcvd{
        type yang:counter32;
        description
          "The number of requests that were cancelled with
           a PCNtf message.
           This might be different than num-pcntf-rcvd because
           not all PCNtf messages are used to cancel requests,
           and a single PCNtf message can cancel multiple
           requests.";
    }

    leaf num-req-sent-error-rcvd{
        type yang:counter32;
        description
          "The number of requests that were rejected with a
           PCErr message.
           This might be different than num-pcerr-rcvd because
           not all PCErr messages are used to reject requests,
           and a single PCErr message can reject multiple
           requests.";
    }

    leaf num-req-sent-timeout{
        type yang:counter32;
        description
          "The number of requests that have been sent to a peer
           and have been abandoned because the peer has taken too
           long to respond to them.";
    }
```

```
    leaf num-req-sent-cancel-sent{
        type yang:counter32;
        description
         "The number of requests that were sent to the peer and
          explicitly cancelled by the local PCEP entity sending
          a PCNtf.";
    }

    leaf num-req-rcvd{
        type yang:counter32;
        description
         "The number of requests received.  A request
          corresponds 1:1 with an RP object in a PCReq
          message.
          This might be greater than num-pcreq-rcvd because
          multiple requests can be batched into a single
          PCReq message.";
    }

    leaf num-req-rcvd-pend-rep{
        type yang:counter32;
        description
         "The number of requests that have been received for
          which a response is still pending.";
    }

    leaf num-req-rcvd-ero-sent{
        type yang:counter32;
        description
         "The number of requests that have been received for
          which a response with an ERO object was sent.  Such
          responses indicate that a path was successfully
          computed by the local PCEP entity.";
    }

    leaf num-req-rcvd-nopath-sent{
        type yang:counter32;
        description
         "The number of requests that have been received for
          which a response with a NO-PATH object was sent. Such
          responses indicate that the local PCEP entity could
          not find a path to satisfy the request.";
    }

    leaf num-req-rcvd-cancel-sent{
        type yang:counter32;
        description
         "The number of requests received that were cancelled
```

```
                by the local PCEP entity sending a PCNtf message.
                This might be different than num-pcntf-sent because
                not all PCNtf messages are used to cancel requests,
                and a single PCNtf message can cancel multiple
                requests.";
        }

        leaf num-req-rcvd-error-sent{
            type yang:counter32;
            description
             "The number of requests received that were cancelled
              by the local PCEP entity sending a PCErr message.
              This might be different than num-pcerr-sent because
              not all PCErr messages are used to cancel requests,
              and a single PCErr message can cancel multiple
              requests.";
        }

        leaf num-req-rcvd-cancel-rcvd{
            type yang:counter32;
            description
             "The number of requests that were received from the
              peer and explicitly cancelled by the peer sending
              a PCNtf.";
        }

        leaf num-rep-rcvd-unknown{
            type yang:counter32;
            description
              "The number of responses to unknown requests
               received. A response to an unknown request is a
               response whose RP object does not contain the
               request ID of any request that is currently
               outstanding on the session.";
        }

        leaf num-req-rcvd-unknown{
            type yang:counter32;
            description
              "The number of unknown requests that have been
               received. An unknown request is a request
               whose RP object contains a request ID of
               zero.";
        }

        container svec{
            if-feature svec;
            description
```

```
                    "If synchronized path computation is supported";
              leaf num-svec-sent{
                  type yang:counter32;
                  description
                    "The number of SVEC objects sent in PCReq messages.
                     An SVEC object represents a set of synchronized
                     requests.";
              }

              leaf num-svec-req-sent{
                  type yang:counter32;
                  description
                    "The number of requests sent that appeared in one
                     or more SVEC objects.";
              }

              leaf num-svec-rcvd{
                  type yang:counter32;
                  description
                   "The number of SVEC objects received in PCReq
                    messages. An SVEC object represents a set of
                    synchronized requests.";
              }

              leaf num-svec-req-rcvd{
                  type yang:counter32;
                  description
                    "The number of requests received that appeared
                     in one or more SVEC objects.";
              }
          }
      container stateful{
          if-feature stateful;
          description
              "Stateful PCE related statistics";
          leaf num-pcrpt-sent{
              type yang:counter32;
              description
               "The number of PCRpt messages sent.";
          }

          leaf num-pcrpt-rcvd{
              type yang:counter32;
              description
                "The number of PCRpt messages received.";
          }

          leaf num-pcupd-sent{
```

```
            type yang:counter32;
            description
             "The number of PCUpd messages sent.";
        }

        leaf num-pcupd-rcvd{
            type yang:counter32;
            description
              "The number of PCUpd messages received.";
        }

        leaf num-rpt-sent{
            type yang:counter32;
            description
              "The number of LSP Reports sent.  A LSP report
               corresponds 1:1 with an LSP object in a PCRpt
               message. This might be greater than
               num-pcrpt-sent because multiple reports can
               be batched into a single PCRpt message.";
        }

        leaf num-rpt-rcvd{
            type yang:counter32;
            description
             "The number of LSP Reports received.  A LSP report
              corresponds 1:1 with an LSP object in a PCRpt
              message.
              This might be greater than num-pcrpt-rcvd because
              multiple reports can be batched into a single
              PCRpt message.";
        }

        leaf num-rpt-rcvd-error-sent{
            type yang:counter32;
            description
              "The number of reports of LSPs received that were
               responded by the local PCEP entity by sending a
               PCErr message.";
        }

        leaf num-upd-sent{
            type yang:counter32;
            description
              "The number of LSP updates sent.  A LSP update
               corresponds 1:1 with an LSP object in a PCUpd
               message. This might be greater than
               num-pcupd-sent because multiple updates can
               be batched into a single PCUpd message.";
```

```
            }

            leaf num-upd-rcvd{
                type yang:counter32;
                description
                 "The number of LSP Updates received.  A LSP update
                  corresponds 1:1 with an LSP object in a PCUpd
                  message.
                  This might be greater than num-pcupd-rcvd because
                  multiple updates can be batched into a single
                  PCUpd message.";
            }

            leaf num-upd-rcvd-unknown{
                type yang:counter32;
                description
                  "The number of updates to unknown LSPs
                   received. An update to an unknown LSP is a
                   update whose LSP object does not contain the
                   PLSP-ID of any LSP that is currently
                   present.";
            }

            leaf num-upd-rcvd-undelegated{
                type yang:counter32;
                description
                  "The number of updates to not delegated LSPs
                   received. An update to an undelegated LSP is a
                   update whose LSP object does not contain the
                   PLSP-ID of any LSP that is currently
                   delegated to current PCEP session.";
            }

            leaf num-upd-rcvd-error-sent{
                type yang:counter32;
                description
                  "The number of updates to LSPs received that were
                   responded by the local PCEP entity by sending a
                   PCErr message.";
            }
            container initiation {
                if-feature pce-initiated;
                description
                    "PCE-Initiated related statistics";
                leaf num-pcinitiate-sent{
                    type yang:counter32;
                    description
                     "The number of PCInitiate messages sent.";
```

```
                }

                leaf num-pcinitiate-rcvd{
                    type yang:counter32;
                    description
                      "The number of PCInitiate messages received.";
                }

                leaf num-initiate-sent{
                    type yang:counter32;
                    description
                      "The number of LSP Initiation sent via PCE.
                       A LSP initiation corresponds 1:1 with an LSP
                       object in a PCInitiate message. This might be
                       greater than num-pcinitiate-sent because
                       multiple initiations can be batched into a
                       single PCInitiate message.";
                }

                leaf num-initiate-rcvd{
                    type yang:counter32;
                    description
                      "The number of LSP Initiation received from
                       PCE.  A LSP initiation corresponds 1:1 with
                       an LSP object in a PCInitiate message. This
                       might be greater than num-pcinitiate-rcvd
                       because multiple initiations can be batched
                       into a single PCInitiate message.";
                }

                leaf num-initiate-rcvd-error-sent{
                    type yang:counter32;
                    description
                      "The number of initiations of LSPs received
                       that were responded by the local PCEP entity
                       by sending a PCErr message.";
                }
            }
        }
        container path-key {
            if-feature path-key;
            description
                "If Path-Key is supported";
            leaf num-unknown-path-key{
                type yang:counter32;
                description
                 "The number of attempts to expand an unknown
                 path-key.";
```

```
            }
            leaf num-exp-path-key{
                type yang:counter32;
                description
                 "The number of attempts to expand an expired
                 path-key.";
            }
            leaf num-dup-path-key{
                type yang:counter32;
                description
                 "The number of duplicate attempts to expand same
                 path-key.";
            }
            leaf num-path-key-no-attempt{
                type yang:counter32;
                description
                 "The number of expired path-keys with no attempt to
                 expand it.";
            }
        }
    }//pcep-stats

    grouping lsp-state{
        description
            "This grouping defines the attributes for LSP in LSP-DB.
             These are the attributes specifically from the PCEP
             perspective";
        leaf plsp-id{
            type uint32{
                range "1..1048575";
            }
            description
                "A PCEP-specific identifier for the LSP.  A PCC
                 creates a unique PLSP-ID for each LSP that is
                 constant for the lifetime of a PCEP session.
                 PLSP-ID is 20 bits with 0 and 0xFFFFF are
                 reserved";
        }
        leaf pcc-id{
            type inet:ip-address;
            description
                "The local internet address of the PCC, that
                 generated the PLSP-ID.";
        }

        container lsp-ref{
            description
                "reference to ietf-te lsp state";
```

```
            leaf source {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:source";
                }
                description
                  "Tunnel sender address extracted from
                  SENDER_TEMPLATE  object";
                reference "RFC3209";
            }
            leaf destination {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:"
                        + "destination";
                }
                description
                    "Tunnel endpoint address extracted from
                    SESSION object";
                reference "RFC3209";
            }
            leaf tunnel-id {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:tunnel-id";
                }
                description
                    "Tunnel identifier used in the SESSION
                    that remains constant over the life
                    of the tunnel.";
                reference "RFC3209";
            }
            leaf lsp-id {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:lsp-id";
                }
                description
                    "Identifier used in the SENDER_TEMPLATE
                    and the FILTER_SPEC that can be changed
                    to allow a sender to share resources with
                    itself.";
                reference "RFC3209";
            }
            leaf extended-tunnel-id {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:"
                    + "extended-tunnel-id";
                }
                description
                    "Extended Tunnel ID of the LSP.";
                reference "RFC3209";
```

```
            }
            leaf type {
                type leafref {
                    path "/te:te/te:lsps-state/te:lsp/te:type";
                }
                description "LSP type P2P or P2MP";
            }
        }

        leaf admin-state{
            type boolean;
            description
                "The desired operational state";
        }
        leaf operational-state{
            type operational-state;
            description
                "The operational status of the LSP";
        }
        container delegated{
            description
                "The delegation related parameters";
            leaf enabled{
                type boolean;
                description
                    "LSP is delegated or not";
            }
            leaf pce{
                type leafref {
                    path "/pcep-state/entity/peers/peer/addr";
                }
                must "(../enabled = true())"
                {
                    error-message
                        "The LSP must be delegated";
                    description
                        "When LSP is a delegated LSP";
                }
                description
                    "The reference to the PCE peer to
                    which LSP is delegated";
            }
            leaf srp-id{
                type uint32;
                description
                    "The last SRP-ID-number associated with this
                    LSP.";
            }
```

```
        }
        container initiation {
            if-feature pce-initiated;
            description
                "The PCE initiation related parameters";
            leaf enabled{
                type boolean;
                description
                    "LSP is PCE-initiated or not";
            }
            leaf pce{
                type leafref {
                    path "/pcep-state/entity/peers/peer/addr";
                }
                must "(../enabled = true())"
                {
                    error-message
                        "The LSP must be PCE-Initiated";
                    description
                        "When the LSP must be PCE-Initiated";
                }
                description
                    "The reference to the PCE
                    that initiated this LSP";
            }
        }
        leaf symbolic-path-name{
            type string;
            description
                "The symbolic path name associated with the LSP.";
        }
        leaf last-error{
            type lsp-error;
            description
                "The last error for the LSP.";
        }
        leaf pst{
            type pst;
            default "rsvp-te";
            description
                "The Path Setup Type";

        }

    }//lsp-state

    grouping notification-instance-hdr {
        description
```

```
            "This group describes common instance specific data
             for notifications.";

        leaf peer-addr {
            type leafref {
                path "/pcep-state/entity/peers/peer/addr";
            }
            description
                "Reference to peer address";
        }

    }// notification-instance-hdr

    grouping notification-session-hdr {
        description
        "This group describes common session instance specific
         data for notifications.";

        leaf session-initiator {
            type leafref {
            path "/pcep-state/entity/peers/peer/sessions/" +
                "session/initiator";
            }
            description
                "Reference to pcep session initiator leaf";
        }
    }// notification-session-hdr

    grouping stateful-pce-parameter {
        description
        "This group describes stateful PCE specific
         parameters.";
        leaf state-timeout{
            type uint32;
            units "seconds";
            description
                "When a PCEP session is terminated, a PCC
                 waits for this time period before flushing
                 LSP state associated with that PCEP session
                 and reverting to operator-defined default
                 parameters or behaviours.";
        }
        leaf redelegation-timeout{
            type uint32;
            units "seconds";
            must "((/pcep-state/entity/role = 'pcc')" +
                " or " +
                "(/pcep-state/entity/role = 'pcc-and-pce'))" +
```

```
                " and " +
                "(/pcep/entity/capability/stateful/active"
                          + "= true())"
            {
                error-message "The PCEP entity must be PCC";
                description
                    "When PCEP entity is PCC";
            }
            description
                "When a PCEP session is terminated, a PCC
                 waits for this time period before revoking
                 LSP delegation to a PCE and attempting to
                 redelegate LSPs associated with the
                 terminated PCEP session to an alternate
                 PCE.";
        }
        leaf rpt-non-pcep-lsp{
            type boolean;
            must "((/pcep-state/entity/role = 'pcc')" +
                " or " +
                "(/pcep-state/entity/role = 'pcc-and-pce'))"
            {
                error-message "The PCEP entity must be PCC";
                description
                    "When PCEP entity is PCC";
            }
            description
                "If set, a PCC reports LSPs that are not
                controlled by any PCE (for example, LSPs
                that are statically configured at the
                PCC). ";
        }

    }

    grouping authentication {
        description "Authentication Information";
        choice auth-type-selection {
            description
                "Options for expressing authentication setting.";
            case auth-key-chain {
                leaf key-chain {
                    type key-chain:key-chain-ref;
                    description
                        "key-chain name.";
                }
            }
            case auth-key {
```

```
                leaf key {
                    type string;
                description
                    "Key string in ASCII format.";
                }
                container crypto-algorithm {
                    uses key-chain:crypto-algorithm-types;
                        description
                            "Cryptographic algorithm associated
                             with key.";
                }
            }
            case auth-tls {
                if-feature tls;
                container tls {
                    description
                        "TLS related information - TBD";
                }
            }
        }
    }

    grouping path-key {
        description "Path-key related information";
        leaf enabled{
            type boolean;
            description
                "Enabled or Disabled";
        }
        leaf discard-timer {
            type uint32;
            units "minutes";
            default 10;
            description
                "A timer to discard unwanted path-keys";
        }
        leaf reuse-time {
            type uint32;
            units "minutes";
            default 30;
            description
                "A time after which the path-keys could be reused";
        }
        leaf pce-id {
            type inet:ip-address;
            description
                "PCE Address to be used in each Path-Key Subobject
                (PKS)";
```

```
        }
    }

    grouping path-key-state {
        description "Table to allow inspection of path-keys";
        list path-keys{
            key "path-key";

            description
                "The list of path-keys generated by the PCE";

            leaf path-key {
                type uint16;
                description
                    "The identifier, or token used to represent
                     the Confidential Path Segment (CPS) within
                     the context of the PCE";
            }
            container cps {
                description
                    "The Confidential Path Segment (CPS)";
                list explicit-route-objects {
                    key "index";
                    description
                        "List of explicit route objects";
                    leaf index {
                        type uint8 {
                            range "0..255";
                        }
                        description
                            "Index of this explicit route object";
                    }
                    leaf explicit-route-usage {
                        type identityref {
                            base te-types:route-usage-type;
                        }
                        description
                            "An explicit-route hop action.";
                    }
                    uses te-types:explicit-route-subobject;
                }
            }
            leaf pcc-original {
                type leafref {
                    path "/pcep-state/entity/peers/peer/addr";
                }
                description
                    "Reference to PCC peer address of
```

```
                the original request";
            }
            leaf req-id {
                type uint32;
                description
                    "The request ID of the original PCReq.";
            }
            leaf retrieved  {
                type boolean;
                description
                    "If path-key has been retrieved yet";
            }
            leaf pcc-retrieved {
                type leafref {
                    path "/pcep-state/entity/peers/peer/addr";
                }
                must "(../retrieved = true())"
                {
                    error-message
                        "The Path-key should be retreived";
                    description
                        "When Path-Key has been retreived";
                }
                description
                    "Reference to PCC peer address which
                    retreived the path-key";
            }
            leaf creation-time {
                type yang:timestamp;
                description
                    "The timestamp value at the time this Path-Key was
                    created.";
            }
            leaf discard-time {
                type uint32;
                units "minutes";
                description
                    "A time after which this path-keys will be
                    discarded";
            }
            leaf reuse-time {
                type uint32;
                units "minutes";
                description
                    "A time after which this path-keys could be
                    reused";
            }
        }
```

```
    }

    grouping of-list {
        description "List of OF";
        list objective-function{
            key "of";

            description
                "The list of authorized OF";

            leaf of {
                type objective-function;
                description
                    "The OF authorized";
            }
        }
    }
    grouping association {
        description
            "Generic Association parameters";
        leaf type {
            type "assoc-type";
            description
                "The PCEP association type";
        }
        leaf id {
            type uint16;
            description
                "PCEP Association ID";
        }
        leaf source {
          type inet:ip-address;
          description
                "PCEP Association Source.";
        }
        leaf global-source {
          type uint32;
          description
                "PCEP Association Global
                Source.";
        }
        leaf extended-id{
            type string;
            description
                "Additional information to
                support unique identification.";
        }
    }
```

```
     grouping association-ref {
         description
             "Generic Association parameters";
         leaf id {
             type leafref {
                 path "/pcep-state/entity/lsp-db/"
                 + "association-list/id";
             }
             description
                 "PCEP Association ID";
         }
         leaf source {
             type leafref {
                 path "/pcep-state/entity/lsp-db/"
                 + "association-list/source";
             }
           description
                 "PCEP Association Source.";
         }
         leaf global-source {
             type leafref {
                 path "/pcep-state/entity/lsp-db/"
                 + "association-list/global-source";
             }
           description
                 "PCEP Association Global
                 Source.";
         }
         leaf extended-id{
             type leafref {
                 path "/pcep-state/entity/lsp-db/"
                 + "association-list/extended-id";
             }
             description
                 "Additional information to
                 support unique identification.";
         }
     }
     /*
      * Configuration data nodes
      */
     container pcep{

         presence
             "The PCEP is enabled";

             description
             "Parameters for list of configured PCEP entities
```

```
        on the device.";

    container entity {

        description
            "The configured PCEP entity on the device.";

        leaf addr {
            type inet:ip-address;
            mandatory true;
            description
                "The local Internet address of this PCEP
                entity.
                If operating as a PCE server, the PCEP
                entity listens on this address.
                If operating as a PCC, the PCEP entity
                binds outgoing TCP connections to this
                address.
                It is possible for the PCEP entity to
                operate both as a PCC and a PCE Server, in
                which case it uses this address both to
                listen for incoming TCP connections and to
                bind outgoing TCP connections.";
        }

        leaf enabled {
            type boolean;
            default true;
            description
                "The administrative status of this PCEP
                 Entity.";
        }

        leaf role {
            type pcep-role;
            mandatory true;
            description
                "The role that this entity can play.
                 Takes one of the following values.
                 - unknown(0): this PCEP Entity role is not
                   known.
                 - pcc(1): this PCEP Entity is a PCC.
                 - pce(2): this PCEP Entity is a PCE.
                 - pcc-and-pce(3): this PCEP Entity is both
                   a PCC and a PCE.";
        }

        leaf description {
```

```
            type string;
            description
                "Description of the PCEP entity configured
                 by the user";
        }

        leaf speaker-entity-id{
            if-feature stateful-sync-opt;
            type string;
            description
                "The Speaker Entity Identifier";
        }

        uses info {
            description
                "Local PCEP entity information";
        }


        container pce-info {
            must "((/pcep-state/entity/role = 'pce')" +
             " or " +
             "(/pcep-state/entity/role = 'pcc-and-pce'))"
            {
                error-message "The PCEP entity must be PCE";
                description
                    "When PCEP entity is PCE";
            }
            uses pce-info {
                description
                    "Local PCE information";
            }
            container path-key {
                if-feature path-key;
                uses path-key {
                    description
                        "Path-Key Configuration";
                }
                description
                    "Path-Key Configuration";
            }

            description
                "The Local PCE Entity PCE information";
        }

        uses authentication {
            description
```

```
                    "Local PCEP entity authentication information";
            }


            uses pcep-entity-info {
                description
                    "The configuration related to the PCEP
                     entity.";
            }



            leaf pcep-notification-max-rate {
                type uint32;
                mandatory true;
                description
                    "This variable indicates the maximum number of
                     notifications issued per second. If events occur
                     more rapidly, the implementation may simply fail
                     to emit these notifications during that period,
                     or may queue them until an appropriate time. A
                     value of 0 means no notifications are emitted
                     and all should be discarded (that is, not
                     queued).";
            }

            container stateful-parameter{
                if-feature stateful;
                must "(/pcep/entity/capability/stateful/enabled" +
                    " = true())"
                {
                    error-message
                        "The Stateful PCE must be enabled";
                    description
                        "When PCEP entity is stateful
                         enabled";
                }
                uses stateful-pce-parameter;

                description
                    "The configured stateful parameters";
            }

            container of-list{
                if-feature objective-function;
                must "((/pcep/entity/role = 'pce')" +
                    " or " +
                    "(/pcep/entity/role = 'pcc-and-pce'))"
```

```
                    {
                        error-message
                            "The PCEP entity must be PCE";
                        description
                            "The authorized OF-List at PCE";
                    }
                    uses of-list;

                    description
                        "The authorized OF-List at PCE for all peers";
                }


            container peers{
                must "((/pcep/entity/role = 'pcc')" +
                     " or " +
                     "(/pcep/entity/role = 'pcc-and-pce'))"
                {
                    error-message
                        "The PCEP entity must be PCC";
                    description
                        "When PCEP entity is PCC, as remote
                         PCE peers are configured.";
                }
                description
                    "The list of configured peers for the
                     entity (remote PCE)";
                list peer{
                    key "addr";

                    description
                        "The peer configured for the entity.
                         (remote PCE)";

                    leaf addr {
                        type inet:ip-address;
                        description
                            "The local Internet address of this
                             PCEP peer.";
                    }

                    leaf description {
                        type string;
                        description
                            "Description of the PCEP peer
                             configured by the user";
                    }
                    uses info {
```

```
                      description
                         "PCE Peer information";
                  }
                  uses pce-info {
                      description
                         "PCE Peer information";
                  }

                  leaf delegation-pref{
                      if-feature stateful;
                      type uint8{
                          range "0..7";
                      }
                      must "(/pcep/entity/capability/stateful/active"
                           + "= true())"
                      {
                          error-message
                             "The Active Stateful PCE must be
                              enabled";
                          description
                             "When PCEP entity is active stateful
                              enabled";
                      }
                      description
                         "The PCE peer delegation preference.";
                  }
                  uses authentication {
                      description
                         "PCE Peer authentication";
                  }
                  container of-list{
                      if-feature objective-function;
                      must "((/pcep/entity/role = 'pce')" +
                      " or " +
                      "(/pcep/entity/role = 'pcc-and-pce'))"
                      {
                          error-message
                             "The PCEP entity must be PCE";
                          description
                             "The authorized OF-List at PCE";
                      }
                      uses of-list;

                      description
                         "The authorized OF-List a specific peer";
                  }
              }//peer
          }//peers
```

```
        }//entity
    }//pcep

    /*
     * Operational data nodes
     */

    container pcep-state{
        config false;
        description
            "The list of operational PCEP entities on the
             device.";

        container entity{
            description
                "The operational PCEP entity on the device.";

            leaf addr {
                type inet:ip-address;
                description
                    "The local Internet address of this PCEP
                    entity.
                    If operating as a PCE server, the PCEP
                    entity listens on this address.
                    If operating as a PCC, the PCEP entity
                    binds outgoing TCP connections to this
                    address.
                    It is possible for the PCEP entity to
                    operate both as a PCC and a PCE Server, in
                    which case it uses this address both to
                    listen for incoming TCP connections and to
                    bind outgoing TCP connections.";
            }

            leaf index{
                type uint32;
                description
                    "The index of the operational PECP
                     entity";
            }


            leaf admin-status {
                type pcep-admin-status;
                description
                    "The administrative status of this PCEP Entity.
                     This is the desired operational status as
                     currently set by an operator or by default in
```

```
                        the implementation.  The value of enabled
                        represents the current status of an attempt
                        to reach this desired status.";
            }

            leaf oper-status {
                type pcep-admin-status;
                description
                    "The operational status of the PCEP entity.
                     Takes one of the following values.
                     - oper-status-up(1): the PCEP entity is
                       active.
                     - oper-status-down(2): the PCEP entity is
                       inactive.
                     - oper-status-going-up(3): the PCEP entity is
                       activating.
                     - oper-status-going-down(4): the PCEP entity is
                       deactivating.
                     - oper-status-failed(5): the PCEP entity has
                       failed and will recover when possible.
                     - oper-status-failed-perm(6): the PCEP entity
                       has failed and will not recover without
                       operator intervention.";
            }

            leaf role {
                type pcep-role;
                description
                    "The role that this entity can play.
                     Takes one of the following values.
                     - unknown(0): this PCEP entity role is
                       not known.
                     - pcc(1): this PCEP entity is a PCC.
                     - pce(2): this PCEP entity is a PCE.
                     - pcc-and-pce(3): this PCEP entity is
                       both a PCC and a PCE.";
            }

            leaf description {
                type string;
                description
                    "Description of the PCEP entity configured
                     by the user";
            }

            leaf speaker-entity-id{
                if-feature stateful-sync-opt;
                type string;
```

```
                description
                    "The Speaker Entity Identifier";
            }

            uses info {
                description
                    "Local PCEP entity information";
            }

            container pce-info {
                when "((/pcep-state/entity/role = 'pce')" +
                 " or " +
                 "(/pcep-state/entity/role = 'pcc-and-pce'))"
                {
                    description
                        "When PCEP entity is PCE";
                }
                uses pce-info {
                    description
                        "Local PCE information";
                }

                container path-key {
                    if-feature path-key;
                    uses path-key {
                        description
                            "Path-Key Configuration";
                    }
                    description
                        "Path-Key Configuration";
                }

                description
                    "The Local PCE Entity PCE information";
            }

            uses authentication {
                description
                    "Local PCEP Entity authentication information";
            }

            uses pcep-entity-info{
                description
                    "The operational information related to the
                     PCEP entity.";
            }

            container stateful-parameter{
```

```
                if-feature stateful;
                must "(/pcep/entity/capability/stateful/enabled" +
                   " = true())"
                {
                    error-message
                       "The Stateful PCE must be enabled";
                    description
                       "When PCEP entity is stateful
                        enabled";
                }
                uses stateful-pce-parameter;

                description
                   "The operational stateful parameters";
            }



            container lsp-db{
                if-feature stateful;
                description
                   "The LSP-DB";
                leaf db-ver{
                    if-feature stateful-sync-opt;
                    type uint64;
                     must "((/pcep/entity/role = 'pcc')" +
                     " or " +
                     "(/pcep/entity/role = 'pcc-and-pce'))"
                    {
                        error-message
                           "The PCEP entity must be PCC";
                        description
                           "When PCEP entity is PCC, as remote
                            PCE peers are configured.";
                    }

                    description
                       "The LSP State Database Version Number";
                }
                list association-list {
                    key "id source global-source extended-id";
                    description
                       "List of all PCEP associations";
                    uses association {
                        description
                           "The Association attributes";
                    }
                    list lsp {
```

```
                        key "plsp-id pcc-id";
                        description
                            "List of all LSP in this association";
                        leaf plsp-id {
                            type leafref {
                                path "/pcep-state/entity/lsp-db/"
                                + "lsp/plsp-id";
                            }
                            description
                                "Reference to PLSP-ID in LSP-DB";
                        }
                        leaf pcc-id {
                            type leafref {
                                path "/pcep-state/entity/lsp-db/"
                                + "lsp/pcc-id";
                            }
                            description
                                "Reference to PCC-ID in LSP-DB";
                        }
                    }
                }
                list lsp{
                    key "plsp-id pcc-id";
                    description
                        "List of all LSPs in LSP-DB";
                    uses lsp-state{
                        description
                            "The PCEP specific attributes for
                             LSP-DB.";
                    }
                    list association-list {
                        key "id source global-source extended-id";
                        description
                            "List of all PCEP associations";
                        uses association-ref {
                            description
                                "Reference to the Association
                                 attributes";
                        }
                    }

                }
            }
            container path-keys {
                if-feature path-key;
                must "((/pcep-state/entity/role = 'pce')" +
                 " or " +
                 "(/pcep-state/entity/role = 'pcc-and-pce'))"
```

```
                {
                    error-message
                        "The PCEP entity must be PCE";
                    description
                        "When PCEP entity is PCE";
                }
                uses path-key-state;
                description
                    "The path-keys generated by the PCE";
            }
            container of-list{
                if-feature objective-function;
                must "((/pcep/entity/role = 'pce')" +
                     " or " +
                     "(/pcep/entity/role = 'pcc-and-pce'))"
                {
                    error-message
                        "The PCEP entity must be PCE";
                    description
                        "The authorized OF-List at PCE";
                }
                uses of-list;

                description
                    "The authorized OF-List at PCE for all peers";
            }
            container peers{
                description
                        "The list of peers for the entity";

                list peer{
                    key "addr";

                    description
                        "The peer for the entity.";

                    leaf addr {
                        type inet:ip-address;
                        description
                            "The local Internet address of this PCEP
                             peer.";
                    }

                    leaf role {
                        type pcep-role;
                        description
                            "The role of the PCEP Peer.
                             Takes one of the following values.
```

```
                        - unknown(0): this PCEP peer role
                          is not known.
                        - pcc(1): this PCEP peer is a PCC.
                        - pce(2): this PCEP peer is a PCE.
                        - pcc-and-pce(3): this PCEP peer
                          is both a PCC and a PCE.";
                }

                uses info {
                    description
                        "PCEP peer information";
                }

                container pce-info {
                    when "((/pcep-state/entity/role = 'pcc')" +
                        " or " +
                        "(/pcep-state/entity/role " +
                        "= 'pcc-and-pce'))"
                    {
                        description
                            "When PCEP entity is PCE";
                    }
                    uses pce-info {
                        description
                            "PCE Peer information";
                    }
                description
                    "The PCE Peer information";
                }

                leaf delegation-pref{
                    if-feature stateful;
                    type uint8{
                        range "0..7";
                    }
                    must "((/pcep-state/entity/role = 'pcc')" +
                        " or " +
                        "(/pcep-state/entity/role" +
                        " = 'pcc-and-pce'))"
                    {
                        error-message
                            "The PCEP entity must be PCC";
                        description
                            "When PCEP entity is PCC";
                    }
                    must "(/pcep/entity/capability/stateful/active"
```

```
                          + "= true())"
                {
                    error-message
                        "The Active Stateful PCE must be
                         enabled";
                    description
                        "When PCEP entity is active stateful
                         enabled";
                }
                description
                    "The PCE peer delegation preference.";
            }

            uses authentication {
                description
                    "PCE Peer authentication";
            }

            container of-list{
                if-feature objective-function;
                must "((/pcep/entity/role = 'pce')" +
                " or " +
                "(/pcep/entity/role = 'pcc-and-pce'))"
                {
                    error-message
                        "The PCEP entity must be PCE";
                    description
                        "The authorized OF-List at PCE";
                }
                uses of-list;

                description
                    "The authorized OF-List of a specific
                    peer";
            }
            leaf discontinuity-time {
                type yang:timestamp;
                description
                    "The timestamp of the time when the
                     information and statistics were
                     last reset.";
            }

            leaf initiate-session {
                type boolean;
                description
                    "Indicates whether the local PCEP
                     entity initiates sessions to this peer,
```

```
                        or waits for the peer to initiate a
                        session.";
                }

                leaf session-exists{
                    type boolean;
                    description
                        "Indicates whether a session with
                        this peer currently exists.";
                }

                leaf num-sess-setup-ok{
                    type yang:counter32;
                    description
                        "The number of PCEP sessions successfully
                        successfully established with the peer,
                        including any current session.  This
                        counter is incremented each time a
                        session with this peer is successfully
                        established.";
                }

                leaf num-sess-setup-fail{
                    type yang:counter32;
                    description
                       "The number of PCEP sessions with the peer
                        that have been attempted but failed
                        before being fully established. This
                        counter is incremented each time a
                        session retry to this peer fails.";
                }

                leaf session-up-time{
                    type yang:timestamp;
                    must "(../num-sess-setup-ok != 0  or " +
                        "(../num-sess-setup-ok = 0  and " +
                        "(. = 0)))" {
                            error-message
                                "Invalid Session Up timestamp";
                            description
                                "If num-sess-setup-ok is zero,
                                 then this leaf contains zero.";
                        }
                    description
                       "The timestamp value of the last time a
                        session with this peer was successfully
                        established.";
                }
```

```
                    leaf session-fail-time{
                        type yang:timestamp;
                        must "(../num-sess-setup-fail != 0  or " +
                            "(../num-sess-setup-fail = 0  and "  +
                            "(. = 0)))" {
                                error-message
                                    "Invalid Session Fail timestamp";
                                description
                                    "If num-sess-setup-fail is zero,
                                     then this leaf contains zero.";
                            }
                        description
                          "The timestamp value of the last time a
                           session with this peer failed to be
                           established.";
                    }

                    leaf session-fail-up-time{
                        type yang:timestamp;
                        must "(../num-sess-setup-ok != 0  or "     +
                            "(../num-sess-setup-ok = 0  and "     +
                            "(. = 0)))" {
                                error-message
                                    "Invalid Session Fail from
                                     Up timestamp";
                                description
                                    "If num-sess-setup-ok is zero,
                                     then this leaf contains zero.";
                            }
                        description
                          "The timestamp value of the last time a
                           session with this peer failed from
                           active.";
                    }

                    container pcep-stats {
                        description
                            "The container for all statistics at peer
                             level.";
                        uses pcep-stats{
                            description
                                "Since PCEP sessions can be
                                 ephemeral, the peer statistics tracks
                                 a peer even when no PCEP session
                                 currently exists to that peer. The
                                 statistics contained are an aggregate
                                 of the statistics for all successive
                                 sessions to that peer.";
```

```
                    }

                    leaf num-req-sent-closed{
                        type yang:counter32;
                        description
                            "The number of requests that were
                             sent to the peer and implicitly
                             cancelled when the session they were
                             sent over was closed.";
                    }

                    leaf num-req-rcvd-closed{
                        type yang:counter32;
                        description
                            "The number of requests that were
                             received from the peer and
                             implicitly cancelled when the
                             session they were received over
                             was closed.";
                    }
                }//pcep-stats


                container sessions {
                    description
                        "This entry represents a single PCEP
                         session in which the local PCEP entity
                         participates.
                         This entry exists only if the
                         corresponding PCEP session has been
                         initialized by some event, such as
                         manual user configuration, auto-
                         discovery of a peer, or an incoming
                         TCP connection.";

                    list session {
                        key "initiator";

                        description
                            "The list of sessions, note that
                             for a time being two sessions
                             may exist for a peer";

                        leaf initiator {
                            type pcep-initiator;
                            description
                                "The initiator of the session,
```

```
                            that is, whether the TCP
                            connection was initiated by
                            the local PCEP entity or the
                            peer.
                            There is a window during
                            session initialization where
                            two sessions can exist between
                            a pair of PCEP speakers, each
                            initiated by one of the
                            speakers. One of these
                            sessions is always discarded
                            before it leaves OpenWait state.
                            However, before it is discarded,
                            two sessions to the given peer
                            appear transiently in this MIB
                            module. The sessions are
                            distinguished by who initiated
                            them, and so this field is the
                            key.";
                    }

                    leaf state-last-change {
                        type yang:timestamp;
                        description
                            "The timestamp value at the
                            time this session entered its
                            current state as denoted by
                            the state leaf.";
                    }

                    leaf state {
                        type pcep-sess-state;
                        description
                            "The current state of the
                            session.
                            The set of possible states
                            excludes the idle state since
                            entries do not exist in the
                            idle state.";
                    }

                    leaf session-creation {
                        type yang:timestamp;
                        description
                            "The timestamp value at the
                            time this session was
                            created.";
                    }
```

```
                         leaf connect-retry {
                             type yang:counter32;
                             description
                                 "The number of times that the
                                  local PCEP entity has
                                  attempted to establish a TCP
                                  connection for this session
                                  without success. The PCEP
                                  entity gives up when this
                                  reaches connect-max-retry.";
                         }

                         leaf local-id {
                             type uint32 {
                               range "0..255";
                             }
                             description
                                 "The value of the PCEP session
                                  ID used by the local PCEP
                                  entity in the Open message
                                  for this session.
                                  If state is tcp-pending then
                                  this is the session ID that
                                  will be used in the Open
                                  message. Otherwise, this is
                                  the session ID that was sent
                                  in the Open message.";
                         }

                         leaf remote-id {
                             type uint32 {
                                 range "0..255";
                             }
                             must "((../state != 'tcp-pending'" +
                                 "and " +
                                 "../state != 'open-wait' )" +
                                 "or " +
                                 "((../state = 'tcp-pending'" +
                                 " or " +
                                 "../state = 'open-wait' )" +
                                 "and (. = 0)))" {
                                   error-message
                                       "Invalid remote-id";
                                   description
                                       "If state is tcp-pending
                                        or open-wait then this
                                        leaf is not used and
                                        MUST be set to zero.";
```

```
                        }
                        description
                            "The value of the PCEP session
                             ID used by the peer in its
                             Open message for this
                             session.";
                    }

                    leaf keepalive-timer {
                        type uint32 {
                          range "0..255";
                        }
                        units "seconds";
                        must "(../state = 'session-up'" +
                            "or " +
                            "(../state != 'session-up'" +
                            "and (. = 0)))" {
                                error-message
                                    "Invalid keepalive
                                     timer";
                                description
                                    "This field is used if
                                     and only if state is
                                     session-up. Otherwise,
                                     it is not used and
                                     MUST be set to
                                     zero.";
                        }
                        description
                            "The agreed maximum interval at
                             which the local PCEP entity
                             transmits PCEP messages on this
                             PCEP session.  Zero means that
                             the local PCEP entity never
                             sends Keepalives on this
                             session.";
                    }

                    leaf peer-keepalive-timer {
                        type uint32 {
                          range "0..255";
                        }
                        units "seconds";
                        must "(../state = 'session-up'" +
                            "or "    +
                            "(../state != 'session-up'" +
                            "and " +
                            "(. = 0)))" {
```

```
                              error-message
                                  "Invalid Peer keepalive
                                   timer";
                              description
                                  "This field is used if
                                   and only if state is
                                   session-up. Otherwise,
                                   it is not used and MUST
                                   be set to zero.";
                     }
                 description
                     "The agreed maximum interval at
                      which the peer transmits PCEP
                      messages on this PCEP session.
                      Zero means that the peer never
                      sends Keepalives on this
                      session.";
             }

             leaf dead-timer {
                 type uint32 {
                   range "0..255";
                 }
                 units "seconds";
                 description
                     "The dead timer interval for
                      this PCEP session.";
             }

             leaf peer-dead-timer {
                 type uint32 {
                   range "0..255";
                 }
                 units "seconds";
                 must "((../state != 'tcp-pending'" +
                     "and " +
                     "../state != 'open-wait' )" +
                     "or " +
                     "((../state = 'tcp-pending'" +
                     " or " +
                     "../state = 'open-wait' )" +
                     "and " +
                     "(. = 0)))" {
                        error-message
                            "Invalid Peer Dead
                             timer";
                        description
                            "If state is tcp-
```

```
                                      pending or open-wait
                                      then this leaf is not
                                      used and MUST be set to
                                      zero.";
                          }
                      description
                          "The peer's dead-timer interval
                           for this PCEP session.";
                  }

                  leaf ka-hold-time-rem {
                      type uint32 {
                        range "0..255";
                      }
                      units "seconds";
                      must "((../state != 'tcp-pending'" +
                          "and " +
                          "../state != 'open-wait' ) " +
                          "or " +
                          "((../state = 'tcp-pending'" +
                          "or " +
                          "../state = 'open-wait' )" +
                          "and " +
                          "(. = 0)))" {
                            error-message
                                "Invalid Keepalive hold
                                 time remaining";
                            description
                                "If state is tcp-pending
                                 or open-wait then this
                                 field is not used and
                                 MUST be set to zero.";
                          }
                      description
                          "The keep alive hold time
                           remaining for this session.";
                  }

                  leaf overloaded {
                      type boolean;
                      description
                          "If the local PCEP entity has
                           informed the peer that it is
                           currently overloaded, then this
                           is set to true.  Otherwise, it
                           is set to false.";
                  }
```

```
                        leaf overload-time {
                            type uint32;
                            units "seconds";
                            must "((../overloaded = true()) or" +
                                "((../overloaded != true()) and" +
                                " (. = 0)))" {
                                    error-message
                                        "Invalid overload-time";
                                    description
                                        "This field is only used
                                         if overloaded is set to
                                         true. Otherwise, it is
                                         not used and MUST be set
                                         to zero.";
                                }
                            description
                                "The interval of time that is
                                 remaining until the local PCEP
                                 entity will cease to be
                                 overloaded on this session.";
                        }

                        leaf peer-overloaded {
                            type boolean;
                            description
                                "If the peer has informed the
                                 local PCEP entity that it is
                                 currently overloaded, then this
                                 is set to true. Otherwise, it
                                 is set to false.";
                        }

                        leaf peer-overload-time {
                            type uint32;
                            units "seconds";
                            must "((../peer-overloaded = true())" +
                                " or " +
                                "((../peer-overloaded != true())" +
                                " and " +
                                "(. = 0)))" {
                                    error-message
                                        "Invalid peer overload
                                         time";
                                    description
                                        "This field is only used
                                         if peer-overloaded is
                                         set to true. Otherwise,
                                         it is not used and MUST
```

```
                                be set to zero.";
                    }
                description
                    "The interval of time that is
                     remaining until the peer will
                     cease to be overloaded.  If it
                     is not known how long the peer
                     will stay in overloaded state,
                     this leaf is set to zero.";
            }
            leaf lspdb-sync {
                if-feature stateful;
                type sync-state;
                description
                    "The LSP-DB state synchronization
                    status.";
            }
            leaf recv-db-ver{
                if-feature stateful;
                if-feature stateful-sync-opt;
                type uint64;
                 must "((/pcep-state/entity/peers/" +
                    "peer/role = 'pcc')" +
                    " or " +
                    "(/pcep-state/entity/peers/" +
                    "peer/role = 'pcc-and-pce'))"
                {
                    error-message
                        "The PCEP peer must be PCC";
                    description
                        "The PCEP peer must be PCC";
                }

                description
                    "The last received LSP State
                    Database Version Number";
            }

            container of-list{
                if-feature objective-function;
                must "((/pcep/entity/role = 'pcc')" +
                " or " +
                "(/pcep/entity/role = 'pcc-and-pce'))"
                {
                    error-message
                        "The PCEP entity must be PCC";
                    description
                        "The OF-list received on the
```

```
                                 session";
                              }
                              uses of-list;

                              description
                                  "Indicate the list of supported OF
                                  on this session";
                          }

                          leaf speaker-entity-id{
                              if-feature stateful-sync-opt;
                              type string;
                              description
                                  "The Speaker Entity Identifier";
                          }

                          leaf discontinuity-time {
                              type yang:timestamp;
                              description
                                  "The timestamp value of the time
                                   when the statistics were last
                                   reset.";
                          }

                          container pcep-stats {
                              description
                                  "The container for all statistics
                                   at session level.";
                              uses pcep-stats{
                                  description
                                      "The statistics contained are
                                       for the current sessions to
                                       that peer. These are lost
                                       when the session goes down.
                                       ";
                              }
                          }//pcep-stats

                      } // session
                  } // sessions
              }//peer
          }//peers
      }//entity
    }//pcep-state

    /*
     * Notifications
     */
```

```
    notification pcep-session-up {
        description
           "This notification is sent when the value of
            '/pcep/pcep-state/peers/peer/sessions/session/state'
            enters the 'session-up' state.";

        uses notification-instance-hdr;

        uses notification-session-hdr;

        leaf state-last-change {
            type yang:timestamp;
            description
               "The timestamp value at the time this session entered
                its current state as denoted by the state leaf.";
        }

        leaf state {
            type pcep-sess-state;
            description
               "The current state of the session.
                The set of possible states excludes the idle state
                since entries do not exist in the idle state.";
        }
    } //notification

    notification pcep-session-down {
        description
           "This notification is sent when the value of
            '/pcep/pcep-state/peers/peer/sessions/session/state'
            leaves the 'session-up' state.";

        uses notification-instance-hdr;

        leaf session-initiator {
            type pcep-initiator;
            description
               "The initiator of the session.";
        }

        leaf state-last-change {
            type yang:timestamp;
            description
               "The timestamp value at the time this session entered
                its current state as denoted by the state leaf.";
        }

        leaf state {
```

```
            type pcep-sess-state;
            description
                "The current state of the session.
                 The set of possible states excludes the idle state
                 since entries do not exist in the idle state.";
        }
    } //notification

    notification pcep-session-local-overload {
        description
            "This notification is sent when the local PCEP entity
            enters overload state for a peer.";

        uses notification-instance-hdr;

        uses notification-session-hdr;

        leaf overloaded {
            type boolean;
            description
                "If the local PCEP entity has informed the peer that
                it is currently overloaded, then this is set to
                true. Otherwise, it is set to false.";
        }

        leaf overload-time {
            type uint32;
            units "seconds";
            description
                "The interval of time that is remaining until the
                local PCEP entity will cease to be overloaded on
                this session.";
        }
    } //notification

    notification pcep-session-local-overload-clear {
        description
            "This notification is sent when the local PCEP entity
            leaves overload state for a peer.";

        uses notification-instance-hdr;

        leaf overloaded {
            type boolean;
            description
                "If the local PCEP entity has informed the peer
                that it is currently overloaded, then this is set
                to true.  Otherwise, it is set to false.";
```

```
        }
    } //notification

    notification pcep-session-peer-overload {
        description
            "This notification is sent when a peer enters overload
            state.";

        uses notification-instance-hdr;

        uses notification-session-hdr;

        leaf peer-overloaded {
            type boolean;
            description
                "If the peer has informed the local PCEP entity that
                it is currently overloaded, then this is set to true.
                Otherwise, it is set to false.";
        }

        leaf peer-overload-time {
            type uint32;
            units "seconds";
            description
                "The interval of time that is remaining until the
                peer will cease to be overloaded.  If it is not known
                how long the peer will stay in overloaded state, this
                leaf is set to zero.";
        }
    } //notification

    notification pcep-session-peer-overload-clear {
        description
            "This notification is sent when a peer leaves overload
            state.";

        uses notification-instance-hdr;

        leaf peer-overloaded {
            type boolean;
            description
                "If the peer has informed the local PCEP entity that
                it is currently overloaded, then this is set to true.
                Otherwise, it is set to false.";
        }
    } //notification
}//module
```

<CODE ENDS>

9.  Security Considerations

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

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

    TBD: List specific Subtrees and data nodes and their sensitivity/
    vulnerability.

10.  Manageability Considerations

10.1.  Control of Function and Policy

10.2.  Information and Data Models

10.3.  Liveness Detection and Monitoring

10.4.  Verify Correct Operations

10.5.  Requirements On Other Protocols

10.6.  Impact On Network Operations

11.  IANA Considerations

    This document registers a URI in the "IETF XML Registry" [RFC3688].
    Following the format in RFC 3688, the following registration has been
    made.

    URI:  urn:ietf:params:xml:ns:yang:ietf-pcep

    Registrant Contact:  The PCE WG of the IETF.

    XML:  N/A; the requested URI is an XML namespace.

This document registers a YANG module in the "YANG Module Names"
registry [RFC6020].

      Name:        ietf-pcep
      Namespace:   urn:ietf:params:xml:ns:yang:ietf-pcep
      Prefix:      pcep
      Reference:   This I-D

12.  Acknowledgements

   The initial document is based on the PCEP MIB [RFC7420].  Further
   this document structure is based on Routing Yang Module
   [I-D.ietf-netmod-routing-cfg].  We would like to thank the authors of
   aforementioned documents.

13.  References

13.1.  Normative References

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

   [RFC3688]  Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
              DOI 10.17487/RFC3688, January 2004,
              <http://www.rfc-editor.org/info/rfc3688>.

   [RFC5440]  Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation
              Element (PCE) Communication Protocol (PCEP)", RFC 5440,
              DOI 10.17487/RFC5440, March 2009,
              <http://www.rfc-editor.org/info/rfc5440>.

   [RFC6020]  Bjorklund, M., Ed., "YANG - A Data Modeling Language for
              the Network Configuration Protocol (NETCONF)", RFC 6020,
              DOI 10.17487/RFC6020, October 2010,
              <http://www.rfc-editor.org/info/rfc6020>.

   [RFC6991]  Schoenwaelder, J., Ed., "Common YANG Data Types",
              RFC 6991, DOI 10.17487/RFC6991, July 2013,
              <http://www.rfc-editor.org/info/rfc6991>.

   [I-D.ietf-pce-stateful-pce]
              Crabbe, E., Minei, I., Medved, J., and R. Varga, "PCEP
              Extensions for Stateful PCE", draft-ietf-pce-stateful-
              pce-16 (work in progress), September 2016.

   [I-D.ietf-pce-pce-initiated-lsp]
             Crabbe, E., Minei, I., Sivabalan, S., and R. Varga, "PCEP
             Extensions for PCE-initiated LSP Setup in a Stateful PCE
             Model", draft-ietf-pce-pce-initiated-lsp-07 (work in
             progress), July 2016.

   [I-D.ietf-pce-lsp-setup-type]
             Sivabalan, S., Medved, J., Minei, I., Crabbe, E., Varga,
             R., Tantsura, J., and J. Hardwick, "Conveying path setup
             type in PCEP messages", draft-ietf-pce-lsp-setup-type-03
             (work in progress), June 2015.

   [I-D.ietf-pce-segment-routing]
             Sivabalan, S., Medved, J., Filsfils, C., Crabbe, E.,
             Raszuk, R., Lopez, V., Tantsura, J., Henderickx, W., and
             J. Hardwick, "PCEP Extensions for Segment Routing", draft-
             ietf-pce-segment-routing-08 (work in progress), October
             2016.

   [I-D.ietf-teas-yang-te]
             Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H.,
             Bryskin, I., Chen, X., Jones, R., and B. Wen, "A YANG Data
             Model for Traffic Engineering Tunnels and Interfaces",
             draft-ietf-teas-yang-te-04 (work in progress), July 2016.

   [I-D.ietf-rtgwg-yang-key-chain]
             Lindem, A., Qu, Y., Yeung, D., Chen, I., Zhang, Z., and Y.
             Yang, "Routing Key Chain YANG Data Model", draft-ietf-
             rtgwg-yang-key-chain-09 (work in progress), September
             2016.

13.2.  Informative References

   [RFC4655]  Farrel, A., Vasseur, J., and J. Ash, "A Path Computation
             Element (PCE)-Based Architecture", RFC 4655,
             DOI 10.17487/RFC4655, August 2006,
             <http://www.rfc-editor.org/info/rfc4655>.

   [RFC6241]  Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
             and A. Bierman, Ed., "Network Configuration Protocol
             (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
             <http://www.rfc-editor.org/info/rfc6241>.

   [RFC6242]  Wasserman, M., "Using the NETCONF Protocol over Secure
             Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011,
             <http://www.rfc-editor.org/info/rfc6242>.

   [RFC6536]  Bierman, A. and M. Bjorklund, "Network Configuration
              Protocol (NETCONF) Access Control Model", RFC 6536,
              DOI 10.17487/RFC6536, March 2012,
              <http://www.rfc-editor.org/info/rfc6536>.

   [RFC7420]  Koushik, A., Stephan, E., Zhao, Q., King, D., and J.
              Hardwick, "Path Computation Element Communication Protocol
              (PCEP) Management Information Base (MIB) Module",
              RFC 7420, DOI 10.17487/RFC7420, December 2014,
              <http://www.rfc-editor.org/info/rfc7420>.

   [I-D.ietf-netmod-routing-cfg]
              Lhotka, L. and A. Lindem, "A YANG Data Model for Routing
              Management", draft-ietf-netmod-routing-cfg-24 (work in
              progress), October 2016.

   [I-D.ietf-netmod-rfc6087bis]
              Bierman, A., "Guidelines for Authors and Reviewers of YANG
              Data Model Documents", draft-ietf-netmod-rfc6087bis-09
              (work in progress), October 2016.

Appendix A.  Contributor Addresses

   Rohit Pobbathi
   Huawei Technologies
   Divyashree Techno Park, Whitefield
   Bangalore, Karnataka  560066
   India

   EMail: rohit.pobbathi@huawei.com

   Vinod KumarS
   Huawei Technologies
   Divyashree Techno Park, Whitefield
   Bangalore, Karnataka  560066
   India

   EMail: vinods.kumar@huawei.com

   Zafar Ali
   Cisco Systems
   Canada

   EMail: zali@cisco.com

   Xufeng Liu
   Ericsson
   1595 Spring Hill Road, Suite 500
   Vienna, VA 22182
   USA

   EMail: xufeng.liu@ericsson.com

   Young Lee
   Huawei Technologies
   5340 Legacy Drive, Building 3
   Plano, TX 75023, USA

   Phone: (469) 277-5838
   EMail: leeyoung@huawei.com

   Udayasree Palle
   Huawei Technologies
   Divyashree Techno Park, Whitefield
   Bangalore, Karnataka  560066
   India

   EMail: udayasree.palle@huawei.com

      Xian Zhang
      Huawei Technologies
      Bantian, Longgang District
      Shenzhen   518129
      P.R.China


      EMail: zhang.xian@huawei.com


      Avantika
      Huawei Technologies
      Divyashree Techno Park, Whitefield
      Bangalore, Karnataka   560066
      India


      EMail: avantika.sushilkumar@huawei.com

Authors' Addresses

      Dhruv Dhody (editor)
      Huawei Technologies
      Divyashree Techno Park, Whitefield
      Bangalore, Karnataka   560066
      India


      EMail: dhruv.ietf@gmail.com



      Jonathan Hardwick
      Metaswitch
      100 Church Street
      Enfield   EN2 6BQ
      UK


      EMail: jonathan.hardwick@metaswitch.com



      Vishnu Pavan Beeram
      Juniper Networks
      USA


      EMail: vbeeram@juniper.net



      Jeff Tantsura
      USA


      EMail: jefftant@gmail.com

CCAMP Working Group                                          A. Sharma
Internet-Draft                                                  R. Rao
Intended status: Standards Track                          Infinera Corp
Expires: January 7, 2017                                     X. Zhang
                                                   Huawei Technologies
                                                         July 6, 2016

                        OTN Service YANG Model
                draft-sharma-ccamp-otn-service-model-00

Abstract

   This document describes the YANG data model for OTN Services.

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 7, 2017.

Table of Contents

1.  Introduction

   OTN transport networks can carry various types of client services.
   In many cases, the client service is an OTN service across connected
   domains in a multi-domain network.  These OTN services can either be
   transported or switched in the OTN network.  If an OTN service is
   switched then additional parameters need to be provided to create a
   Mux OTN service.

   This document provides YANG model for creating OTN service.  The
   model augments the TE Tunnel model, which is an abstract model to
   create TE Tunnels.


2.  Model Overview

   This section provides an overview of the OTN Service Model.

2.1.  OTN Mux Service

```
                              OTN Mux Service
                    <------------------------------->


   XXXXXXXXXX              XXXXXXXXXXXXXXXXXXXXXXXX              XXXXXXXXX
  XX        XXX           XX                     XXX           XX       XX
 XX   +---+   X        XX +---+          +---+    X      XX +---+   XX
 X   |NE1+--------+-----+NE2+-----------+NE3+--------+-X---+NE4|    X
 X   +---+   XX   ^  X  +---+          +---+   XX   ^ XX  +---+    X
  XX        XX    |    XX                     XX    |  XX       XXX
   XXXXXXXXXX     |      XXXXXXXXXXXXXXXXXXXXXXXX    |   XXXXXXXXXX
     Domain-1     |           Domain-2              |    Domain-3
                  +                                 +
          Same OTN Service attributes:     Same OTN Service attributes:
          1. Client Signal                 1. Client Signal
          2. Tributary Port Number         2. Tributary Port Number
          3. Tributary Slot Granularity    3. Tributary Slot Granularity
          4. Timeslots                     4. Timeslots
```
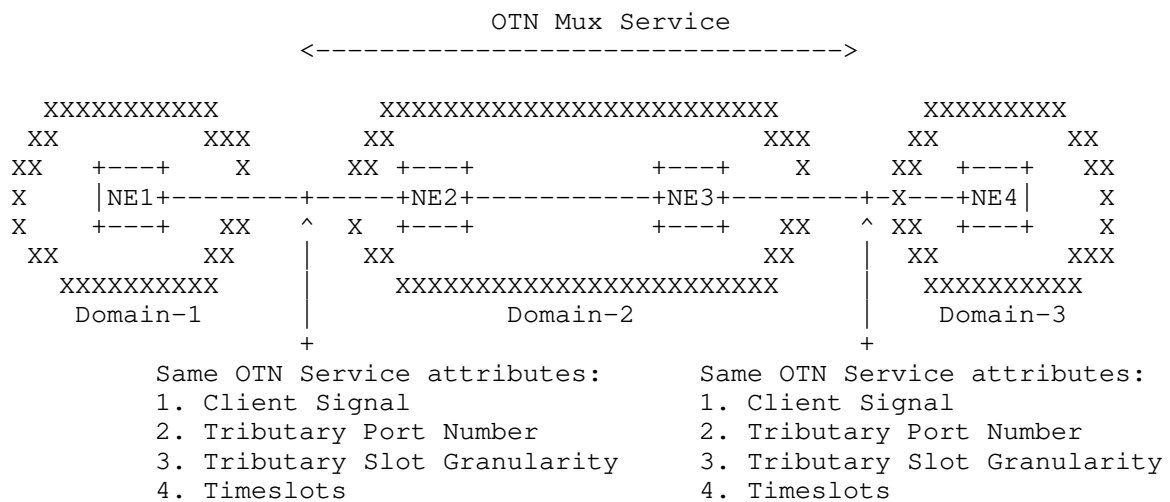

        Figure 1: OTN Mux Service in a multi-domain network topology

   Figure 1 shows a multi-domain OTN network with three domains.  In
   this example, user wants to setup an end-to-end OTN service that
   passes through Domain-2.  In order to create an OTN mux service in
   Domain-2, user will need to specify the exact details of the client
   side LO-ODU on NE2 and NE3, so that these service endpoints can be
   paired with the LO-ODU endpoints on NE1 and NE4, respectively.

Let's assume that ODU4 is the client side HO-ODU on NE2 and NE3, and
the client signal is ODU2.  User will need to specify the OTN client
signal (ODU2 in this example), the Tributary Port Number (TPN),
Tributary Slot Granularities (TSG) and timeslots to be used.  As
shown in the figure above, these service parameters must be the same
between NE1 and NE2, and NE3 and NE4.

Once the OTN Mux service is setup in Domain-2, the incoming signal
from either NE1 and/or NE4 will be switched inside Domain-2, and
delivered to NE at the other end.

2.2.  Model Tree

```
module: ietf-otn-service
augment /te:te/te:tunnels/te:tunnel/te:config:
   +--rw payload-treatment?    enumeration
   +--rw src-client-signal?    identityref
   +--rw src-tpn?              uint16
   +--rw src-tsg?              identityref
   +--rw src-timeslot-count?   uint16
   +--rw src-timeslots
   |  +--rw values*    uint8
   +--rw dst-client-signal?    identityref
   +--rw dst-tpn?              uint16
   +--rw dst-tsg?              identityref
   +--rw dst-timeslot-count?   uint16
   +--rw dst-timeslots
      +--rw values*    uint8
augment /te:te/te:tunnels/te:tunnel/te:state:
   +--ro payload-treatment?    enumeration
   +--ro src-client-signal?    identityref
   +--ro src-tpn?              uint16
   +--ro src-tsg?              identityref
   +--ro src-timeslot-count?   uint16
   +--ro src-timeslots
   |  +--ro values*    uint8
   +--ro dst-client-signal?    identityref
   +--ro dst-tpn?              uint16
   +--ro dst-tsg?              identityref
   +--ro dst-timeslot-count?   uint16
   +--ro dst-timeslots
      +--ro values*    uint8
```

2.3.  OTN Service YANG Model

  <CODE BEGINS> file "ietf-otn-service@2016-06-24.yang"

  module ietf-otn-service {

```
      yang-version 1;
      namespace "urn:ietf:params:xml:ns:yang:ietf-otn-service";
      prefix "otn-svc";

      import ietf-te { prefix "te"; }
      import ietf-transport-types { prefix "tran-types"; }
      import yang-ext { prefix ext; revision-date 2013-07-09; }

      organization
          "IETF CCAMP Working Group";

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

          Editor: Anurag Sharma
                  <mailto:AnSharma@infinera.com>

          Editor: Rajan Rao
                  <mailto:rrao@infinera.com>

          Editor: Xian Zhang
                  <mailto:zhang.xian@huawei.com>";

      description
          "This module defines a model for OTN Services.";

      revision "2016-06-24" {
          description "Initial revision";
          reference "TBD";
      }

      grouping otn-tunnel-endpoint {
          description "Parameters for OTN service.";

                  leaf payload-treatment {
              type enumeration {
                  enum switching;
                  enum transport;
              }
              default switching;
              description
                  "Treatment of the incoming payload. Payload can
                  either be switched, or transported as is.";
          }

          leaf src-client-signal {
                  type identityref {
```

```
                            base tran-types:client-signal;
                    }
                    description
                        "Client signal at the source endpoint of
                        the tunnel.";
            }

            leaf src-tpn {
                type uint16 {
                    range "0..4095";
                }
                description
                    "Tributary Port Number. Applicable in case of mux
                    services.";
                reference
                    "RFC7139: GMPLS Signaling Extensions for Control of
                    Evolving G.709 Optical Transport Networks.";
            }

            leaf src-tsg {
                type identityref {
                    base tran-types:tributary-slot-granularity;
                }
                description
                    "Tributary slot granularity. Applicable in case of mux
                    services.";
                reference
                    "G.709/Y.1331, February 2012: Interfaces for the
                    Optical Transport Network (OTN)";
            }

            leaf src-timeslot-count {
                 type uint16;
                 description
                     "Number of timeslots used at the source.";
            }

            container src-timeslots {
                description
                    "A list of tributary timeslots used by the
                     client service. Applicable in case of mux
                     services.";
                leaf-list values {
                    type uint8;
                    description
                            "Tributary timeslot value.";
                    reference
                        "G.709/Y.1331, February 2012: Interfaces for the
```

```
                             Optical Transport Network (OTN)";
            }
        }

        leaf dst-client-signal {
                type identityref {
                        base tran-types:client-signal;
                }
                description
                        "Client signal at the destination endpoint of
                        the tunnel.";
        }

        leaf dst-tpn {
            type uint16 {
                range "0..4095";
            }
            description
                "Tributary Port Number. Applicable in case of mux
                services.";
            reference
                "RFC7139: GMPLS Signaling Extensions for Control of
                Evolving G.709 Optical Transport Networks.";
        }

        leaf dst-tsg {
            type identityref {
                base tran-types:tributary-slot-granularity;
            }
            description
                "Tributary slot granularity. Applicable in case of mux
                services.";
            reference
                "G.709/Y.1331, February 2012: Interfaces for the
                Optical Transport Network (OTN)";
        }

        leaf dst-timeslot-count {
             type uint16;
             description
                 "Number of timeslots used at the destination.";
        }

        container dst-timeslots {
            description
                "A list of tributary timeslots used by the
                client service. Applicable in case of mux
                services.";
```

```
            leaf-list values {
                type uint8;
                description
                      "Tributary timeslot value.";
                reference
                    "G.709/Y.1331, February 2012: Interfaces for the
                        Optical Transport Network (OTN)";
            }
        }
    }

    /*
    Note: Comment has been given to authors of TE Tunnel model to add
    tunnel-types to the model in order to identify the technology
    type of the service.

    grouping otn-service-type {
        description
          "Identifies the OTN Service type.";
        container otn-service {
          presence "Indicates OTN Service.";
          description
            "Its presence identifies the OTN Service type.";
        }
    } // otn-service-type

    augment "/te:te/te:tunnels/te:tunnel/te:tunnel-types" {
        description
                "Introduce OTN service type for tunnel.";
        ext:augment-identifier otn-service-type-augment;
        uses otn-service-type;
    }
    */

    /*
    Note: Comment has been given to authors of TE Tunnel model to add
    list of endpoints under config to support P2MP tunnel.
    */
    augment "/te:te/te:tunnels/te:tunnel/te:config" {
        description
                "Augment with additional parameters required for OTN
                service.";
        ext:augment-identifier otn-tunnel-endpoint-config-augment;
        uses otn-tunnel-endpoint;
    }

    augment "/te:te/te:tunnels/te:tunnel/te:state" {
        description
```

```
                    "Augment with additional parameters required for OTN
                    service.";
            ext:augment-identifier otn-tunnel-endpoint-state-augment;
            uses otn-tunnel-endpoint;
        }

        /*
        Note: Comment has been given to authors of TE Tunnel model to add
        tunnel-lifecycle-event to the model. This notification is reported
        for all lifecycle changes (create, delete, and update) to the
        tunnel or lsp.
        augment "/te:tunnel-lifecycle-event" {
            description
              "OTN service event";
            uses otn-service-type;
            uses otn-tunnel-params;
            list endpoint {
              key
                "endpoint-address tp-id";
              description
                "List of Tunnel Endpoints.";
              uses te:tunnel-endpoint;
              uses otn-tunnel-params;
            }
          }
        */
    }

  <CODE ENDS>

2.4.  Transport Types YANG Model

  <CODE BEGINS> file "ietf-transport-types@2016-06-24.yang"

   module ietf-transport-types {
      yang-version 1;
      namespace "urn:ietf:params:xml:ns:yang:ietf-transport-types";
      prefix "tran-types";

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

          Editor: Anurag Sharma
                   <mailto:AnSharma@infinera.com>
```

```
        Editor: Rajan Rao
               <mailto:rrao@infinera.com>

        Editor: Xian Zhang
               <mailto:zhang.xian@huawei.com>";

    description
        "This module defines transport types.";

    revision "2016-06-24" {
        description "Initial revision";
        reference "TBD";
    }

    identity tributary-slot-granularity {
        description
               "Tributary slot granularity.";
        reference
               "G.709/Y.1331, February 2012: Interfaces for the
                        Optical Transport Network (OTN)";
    }

    identity tsg-1.25G {
        base tributary-slot-granularity;
        description
               "1.25G tributary slot granularity.";
    }

    identity tsg-2.5G {
        base tributary-slot-granularity;
        description
               "2.5G tributary slot granularity.";
    }

    identity tributary-protocol-type {
        description
               "Base identity for protocol framing used by
               tributary signals.";
    }

    identity prot-OTU1 {
        base tributary-protocol-type;
        description
               "OTU1 protocol (2.66G)";
    }

    identity prot-OTU1e {
        base tributary-protocol-type;
```

```
        description
            "OTU1e protocol (11.04G)";
    }


    identity prot-OTU2 {
        base tributary-protocol-type;
        description
            "OTU2 protocol (10.70G)";
    }

    identity prot-OTU2e {
        base tributary-protocol-type;
        description
            "OTU2e protocol (11.09G) for 10G LAN PHY";
    }

    identity prot-OTU2f {
        base tributary-protocol-type;
        description
            "OTU2f protocol (11.32G) for transporting a 10
            fiber channel.";
    }

    identity prot-OTU3 {
        base tributary-protocol-type;
        description
            "OTU3 protocol (43.01G)";
    }

    identity prot-OTU3e {
        base tributary-protocol-type;
        description
            "OTU3e protocol (44.57G) for transporting four OTU2e
            signals.";
    }

    identity prot-OTU3e2 {
        base tributary-protocol-type;
        description
            "OTU3e2 protocol (44.58G).";
    }


    identity prot-OTU4 {
        base tributary-protocol-type;
        description
            "OTU4 protocol (112G) for transporting 100GE
```

```
                signal.";
        }

        identity prot-OTUCn {
            base tributary-protocol-type;
            description
                    "OTUCn protocol (beyond 100G) for transporting
                    more than 100G signals.";
        }

        identity prot-ODU0 {
            base tributary-protocol-type;
            description
                    "ODU0 protocol (1.24G).";
        }

        identity prot-ODU1 {
            base tributary-protocol-type;
            description
                    "ODU1 protocol (2.49G).";
        }

        identity prot-ODU1e {
            base tributary-protocol-type;
            description
                    "ODU1e protocol (10.35G).";
        }

        identity prot-ODU2 {
            base tributary-protocol-type;
            description
                    "ODU2 protocol (10.03G).";
        }

        identity prot-ODU2e {
            base tributary-protocol-type;
            description
                     "ODU2e protocol (10.39G).";
        }

        identity prot-ODU3 {
            base tributary-protocol-type;
            description
                    "ODU 3 protocol (40.31G).";
        }

        identity prot-ODU3e2 {
            base tributary-protocol-type;
```

```
        description
                "ODU3e2 protocol (41.78G).";
    }

    identity prot-ODU4 {
        base tributary-protocol-type;
        description
                "ODU4 protocol (104.79G).";
    }

    identity prot-ODUFlex-cbr {
        base tributary-protocol-type;
        description
                "ODU Flex CBR protocol for transporting constant bit
                rate signal.";
    }

    identity prot-ODUFlex-gfp {
        base tributary-protocol-type;
        description
                "ODU Flex GFP protocol for transporting stream
                of packets using Generic Framing Procedure.";
    }

    identity prot-ODUCn {
        base tributary-protocol-type;
        description
                "ODUCn protocol (beyond 100G).";
    }

    identity prot-1GbE {
        base tributary-protocol-type;
        description
                "1G Ethernet protocol";
    }

    identity prot-10GbE-LAN {
        base tributary-protocol-type;
        description
                "10G Ethernet LAN protocol";
    }

    identity prot-40GbE {
        base tributary-protocol-type;
        description
                "40G Ethernet protocol";
    }
```

```
        identity prot-100GbE {
            base tributary-protocol-type;
            description
                "100G Ethernet protocol";
        }

        identity client-signal {
            description
                "Base identity from which specific client signals
                for the tunnel are derived.";
          }

        identity client-signal-1GbE {
            base client-signal;
            description
                "Client signal type of 1GbE";
          }

        identity client-signal-10GbE-LAN {
            base client-signal;
            description
                "Client signal type of 10GbE LAN";
          }

        identity client-signal-10GbE-WAN {
            base client-signal;
            description
                "Client signal type of 10GbE WAN";
          }

        identity client-signal-40GbE {
            base client-signal;
            description
                "Client signal type of 40GbE";
          }

        identity client-signal-100GbE {
            base client-signal;
            description
                "Client signal type of 100GbE";
          }

        identity client-signal-OC3_STM1 {
            base client-signal;
            description
                "Client signal type of OC3 and STM1";
          }
```

```
     identity client-signal-OC12_STM4 {
         base client-signal;
         description
               "Client signal type of OC12 and STM4";
       }

     identity client-signal-OC48_STM16 {
         base client-signal;
         description
               "Client signal type of OC48 and STM16";
       }

     identity client-signal-OC192_STM64 {
         base client-signal;
         description
               "Client signal type of OC192 and STM64";
       }

     identity client-signal-OC768_STM256 {
         base client-signal;
         description
               "Client signal type of OC768 and STM256";
       }

     identity client-signal-OTU1 {
         base client-signal;
         description
               "Client signal type of OTU1 (2.66G)";
     }

     identity client-signal-OTU2 {
         base client-signal;
         description
               "Client signal type of OTU2 (10.70G)";
     }

     identity client-signal-OTU2e {
         base client-signal;
         description
               "Client signal type of OTU2e (11.09G)";
     }

     identity client-signal-OTU2f {
         base client-signal;
         description
               "Client signal type of OTU2f (11.32G)";
     }
```

```
        identity client-signal-OTU3 {
            base client-signal;
            description
                  "Client signal type of OTU3 (43.01G)";
        }

        identity client-signal-OTU3e {
            base client-signal;
            description
                  "Client signal type of OTU3e (44.58G)";
        }

        identity client-signal-OTU4 {
            base client-signal;
            description
                  "Client signal type of OTU4 (112G)";
        }

        identity client-signal-OTUCn {
            base client-signal;
            description
                  "Client signal type of OTUCn (beyond 100G)";
        }

        identity client-signal-ODU0 {
            base client-signal;
            description
                  "Client signal type of ODU0 (1.24G)";
        }

        identity client-signal-ODU1 {
            base client-signal;
            description
                  "ODU1 protocol (2.49G)";
        }

        identity client-signal-ODU2 {
            base client-signal;
            description
                  "Client signal type of ODU2 (10.03G)";
        }

        identity client-signal-ODU2e {
            base client-signal;
            description
                  "Client signal type of ODU2e (10.39G)";
        }
```

```
        identity client-signal-ODU3 {
            base client-signal;
            description
                "Client signal type of ODU 3 (40.31G)";
        }

        identity client-signal-ODU3e2 {
            base client-signal;
            description
                "Client signal type of ODU3e2 (41.78G)";
        }

        identity client-signal-ODU4 {
            base client-signal;
            description
                "Client signal type of ODU4 (104.79G)";
        }

        identity client-signal-ODUFlex-cbr {
            base client-signal;
            description
                "Client signal type of ODU Flex CBR";
        }

        identity client-signal-ODUFlex-gfp {
            base client-signal;
            description
                "Client signal type of ODU Flex GFP";
        }

        identity client-signal-ODUCn {
            base client-signal;
            description
                "Client signal type of ODUCn (beyond 100G).";
        }

        identity client-signal-FC400 {
            base client-signal;
            description
                "Client signal type of Fibre Channel FC400.";
        }

        identity client-signal-FC800 {
            base client-signal;
            description
                "Client signal type of Fibre Channel FC800.";
        }
```

```
       identity client-signal-FICON-4G {
           base client-signal;
           description
                 "Client signal type of Fibre Connection 4G.";
       }

       identity client-signal-FICON-8G {
           base client-signal;
           description
                 "Client signal type of Fibre Connection 8G.";
       }
   }

   <CODE ENDS>
```

3.  Security Considerations

   TBD


4.  IANA Considerations

   TBD


5.  Acknowledgements


6.  Normative References

   [G.709]    ITU-T, "Interfaces for the Optical Transport Network
              (OTN), G.709/Y.1331 Recommendation", February 2012.

   [RFC6020]  Bjorklund, M., Ed., "YANG - A Data Modeling Language for
              the Network Configuration Protocol (NETCONF)", RFC 6020,
              DOI 10.17487/RFC6020, October 2010,
              <http://www.rfc-editor.org/info/rfc6020>.

   [RFC7139]  Zhang, F., Ed., Zhang, G., Belotti, S., Ceccarelli, D.,
              and K. Pithewan, "GMPLS Signaling Extensions for Control
              of Evolving G.709 Optical Transport Networks", RFC 7139,
              DOI 10.17487/RFC7139, March 2014,
              <http://www.rfc-editor.org/info/rfc7139>.

Authors' Addresses

   Anurag Sharma
   Infinera Corp
   169 Java Drive
   Sunnyvale, CA  94089
   USA

   Phone: +1-408-572-5365
   Email: AnSharma@infinera.com


   Rajan Rao
   Infinera Corp
   169 Java Drive
   Sunnyvale, CA  94089
   USA

   Phone: +1-408-543-7755
   Email: rrao@infinera.com


   Xian Zhang
   Huawei Technologies
   F3-5-B R&D Center, Huawei Industrial Base, Bantian, Longgang District
   Shenzhen, Guangdong  518129
   P.R.China

   Email: zhang.xian@huawei.com

MPLS Working Group                                              X. Xu
Internet-Draft                                               S. Bryant
Intended status: Standards Track                                Huawei
Expires: April 16, 2017                                   H. Assarpour
                                                             Broadcom
                                                              H. Shah
                                                                Ciena
                                                         L. Contreras
                                                        Telefonica I+D
                                                            D. Bernier
                                                           Bell Canada
                                                      October 13, 2016

                 Service Chaining using MPLS Source Routing
                    draft-xu-mpls-service-chaining-00

Abstract

   Source Packet Routing in Networking (SPRING) WG is developing an MPLS
   source routing mechanism.  This MPLS source routing mechanism can be
   leveraged to realize the service path layer functionality of the
   service function chaining (i.e., steering the selected traffic
   through a particular service function path) by encoding the service
   function path information as an MPLS label stack.  This document
   describes how to use the MPLS source routing mechanism as developed
   by the SPRING WG to realize the service path layer functionality of
   service function chaining.

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 April 16, 2017.

Copyright Notice

Table of Contents

1.  Introduction

When applying a particular Service Function Chain (SFC) [RFC7665] to
the traffic selected by a service classifier, the traffic need to be
steered through an ordered set of Service Functions (SF) in the
network.  This ordered set of SFs in the network indicates the
Service Function Path (SFP) associated with the above SFC.  In order
to steer the selected traffic through the required ordered list of
SFs, the service classifier needs to attach information to the packet
specifying which Service Function Forwarders (SFFs) and which SFs are
to be visited by the selected traffic.  The Source Packet Routing in
Networking (SPRING) WG is developing an MPLS source routing mechanism
which can be used to steer traffic through an ordered set of routers
(i.e., an explicit path) and instruct nodes on that path to execute

specific operations on the packet.  This MPLS source routing
mechanism thus can be leveraged to realize the service path layer
functionality of the SFC (i.e., steering traffic through a particular
SFP) by encoding the SFP information as a label stack.  This document
describes how to use the MPLS source routing mechanisms to realize
the service path layer functionality of the service function
chaining.  Note that this approach is aligned with the Transport
Derived SFF mode as described in Section 4.3.1 of [RFC7665].

2.  Terminology

   This memo makes use of the terms defined in
   [I-D.ietf-spring-segment-routing-mpls] and [RFC7665].

3.  Solution Description

```
       +------------------------------------------------ ----+
       |                 MPLS SPRING Networks                |
       |         +---------+          +---------+            |
       |         | SF1     |          | SF2     |            |
       |         +----+----+          +----+----+            |
       |              ^  |  |(3)            ^  |  |(6)        |
       |         (1)  (2)|  | V    (4)  (5)|  | V    (7)      |
  +----+-----+ --->  +----+----+ ---->  +----+----+ --->  +---+---+
  |Classifier+------+  SFF1   +-------+  SFF2   +-------+  D    |
  +----------+        +---------+        +---------+        +---+---+
       |                                                    |
       +----------------------------------------------------+
```
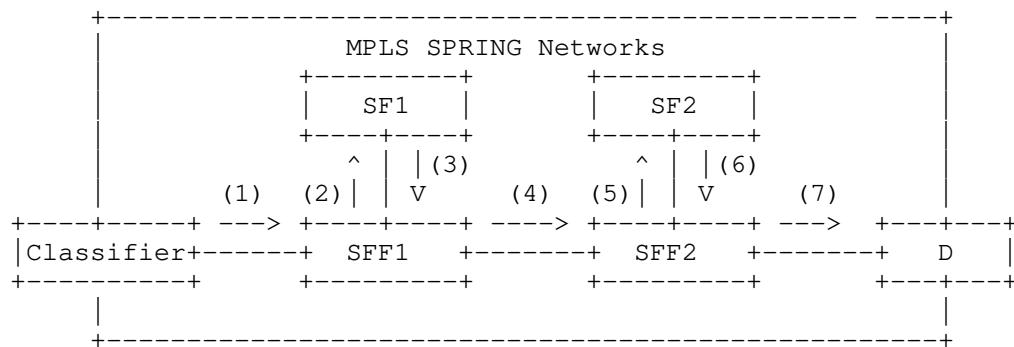          Figure 1: Service Function Chaining in MPLS-SPRING Networks

   As shown in Figure 1, SFF1 and SFF2 are two MPLS-SPRING-capable
   nodes.  They are also SFFs, each with one SF attached.  In addition,
   they have allocated and advertised Segment IDs (SID) for their
   locally attached SFs.  In the MPLS-SPRING context, SIDs are encoded
   as MPLS labels.  For example, SFF1 allocates and advertises a SID
   (i.e., SID(SF1)) for SF1 while SFF2 allocates and advertises an SID (
   i.e., SID(SF2)) for SF2.  These SIDs, which are used to indicate SFs
   are referred to as SF SIDs.  To encode the SFP information as an MPLS
   label stack, those SF SIDs as mentioned above would be interpreted as
   local MPLS labels.  More specifically, local MPLS labels are
   allocated from SFFs' (e.g., SFF1 in Figure 1) label spaces to
   identify their attached SFs (e.g., SF1 in Figure 1), whilst the SFFs
   are identified by either nodal SIDs or adjacency SIDs depending on
   how strictly the network path needs to be specified.  In addition,
   assume node SIDs for SFF1 and SFF2 are SID(SFF1) and SID(SFF2)
   respectively.  Now assume a given traffic flow destined for
   destination D is selected by the service classifier to go through a
   particular SFC (i.e., SF1-> SF2) before reaching its final

destination D.  Section 3.1 describes how to use the MPLS-based
source routing mechanism to realize the service path functionality of
the service function chaining (i.e., by encoding the SFP information
within an MPLS label stack).

3.1.  Encoding SFP Information by an MPLS Label Stack

```
          +---------------------------------------------- ----+
          |            MPLS SPRING Networks                   |
          |        +---------+        +---------+             |
          |        |  SF1    |        |  SF2    |             |
          |        +----+----+        +----+----+             |
          |+---------+   |                 |       +---------+ |
          ||SID(SFF2)|   |                 |       |Pkt to D | |
          |+---------+   |                 |       +---------+ |
          ||SID(SF2) |   |                 |                   |
          |+---------+   |                 ^                   |
          ||Pkt to D | ^ |                 |                   |
          |+---------+ | |           (5)|  | |(6)              |
          |        (2)| | |(3)          |  | V                 |
          |   (1)     | | V    (4)         |       (7)         |
    +----+-----+ ---> +----+----+ ----> +----+----+ ---> +---+---+
    |Classifier+------+  SFF1  +-------+ SFF2   +-------+   D   |
    +---------+       +---------+       +---------+       +---+---+
          |  +---------+     +---------+     +---------+       |
          |  |SID(SFF1)|     |SID(SFF2)|     |Pkt to D |       |
          |  +---------+     +---------+     +---------+       |
          |  |SID(SF1) |     |SID(SF2) |                       |
          |  +---------+     +---------+                       |
          |  |SID(SFF2)|     |Pkt to D |                       |
          |  +---------+     +---------+                       |
          |  |SID(SF2) |                                       |
          |  +---------+                                       |
          |  |Pkt to D |                                       |
          |  +---------+                                       |
          +---------------------------------------------------+
```
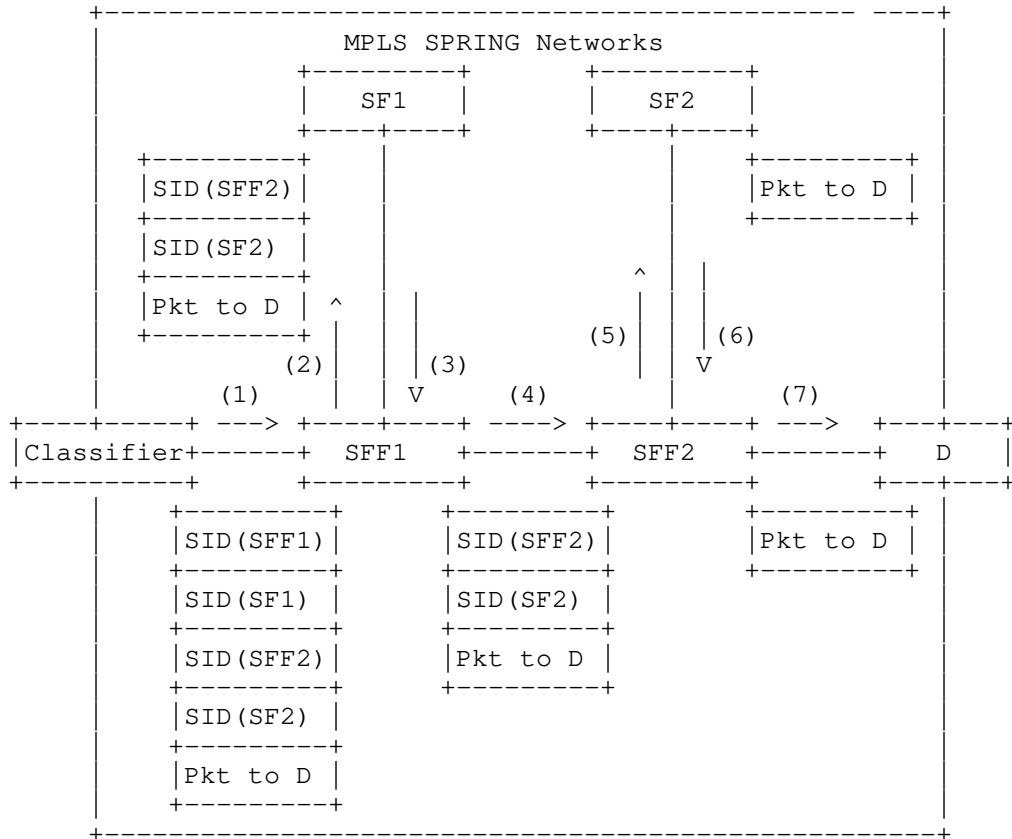                Figure 2: Packet Walk in MPLS Source Routing based SFC

   As shown in Figure 2, since the selected packet needs to travel
   through an SFC (i.e., SF1->SF2), the service classifier would attach
   a segment list of (i.e., SID(SFF1)->SID(SF1)->SID(SFF2)-> SID(SF2))
   which indicates the corresponding SFP to the packet.  This segment
   list is represented by an MPLS label stack.  To some extent, the MPLS
   label stack here could be looked as a specific implementation of the
   SFC encapsulation used for containing the SFP information [RFC7665].
   When the encapsulated packet arrives at SFF1, SFF1 would know which
   SF should be performed according to the top label (i.e., SID (SF1))
   of the received MPLS packet.  We first consider the case where SF1 is

an encapsulation aware SF, i.e., it understands how to process a packet with a pre-pended MPLS label stack.  In this case the packet would be sent to SF1 by SFF1 with the label stack SID(SFF2)-> SID(SF2).  SF1 would perform the required service function on the received MPLS packet where the payload is constrained to be an IP packet, and the SF needs to process both IPv4 and IPv6 packets (note that the SF would use the first nibble of the MPLS payload to identify the payload type).  After the MPLS packet is returned from SF1, SFF1 would send it to SFF2 according to the top label (i.e., SID (SFF2) ).

If SF1 is a legacy SF, i.e. one that is unable to process the MPLS label stack, the remaining MPLS label stack (i.e., SID(SFF2)->SID(SF2)) MUST be saved and stripped from the packet before sending the packet to SF1.  When the packet is returned from SF1, SFF1 would re-impose the MPLS label stack which had been previously stripped and then send the packet to SFF2 according to the current top label (i.e., SID (SFF2) ).  As for how to associate the corresponding MPLS label stack with the packets returned from legacy SFs, those mechanisms as described in [I-D.song-sfc-legacy-sf-mapping] could be considered.

When the encapsulated packet arrives at SFF2, SFF2 would perform the similar action to that described above.

If there is no MPLS LSP towards the next node segment (i.e., the next SFF identified by the current top label), the corresponding IP-based tunnel (e.g., MPLS-in-IP/GRE tunnel [RFC4023], MPLS-in-UDP tunnel [RFC7510] or MPLS-in-L2TPv3 tunnel [RFC4817]) would be used instead (for more details about this special usage, please refer to [I-D.xu-mpls-spring-islands-connection-over-ip]).  Since the transport (i.e., the underlay) could be IPv4, IPv6 or even MPLS networks, the above approach of encoding the SFP information by an MPLS label stack is fully transport-independent which is one of the major requirements for the SFC encapsulation [RFC7665].

4.  Acknowledgements

   The authors would like to thank Loa Andersson, Andrew G.  Malis, Adrian Farrel, Alexander Vainshtein and Joel M.  Halpern for their valuable comments and suggestions on the document

5.  IANA Considerations

   This document makes no request of IANA.

6.  Security Considerations

   It is fundamental to the SFC design that the classifier is a trusted
   resource which determines the processing that the packet will be
   subject to, including for example the firewall.  It is also
   fundamental to the SPRING design that packets are routed through the
   network using the path specified by the node imposing the SIDs.
   Where an SF is not encapsulation aware the packet may exist as an IP
   packet, however this is an intrinsic part of the SFC design which
   needs to define how a packet is protected in that environment.  Where
   a tunnel is used to link two non-MPLS domains, the tunnel design
   needs to specify how it is secured.  Thus the secutity
   vulnerabilities are addressed in the underlying technologies used by
   this design, which itself does not introduce any new security
   vulnerabilities.

7.  References

7.1.  Normative References

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

7.2.  Informative References

   [I-D.ietf-sfc-nsh]
              Quinn, P. and U. Elzur, "Network Service Header", draft-
              ietf-sfc-nsh-10 (work in progress), September 2016.

   [I-D.ietf-spring-segment-routing-mpls]
              Filsfils, C., Previdi, S., Bashandy, A., Decraene, B.,
              Litkowski, S., Horneffer, M., Shakir, R.,
              jefftant@gmail.com, j., and E. Crabbe, "Segment Routing
              with MPLS data plane", draft-ietf-spring-segment-routing-
              mpls-05 (work in progress), July 2016.

   [I-D.song-sfc-legacy-sf-mapping]
              Song, H., You, J., Yong, L., Jiang, Y., Dunbar, L.,
              Bouthors, N., and D. Dolson, "SFC Header Mapping for
              Legacy SF", draft-song-sfc-legacy-sf-mapping-08 (work in
              progress), September 2016.

   [I-D.xu-mpls-spring-islands-connection-over-ip]
             Xu, X., Raszuk, R., Chunduri, U., Contreras, L., and L.
             Jalil, "Connecting MPLS-SPRING Islands over IP Networks",
             draft-xu-mpls-spring-islands-connection-over-ip-00 (work
             in progress), October 2016.

   [RFC4023]  Worster, T., Rekhter, Y., and E. Rosen, Ed.,
             "Encapsulating MPLS in IP or Generic Routing Encapsulation
             (GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005,
             <http://www.rfc-editor.org/info/rfc4023>.

   [RFC4817]  Townsley, M., Pignataro, C., Wainner, S., Seely, T., and
             J. Young, "Encapsulation of MPLS over Layer 2 Tunneling
             Protocol Version 3", RFC 4817, DOI 10.17487/RFC4817, March
             2007, <http://www.rfc-editor.org/info/rfc4817>.

   [RFC7510]  Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black,
             "Encapsulating MPLS in UDP", RFC 7510,
             DOI 10.17487/RFC7510, April 2015,
             <http://www.rfc-editor.org/info/rfc7510>.

   [RFC7665]  Halpern, J., Ed. and C. Pignataro, Ed., "Service Function
             Chaining (SFC) Architecture", RFC 7665,
             DOI 10.17487/RFC7665, October 2015,
             <http://www.rfc-editor.org/info/rfc7665>.

Authors' Addresses

   Xiaohu Xu
   Huawei

   Email: xuxiaohu@huawei.com


   Stewart Bryant
   Huawei

   Email: stewart.bryant@gmail.com


   Hamid Assarpour
   Broadcom

   Email: hamid.assarpour@broadcom.com

Himanshu Shah
Ciena

Email: hshah@ciena.com


Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid,  28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI:   http://people.tid.es/LuisM.Contreras/


Daniel Bernier
Bell Canada

Email: daniel.bernier@bell.ca

            Service Chaining using Unified Source Routing Instructions
                    draft-xu-mpls-service-chaining-03

Abstract

   Source Packet Routing in Networking (SPRING) WG is developing an MPLS
   source routing mechanism.  The MPLS source routing mechanism can be
   leveraged to realize a unified source routing instruction which works
   across both IPv4 and IPv6 underlays in addition to the MPLS underlay.
   This document describes how to leverage the unified source routing
   instruction to realize a transport-independent service function
   chaining by encoding the service function path information or service
   function chain information as an MPLS label stack.

Requirements Language

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

Status of This Memo

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

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF).  Note that other groups may also distribute

working documents as Internet-Drafts.  The list of current Internet-
Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on December 31, 2017.

Copyright Notice

Table of Contents

1.  Introduction

When applying a particular Service Function Chain (SFC) [RFC7665] to
the traffic selected by a service classifier, the traffic need to be
steered through an ordered set of Service Functions (SF) in the
network.  This ordered set of SFs in the network indicates the
Service Function Path (SFP) associated with the above SFC.  In order

to steer the selected traffic through the required ordered list of
SFs, the service classifier needs to attach information to the packet
specifying exactly which Service Function Forwarders (SFFs) and which
SFs are to be visited by traffic), the SFC, or the partially
specified SFP which is in between the former two extremes.

The Source Packet Routing in Networking (SPRING) WG is developing an
MPLS source routing mechanism which can be used to steer traffic
through an ordered set of routers (i.e., an explicit path) and
instruct nodes on that path to execute specific operations on the
packet.  By leveraging the MPLS source routing mechanism,
[I-D.xu-mpls-unified-source-routing-instruction] describes a unified
source routing instruction which works across both IPv4 and IPv6
underlays in addition to the MPLS underlay.  This document describes
how to leverage the unified source routing instruction to realize a
transport-independent service function chaining by encoding the
service function path information or service function chain
information as an MPLS label stack.

2.  Terminology

   This memo makes use of the terms defined in
   [I-D.ietf-spring-segment-routing-mpls],
   [I-D.xu-mpls-unified-source-routing-instruction] and [RFC7665].

3.  Solution Description

```
          +------------------------------------------------- ----+
          |                 MPLS SPRING Networks                 |
          |           +--------+           +---------+           |
          |           |  SF1   |           |   SF2   |           |
          |           +----+---+           +----+----+           |
          |              ^  | |(3)            ^  | |(6)           |
          |      (1)  (2)|  | V     (4)  (5)  |  | V    (7)       |
      +----+-----+ ---> +----+----+ ----> +----+----+ ---> +---+---+
      |Classifier+------+  SFF1   +-------+  SFF2   +-------+  D   |
      +----------+      +---------+       +---------+       +---+---+
          |                                                      |
          +------------------------------------------------------+
```
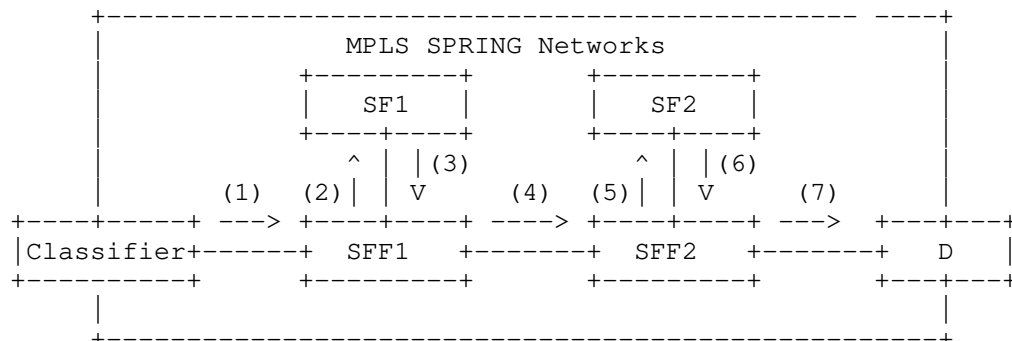       Figure 1: Service Function Chaining in MPLS-SPRING Networks

   As shown in Figure 1, SFF1 and SFF2 are two MPLS-SPRING-capable
   nodes.  They are also SFFs, each with one SF attached.  In addition,
   they have allocated and advertised MPLS labels for their locally
   attached SFs.  For example, SFF1 allocates and advertises a label
   (i.e., L(SF1)) for SF1 while SFF2 allocates and advertises a label (
   i.e., L(SF2)) for SF2.  These labels, which are used to indicate SFs
   are referred to as SF labels.  To encode the SFP information as an

MPLS label stack, local MPLS labels are allocated from SFFs' (e.g.,
SFF1 in Figure 1) label spaces to identify their locally attached SFs
(e.g., SF1 in Figure 1), whilst the SFFs are identified by either
nodal SIDs or adjacency SIDs depending on how strictly the network
path needs to be specified.  In addition, assume node SIDs for SFF1
and SFF2 are L(SFF1) and L(SFF2) respectively.  In contrast, to
encode the SFC information by an MPLS label stack, those SF labels
MUST be domain-wide unique MPLS labels.

Now assume a given traffic flow destined for destination D is
selected by the service classifier to go through a particular SFC
(i.e., SF1-> SF2) before reaching its final destination D.
Section 3.1 and 3.2 describe approaches of leveraging the MPLS- based
source routing mechanisms to realize the service function chaining by
encoding the SFP information within an MPLS label stack and by
encoding the SFC information within an MPLS label stack respectively.
Since the encoding of the partially specified SFP is just a simple
combination of the encoding of the SFP and the encoding of the SFC,
this document would not describe how to encode the partially
specified SFP anymore.

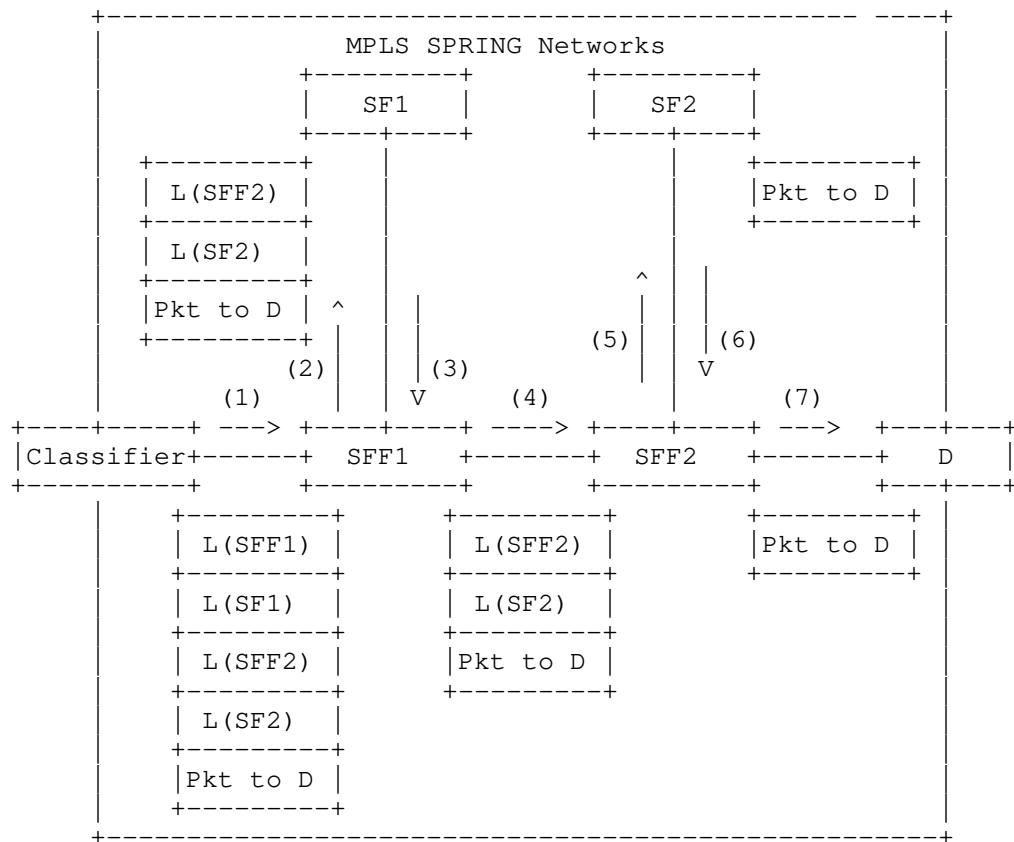3.1.  Encoding SFP Information by an MPLS Label Stack

```
             +------------------------------------------- ----+
             |            MPLS  SPRING Networks                |
             |       +--------+         +--------+             |
             |       | SF1    |         | SF2    |             |
             |       +---+----+         +----+---+             |
             |+--------+  |                |  +--------+       |
             || L(SFF2)|  |                |  |Pkt to D |      |
             |+--------+  |                |  +--------+       |
             || L(SF2) |  |                |                   |
             |+--------+  |              ^ |                   |
             ||Pkt to D| ^ |             | |                   |
             |+--------+ | |         (5) | | |(6)              |
             |          (2)|  |(3)        | | V                |
             |    (1)     | |  V     (4)  | |       (7)        |
       +----+-----+ ---> +----+----+ ----> +----+----+ ---> +---+---+
       |Classifier+------+  SFF1   +-------+  SFF2   +-------+  D    |
       +----------+      +--------+       +--------+       +---+---+
             |+--------+      +--------+        +--------+        |
             || L(SFF1)|      | L(SFF2)|        |Pkt to D |       |
             |+--------+      +--------+        +--------+        |
             || L(SF1) |      | L(SF2) |                          |
             |+--------+      +--------+                          |
             || L(SFF2)|      |Pkt to D|                          |
             |+--------+      +--------+                          |
             || L(SF2) |                                         |
             |+--------+                                         |
             ||Pkt to D|                                         |
             |+--------+                                         |
             +--------------------------------------------------+
             Figure 2: Packet Walk in MPLS underlay
```

As shown in Figure 2, since the selected packet needs to travel
through an SFC (i.e., SF1->SF2), the service classifier would attach
a segment list of (i.e., SID(SFF1)->SID(SF1)->SID(SFF2)-> SID(SF2))
which indicates the corresponding SFP to the packet.  This segment
list is represented by an MPLS label stack.  To some extent, the MPLS
label stack here could be looked as a specific implementation of the
SFC encapsulation used for containing the SFP information [RFC7665].
When the encapsulated packet arrives at SFF1, SFF1 would know which
SF should be performed according to the top label (i.e., SID (SF1))
of the received MPLS packet.  We first consider the case where SF1 is
an encapsulation aware SF, i.e., it understands how to process a
packet with a pre-pended MPLS label stack.  In this case the packet
would be sent to SF1 by SFF1 with the label stack SID(SFF2)->
SID(SF2).  SF1 would perform the required service function on the
received MPLS packet where the payload is constrained to be an IP
packet, and the SF needs to process both IPv4 and IPv6 packets (note
that the SF would use the first nibble of the MPLS payload to

identify the payload type).  After the MPLS packet is returned from
SF1, SFF1 would send it to SFF2 according to the top label (i.e., SID
(SFF2) ).

If SF1 is a legacy SF, i.e. one that is unable to process the MPLS
label stack, the remaining MPLS label stack (i.e.,
SID(SFF2)->SID(SF2)) MUST be saved and stripped from the packet
before sending the packet to SF1.  When the packet is returned from
SF1, SFF1 would re-impose the MPLS label stack which had been
previously stripped and then send the packet to SFF2 according to the
current top label (i.e., SID (SFF2) ).  As for how to associate the
corresponding MPLS label stack with the packets returned from legacy
SFs, those mechanisms as described in
[I-D.song-sfc-legacy-sf-mapping] could be considered.

When the encapsulated packet arrives at SFF2, SFF2 would perform the
similar action to that described above.

As shown in Figure 3, if there is no MPLS LSP towards the next node
segment (i.e., the next SFF identified by the current top label), the
corresponding IP-based tunnel for MPLS (e.g., MPLS-in-IP/GRE tunnel
[RFC4023], MPLS-in-UDP tunnel [RFC7510] or MPLS-in-L2TPv3 tunnel
[RFC4817]) would be used instead, according to the unified source
routing instruction as described in
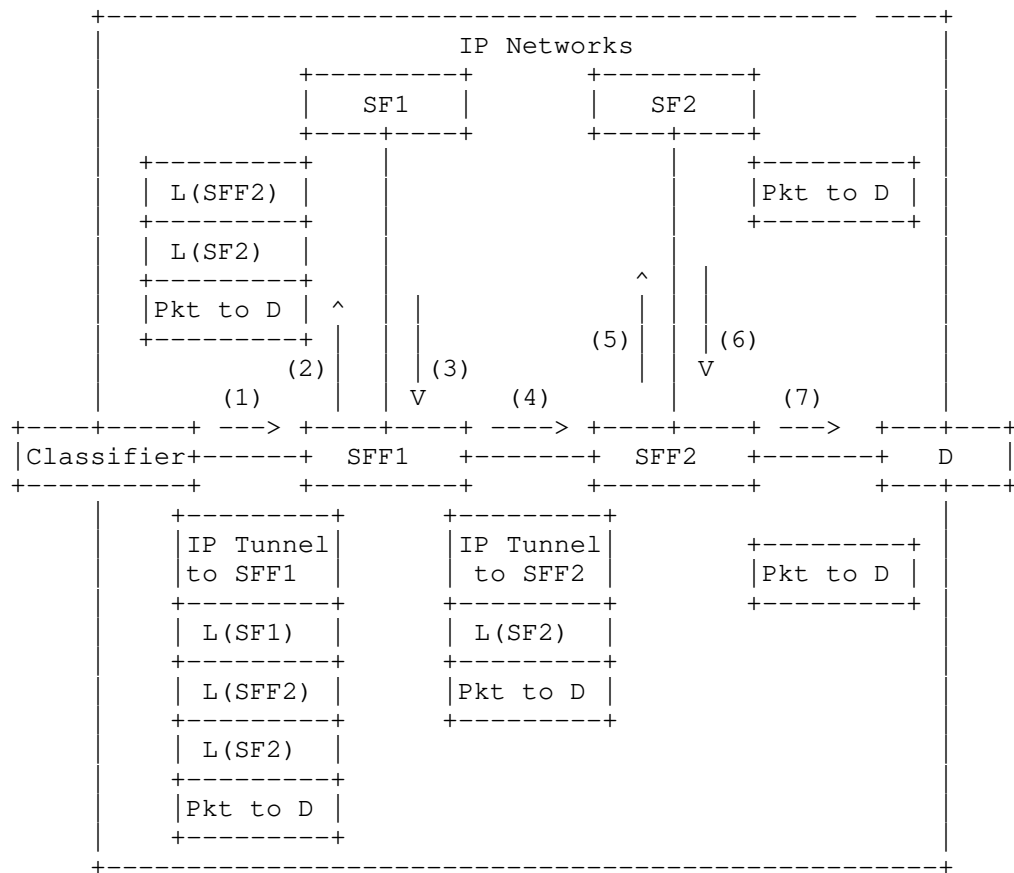[I-D.xu-mpls-unified-source-routing-instruction].

```
          +------------------------------------------- ----+
          |                  IP Networks                   |
          |        +---------+          +---------+        |
          |        |   SF1   |          |   SF2   |        |
          |        +----+----+          +----+----+        |
          | +---------+   |                |   +---------+ |
          | | L(SFF2) |   |                |   |Pkt to D | |
          | +---------+   |                |   +---------+ |
          | | L(SF2)  |   |                |             |
          | +---------+   |             ^  |             |
          | |Pkt to D | ^ |             |  |             |
          | +---------+ | |          (5)|  | |(6)        |
          |          (2)| | |(3)         |  V            |
          |      (1)    | | V    (4)     | |    (7)      |
     +----+-----+ ---> +----+----+ ----> +----+----+ ---> +---+---+
     |Classifier+------+  SFF1   +-------+  SFF2   +-------+  D   |
     +----------+      +---------+       +---------+       +---+---+
          | +---------+   +---------+                |
          | |IP Tunnel|   |IP Tunnel|     +---------+ |
          | |to SFF1  |   | to SFF2 |     |Pkt to D | |
          | +---------+   +---------+     +---------+ |
          | | L(SF1)  |   | L(SF2)  |                |
          | +---------+   +---------+                |
          | | L(SFF2) |   |Pkt to D |                |
          | +---------+   +---------+                |
          | | L(SF2)  |                              |
          | +---------+                              |
          | |Pkt to D |                              |
          | +---------+                              |
          +------------------------------------------------+
```
                  Figure 3: Packet Walk in IP underlay

   Since the transport (i.e., the underlay) could be IPv4, IPv6 or even
   MPLS networks, the above approach of encoding the SFP information by
   an MPLS label stack is fully transport-independent which is one of
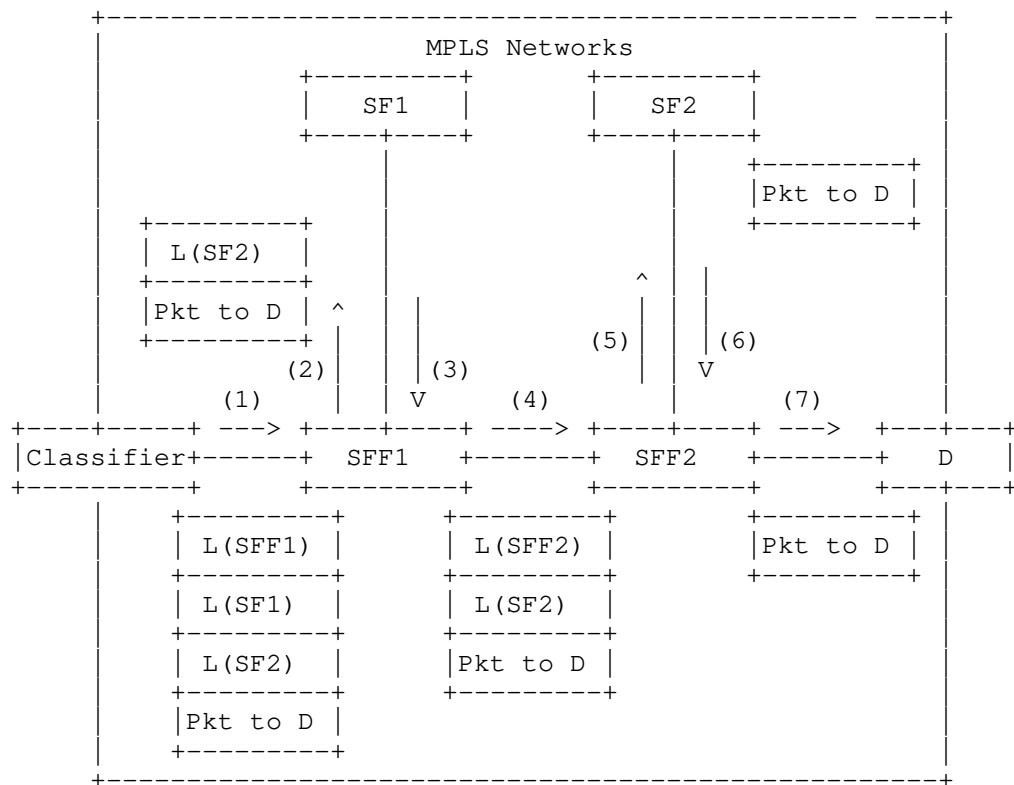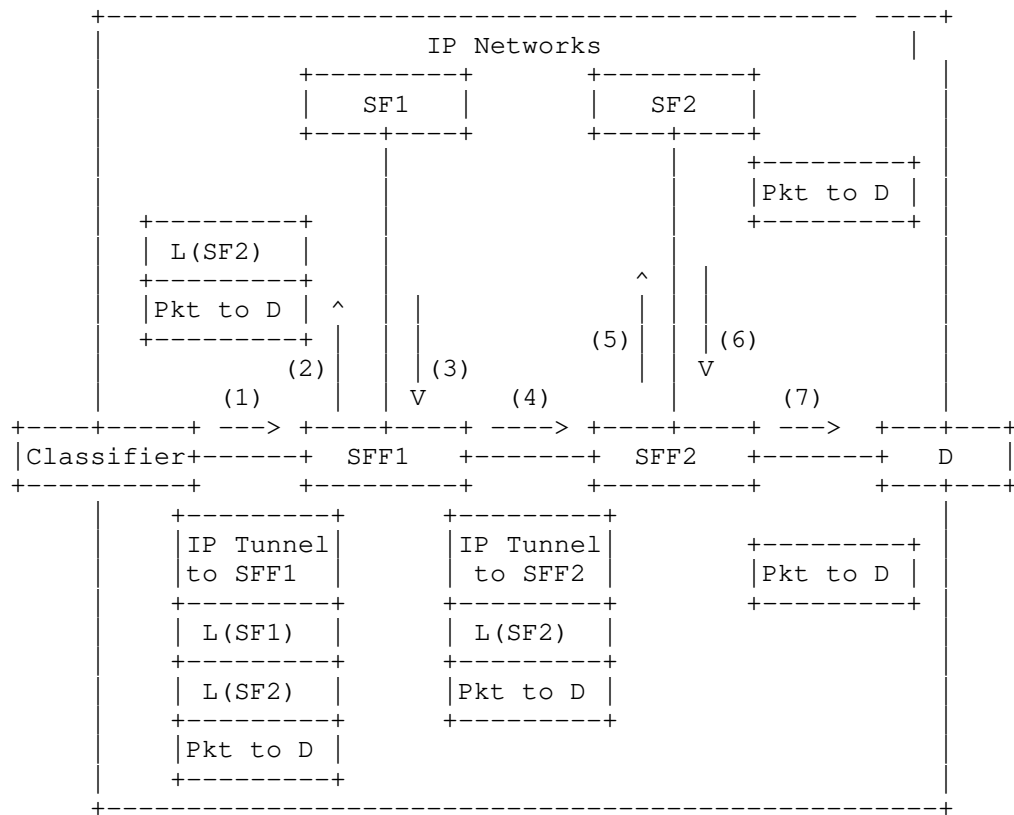   the major requirements for the SFC encapsulation [RFC7665].

3.2.  Encoding SFC Information by an MPLS Label Stack

   Since the selected packet needs to travel through an SFC (i.e.,
   SF1->SF2), the service classifier would attach an MPLS label stack
   (i.e., L(SF1)->L(SF2)) which indicates that SFC to the packet.  Since
   it's known to the service classifier that SFF1 is attached with an
   instance of SF1, the service classifier would therefore send the MPLS
   encapsulated packet through either an MPLS LSP tunnel or an IP-based
   tunnel towards SFF1 (as shown in Figure 4 and 5 respectively).  When
   the MPLS encapsulated packet arrives at SFF1, SFF1 would know which
   SF should be performed according to the current top label (i.e.,

L(SF1)).  Similarly, SFF1 would send the packet returned from SF1 to
SFF2 through either an MPLS LSP tunnel or an IP-based tunnel towards
SFF2 since it's known to SFF1 that SFF2 is attached with an instance
of SF2.  When the encapsulated packet arrives at SFF2, SFF2 would do
the similar action as what has been done by SFF1.  Since the
transport (i.e., the underlay) could be IPv4, IPv6 or even MPLS
networks, the above approach of encoding the SFC information by an
MPLS label stack is fully transport-independent which is one of the
major requirements for the SFC encapsulation [RFC7665].

```
+------------------------------------------------ ----+
|                    MPLS Networks                    |
|           +---------+      +---------+              |
|           |   SF1   |      |   SF2   |              |
|           +----+----+      +----+----+              |
|                |                |        +---------+ |
|                |                |        |Pkt to D | |
|                |                |        +---------+ |
|  +---------+   |                |                    |
|  | L(SF2)  |   |                |  ^    |            |
|  +---------+   |                | (5)   | (6)        |
|  |Pkt to D | ^ |  |             |       | V          |
|  +---------+ | |  |(3)          |                    |
|      (2)|  | |  V        (4)    |            (7)     |
|    (1)  |  |  V              |              |        |
+----+-----+ ---> +----+----+ ----> +----+----+ ---> +---+---+
|Classifier+------+  SFF1   +-------+  SFF2   +-------+   D   |
+----------+      +---------+       +---------+       +---+---+
|   +---------+        +---------+        +---------+   |
|   | L(SFF1) |        | L(SFF2) |        |Pkt to D |   |
|   +---------+        +---------+        +---------+   |
|   | L(SF1)  |        | L(SF2)  |                      |
|   +---------+        +---------+                      |
|   | L(SF2)  |        |Pkt to D |                      |
|   +---------+        +---------+                      |
|   |Pkt to D |                                        |
|   +---------+                                        |
+-----------------------------------------------------+
```
                  Figure 4: Packet Walk in MPLS underlay

```
       +----------------------------------------------- ----+
       |                     IP Networks                    |
       |          +--------+             +---------+        |
       |          |  SF1   |             |  SF2    |        |
       |          +----+---+             +----+----+        |
       |               |                      |   +---------+
       |               |                      |   |Pkt to D |
       |  +---------+   |                      |   +---------+
       |  | L(SF2)  |   |                      |            |
       |  +---------+   |                  ^   |            |
       |  |Pkt to D | ^ |                  |   |            |
       |  +---------+ | |                (5)| |(6)          |
       |          (2)| | |(3)                | V            |
       |      (1)    | | V        (4)        | |   (7)      |
  +----+-----+ ---> +----+----+ ----> +----+----+ ---> +---+---+
  |Classifier+------+ SFF1    +-------+ SFF2    +-------+  D    |
  +----------+      +---------+       +---------+       +---+---+
       |  +---------+    +---------+                        |
       |  |IP Tunnel|    |IP Tunnel|                        |
       |  |to SFF1  |    | to SFF2 |     +---------+        |
       |  +---------+    +---------+     |Pkt to D |        |
       |  | L(SF1)  |    | L(SF2)  |     +---------+        |
       |  +---------+    +---------+                        |
       |  | L(SF2)  |    |Pkt to D |                        |
       |  +---------+    +---------+                        |
       |  |Pkt to D |                                       |
       |  +---------+                                       |
       +---------------------------------------------------+
           Figure 5: Packet Walk in IP underlay
```

3.3.  How to Contain Metadata within an MPLS Packet

   Since the MPLS encapsulation has no explicit protocol identifier
   field to indicate the protocol type of the MPLS payload, how to
   indicate the presence of metadata (i.e., the NSH which is only used
   as a metadata containner) in an MPLS packet is a potential issue to
   be addressed.  One possible way to address the above issue is: SFFs
   allocate two different labels for a given SF, one indicates the
   presence of NSH while the other indicates the absence of NSH.  This
   approach has no change to the current MPLS architecture but it would
   require more than one label binding for a given SF.  Another possible
   way is to introduce a protocol identifier field within the MPLS
   packet as described in [I-D.xu-mpls-payload-protocol-identifier].

   More details about how to contain metadata within an MPLS packet
   would be considered in the future version of this draft.

4.  Acknowledgements

   The authors would like to thank Loa Andersson, Andrew G.  Malis,
   Adrian Farrel, Alexander Vainshtein and Joel M.  Halpern for their
   valuable comments and suggestions on the document.

5.  IANA Considerations

   This document makes no request of IANA.

6.  Security Considerations

   It is fundamental to the SFC design that the classifier is a trusted
   resource which determines the processing that the packet will be
   subject to, including for example the firewall.  It is also
   fundamental to the SPRING design that packets are routed through the
   network using the path specified by the node imposing the SIDs.
   Where an SF is not encapsulation aware the packet may exist as an IP
   packet, however this is an intrinsic part of the SFC design which
   needs to define how a packet is protected in that environment.  Where
   a tunnel is used to link two non-MPLS domains, the tunnel design
   needs to specify how it is secured.  Thus the secutity
   vulnerabilities are addressed in the underlying technologies used by
   this design, which itself does not introduce any new security
   vulnerabilities.

7.  References

7.1.  Normative References

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

7.2.  Informative References

   [I-D.ietf-sfc-nsh]
              Quinn, P. and U. Elzur, "Network Service Header", draft-
              ietf-sfc-nsh-12 (work in progress), February 2017.

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

   [I-D.song-sfc-legacy-sf-mapping]
              Song, H., You, J., Yong, L., Jiang, Y., Dunbar, L.,
              Bouthors, N., and D. Dolson, "SFC Header Mapping for
              Legacy SF", draft-song-sfc-legacy-sf-mapping-08 (work in
              progress), September 2016.

   [I-D.xu-mpls-payload-protocol-identifier]
              Xu, X., "MPLS Payload Protocol Identifier", draft-xu-mpls-
              payload-protocol-identifier-02 (work in progress),
              December 2016.

   [I-D.xu-mpls-unified-source-routing-instruction]
              Xu, X., Bryant, S., Raszuk, R., Chunduri, U., Contreras,
              L., Jalil, L., Assarpour, H., Velde, G., Tantsura, J., and
              S. Ma, "Unified Source Routing Instruction using MPLS
              Label Stack", draft-xu-mpls-unified-source-routing-
              instruction-02 (work in progress), June 2017.

   [RFC4023]  Worster, T., Rekhter, Y., and E. Rosen, Ed.,
              "Encapsulating MPLS in IP or Generic Routing Encapsulation
              (GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005,
              <http://www.rfc-editor.org/info/rfc4023>.

   [RFC4817]  Townsley, M., Pignataro, C., Wainner, S., Seely, T., and
              J. Young, "Encapsulation of MPLS over Layer 2 Tunneling
              Protocol Version 3", RFC 4817, DOI 10.17487/RFC4817, March
              2007, <http://www.rfc-editor.org/info/rfc4817>.

   [RFC7510]  Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black,
              "Encapsulating MPLS in UDP", RFC 7510,
              DOI 10.17487/RFC7510, April 2015,
              <http://www.rfc-editor.org/info/rfc7510>.

   [RFC7665]  Halpern, J., Ed. and C. Pignataro, Ed., "Service Function
              Chaining (SFC) Architecture", RFC 7665,
              DOI 10.17487/RFC7665, October 2015,
              <http://www.rfc-editor.org/info/rfc7665>.

Authors' Addresses

   Xiaohu Xu
   Huawei

   Email: xuxiaohu@huawei.com

Stewart Bryant
Huawei

Email: stewart.bryant@gmail.com


Hamid Assarpour
Broadcom

Email: hamid.assarpour@broadcom.com


Himanshu Shah
Ciena

Email: hshah@ciena.com


Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid,  28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI:   http://people.tid.es/LuisM.Contreras/


Daniel Bernier
Bell Canada

Email: daniel.bernier@bell.ca


Jeff Tantsura
Individual

Email: jefftant@gmail.com


Shaowen Ma
Juniper

Email: mashaowen@gmail.com

Martin Vigoureux
Nokia

Email: martin.vigoureux@nokia.com

Network Working Group                                      X. Xu, Ed.
Internet-Draft                                                  Huawei
Intended status: Standards Track                            R. Raszuk
Expires: April 15, 2017                                   Bloomberg LP
                                                          U. Chunduri

                                                         L. Contreras
                                                        Telefonica I+D
                                                             L. Jalil
                                                              Verizon
                                                     October 12, 2016

              Connecting MPLS-SPRING Islands over IP Networks
               draft-xu-mpls-spring-islands-connection-over-ip-00

Abstract

   MPLS-SPRING is an MPLS-based source routing paradigm in which a
   sender of a packet is allowed to partially or completely specify the
   route the packet takes through the network by imposing stacked MPLS
   labels to the packet.  To facilitate the incremental deployment of
   this new technology, this document describes a mechanism which allows
   the outermost LSP be replaced by an IP-based tunnel.

Status of This Memo

Copyright Notice

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

Table of Contents

1.  Introduction

   MPLS-SPRING [I-D.ietf-spring-segment-routing-mpls] is a MPLS-based
   source routing paradigm in which a sender of a packet is allowed to
   partially or completely specify the route the packet takes through
   the network by imposing stacked MPLS labels to the packet.  To
   facilitate the incremental deployment of this new technology, this
   document describes a mechanism which allows the outermost LSP to be
   replaced by an IP-based tunnel (e.g., MPLS-in-IP/GRE tunnel
   [RFC4023], MPLS-in-UDP tunnel [RFC7510] or MPLS-in-L2TPv3 tunnel
   [RFC4817] and etc) when the nexthop along the LSP is not MPLS-SPRING-
   enabled.  The tunnel destination address would be the address of the
   egress of the outmost LSP (e.g., the egress of the active node
   segment).

   This mechanism is much useful in the MPLS-SPRING-based Service
   Function Chainning (SFC) case [I-D.xu-sfc-using-mpls-spring] where
   only a few specific routers (e.g., Service Function Forwarders (SFF)
   and classifiers) are required to be MPLS-SPRING-capable while the
   remaining routers are just required to support IP forwarding
   capability.  In addition, this mechanism is also useful in some
   specific Traffic Engineering scenarios where only a few routers
   (e.g., the entry and exit nodes of each plane in the dual-plane
   network ) are specified as segments of explicit paths.  In this way,

only a few routers are required to support the MPLS-SPRING capability
while all the other routers just need to support IP forwarding
capability, which would significantly reduce the deployment cost of
this new technology.  Furthermore, since there is no need to run any
other label distribution protocol (e.g., LDP), the network
provisioning is greatly simplified, which is one of the major claimed
benefits of the MPLS-SPRING technology.

## 1.1.  Requirements Language

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

## 2.  Terminology

This memo makes use of the terms defined in [RFC3031] and
[I-D.ietf-spring-segment-routing-mpls].

## 3.  Packet Forwarding Procedures

Assume an MPLS-SPRING-enabled router X prepares to forward an MPLS
packet to the next node segment (i.e., the node segment of MPLS-
SPRING- enabled router Y) which is identified by the top label of the
MPLS packet.  If the next-hop router of the best path to Y is a non-
MPLS router, X couldn't map the packet's top label into an Next Hop
Label Forwarding Entry (NHLFE) , even though the top label itself is
a valid incoming label.  Acorrding to the following specification as
quoted from Section 3.22 of [RFC3031], the MPLS packet would be
discarded in the currenet MPLS implementations:

> "When a labeled packet is traveling along an LSP, it may
> occasionally happen that it reaches an LSR at which the ILM does
> not map the packet's incoming label into an NHLFE, even though the
> incoming label is itself valid...Unless it can be determined
> (through some means outside the scope of this document) that
> neither of these situations obtains, the only safe procedure is to
> discard the packet. "

This document proposes an improved procedure to deal with the above
case.  The basic idea is to set an IP tunnel towards the egress of
topmost LSP as the NHLFE of that incoming label.  More specifically,
if the label is not a Penultimate Hop Popping (PHP) label (i.e., the
NP-flag [I-D.ietf-isis-segment-routing-extensions] associated with
the corresponding prefix SID of that top label is set), X SHOULD swap
the label to the corresponding label significant to Y and then
encapsulate the MPLS packet into the IP-based tunnel towards Y.  The
tunnel destination address is the IP address of Y (e.g., the /32 or

/128 prefix FEC associated with that top label) and the tunnel source
address is the IP address of X.  If the label is a PHP label and not
at the bottom of the label stack, X SHOULD pop that label before
performing the above MPLS over IP encapsulation.  The IP encapsulated
MPLS packet would be forwarded according to the IP routing table.
Upon receipt of that IP encapsulated MPLS packet, Y would decapsulate
it and then process the decapsulated MPLS packet accordingly.  As for
which tunnel encapsulation type should be used by X, it can be
manually specified on X or be learnt from Y's advertisement of its
tunnel encapsulation capability.  How to advertise the tunnel
encapsulation capability using IS-IS or OSPF are specified in
[I-D.xu-isis-encapsulation-cap] and [I-D.ietf-ospf-encapsulation-cap]
respectively.

4.  Acknowledgements

   Thanks Joel Halpern, Bruno Decraene and Loa Andersson for their
   insightful comments on this draft.

5.  IANA Considerations

   No IANA action is required.

6.  Security Considerations

   TBD.

7.  References

7.1.  Normative References

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

7.2.  Informative References

   [I-D.ietf-isis-segment-routing-extensions]
              Previdi, S., Filsfils, C., Bashandy, A., Gredler, H.,
              Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS
              Extensions for Segment Routing", draft-ietf-isis-segment-
              routing-extensions-07 (work in progress), June 2016.

   [I-D.ietf-ospf-encapsulation-cap]
             Xu, X., Decraene, B., Raszuk, R., Chunduri, U., Contreras,
             L., and L. Jalil, "Advertising Tunnelling Capability in
             OSPF", draft-ietf-ospf-encapsulation-cap-00 (work in
             progress), October 2015.

   [I-D.ietf-spring-segment-routing-mpls]
             Filsfils, C., Previdi, S., Bashandy, A., Decraene, B.,
             Litkowski, S., Horneffer, M., Shakir, R.,
             jefftant@gmail.com, j., and E. Crabbe, "Segment Routing
             with MPLS data plane", draft-ietf-spring-segment-routing-
             mpls-05 (work in progress), July 2016.

   [I-D.xu-isis-encapsulation-cap]
             Xu, X., Decraene, B., Raszuk, R., Chunduri, U., Contreras,
             L., and L. Jalil, "Advertising Tunnelling Capability in
             IS-IS", draft-xu-isis-encapsulation-cap-06 (work in
             progress), November 2015.

   [I-D.xu-sfc-using-mpls-spring]
             Xu, X., Shah, H., Contreras, L., and d.
             daniel.bernier@bell.ca, "Service Function Chaining Using
             MPLS-SPRING", draft-xu-sfc-using-mpls-spring-06 (work in
             progress), July 2016.

   [RFC3031]  Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol
             Label Switching Architecture", RFC 3031,
             DOI 10.17487/RFC3031, January 2001,
             <http://www.rfc-editor.org/info/rfc3031>.

   [RFC4023]  Worster, T., Rekhter, Y., and E. Rosen, Ed.,
             "Encapsulating MPLS in IP or Generic Routing Encapsulation
             (GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005,
             <http://www.rfc-editor.org/info/rfc4023>.

   [RFC4817]  Townsley, M., Pignataro, C., Wainner, S., Seely, T., and
             J. Young, "Encapsulation of MPLS over Layer 2 Tunneling
             Protocol Version 3", RFC 4817, DOI 10.17487/RFC4817, March
             2007, <http://www.rfc-editor.org/info/rfc4817>.

   [RFC7510]  Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black,
             "Encapsulating MPLS in UDP", RFC 7510,
             DOI 10.17487/RFC7510, April 2015,
             <http://www.rfc-editor.org/info/rfc7510>.

Authors' Addresses

    Xiaohu Xu (editor)
    Huawei

    Email: xuxiaohu@huawei.com


    Robert Raszuk
    Bloomberg LP

    Email: robert@raszuk.net


    Uma Chunduri

    Email: uma.chunduri@gmail.com


    Luis M. Contreras
    Telefonica I+D

    Email: luismiguel.contrerasmurillo@telefonica.com


    Luay Jalil
    Verizon

    Email: luay.jalil@verizon.com

CCAMP Working Group                                         X. Zhang
Internet-Draft                                   Huawei Technologies
Intended status: Standards Track                           A. Sharma
Expires: April 29, 2017                                     Infinera
                                                             X. Liu
                                                            Ericsson
                                                    October 26, 2016

         A YANG Data Model for Optical Transport Network Topology
                   draft-zhang-ccamp-l1-topo-yang-05

Abstract

   A transport network is a server-layer network designed to provide
   connectivity services for a client-layer network to carry the client
   traffic transparently across the server-layer network resources.  A
   transport network can be constructed from equipments utilizing any of
   a number of different transport technologies such as the evolving
   Optical Transport Networks (OTN) or packet transport as provided by
   the MPLS-Transport Profile (MPLS-TP).

   This draft describes a YANG data model to describe the topologies of
   an Optical Transport Network (OTN).  It is independent of control
   plane protocols and captures topological and resource related
   information pertaining to OTN.  This model enables clients, which
   interact with a transport domain controller via a REST interface, for
   OTN topology related operations such as obtaining the relevant
   topology resource information.

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

time.  It is inappropriate to use Internet-Drafts as reference
material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 29, 2017.

Copyright Notice

Table of Contents

1.  Introduction

A transport network is a server-layer network designed to provide
connectivity services for a client-layer network to carry the client
traffic transparently across the server-layer network resources.  A
transport network can be constructed of equipments utilizing any of a
number of different transport technologies such as the Optical
Transport Networks (OTN) or packet transport as provided by the MPLS-
Transport Profile (MPLS-TP).

This document defines a data model of an OTN network topology, using
YANG [RFC6020].  The model can be used by an application exposing to
a transport controller via a REST interface.  Furthermore, it can be
used by an application for the following purposes (but not limited
to):

   o  To obtain a whole view of the network topology information of its
      interest;

   o  To receive notifications with regard to the information change of
      the OTN topology;

   o  To enforce the establishment and update of a network topology with
      the characteristic specified in the data model, e.g., by a client
      controller;

   The YANG model defined in this draft is independent of control plane
   protocols and captures topology related information pertaining to an
   Optical Transport Networks (OTN)-electrical layer, as the scope
   specified by [RFC7062] and [RFC7138].  Furthermore, it is not a
   stand-alone model, but augmenting from the TE topology YANG model
   defined in [I-D.ietf-teas-yang-te-topo].

   Optical network technologies, including fixed Dense Wavelength
   Switched Optical Network (WSON) and flexible optical networks
   (a.k.a., flexi-grid networks), are covered in
   [I-D.ietf-ccamp-wson-yang] and [I-D.vergara-ccamp-flexigrid-yang],
   respectively.

2.  Terminology and Notations

   A simplified graphical representation of the data model is used in
   this document.  The meaning of the symbols in the YANG data tree
   presented later in this draft is defined in
   [I-D.ietf-netmod-rfc6087bis].  They are provided below for reference.

   o  Brackets "[" and "]" enclose list keys.

   o  Abbreviations before data node names: "rw" means configuration
      (read-write) and "ro" state data (read-only).

   o  Symbols after data node names: "?" means an optional node, "!"
      means a presence container, and "*" denotes a list and leaf-list.

   o  Parentheses enclose choice and case nodes, and case nodes are also
      marked with a colon (":").

   o  Ellipsis ("...") stands for contents of subtrees that are not
      shown.

3.  YANG Data Model for OTN Topology

3.1.  the YANG Tree


   module: ietf-otn-topology
   augment /nd:networks/nd:network/nd:network-types/tet:te-topology:
      +--rw otn-topology!
   augment /nd:networks/nd:network:
      +--rw name?    string
   augment /nd:networks/nd:network/nd:node:
      +--rw name?    string
   augment /nd:networks/nd:network/lnk:link/tet:te/tet:config:
      +--rw available-odu-info* [priority]
      │  +--rw priority    uint8
      │  +--rw odulist* [odu-type]
      │     +--rw odu-type    identityref
      │     +--rw number?     uint16
      +--rw distance?              uint32
   augment /nd:networks/nd:network/lnk:link/tet:te/tet:state:
      +--ro available-odu-info* [priority]
      │  +--ro priority    uint8
      │  +--ro odulist* [odu-type]
      │     +--ro odu-type    identityref
      │     +--ro number?     uint16
      +--ro distance?              uint32
   augment /nd:networks/nd:network/nd:node/lnk:termination-point
   /tet:te/tet:config:
      +--rw client-facing?           empty
      +--rw tpn?                     uint16
      +--rw tsg?                     identityref
      +--rw protocol-type?           identityref
      +--rw adaptation-type?         adaptation-type
      +--rw sink-adapt-active?       boolean
      +--rw source-adapt-active?     boolean
      +--rw tributaryslots
      │  +--rw values*    uint8
      +--rw supported-payload-types* [index]
         +--rw index           uint16
         +--rw payload-type?   string
   augment /nd:networks/nd:network/nd:node/lnk:termination-point/
   tet:te/tet:state:
      +--ro client-facing?           empty
      +--ro tpn?                     uint16

```
    +--ro tsg?                      identityref
    +--ro protocol-type?            identityref
    +--ro adaptation-type?          adaptation-type
    +--ro sink-adapt-active?        boolean
    +--ro source-adapt-active?      boolean
    +--ro tributaryslots
    |  +--ro values*    uint8
    +--ro supported-payload-types* [index]
        +--ro index           uint16
        +--ro payload-type?   string
```

## 3.2.  Explanation of the OTN Topology Data Model

As can be seen, from the data tree shown in Section 3.1, the YANG
module presented in this draft augments from a more generic Traffic
Engineered (TE) network topology data model, i.e., the ietf-te-
topology.yang as specified in [I-D.ietf-teas-yang-te-topo].  The
entities and their attributes, such as node, termination points and
links, are still applicable for describing an OTN topology and the
model presented in this draft only specifies with technology-specific
attributes/information.  For example, if the data plane complies with
ITU-T G.709 (2012) standards, the switching-capability and encoding
attributes MUST be filled as OTN-TDM and G.709 ODUk(Digital Path)
respectively.

Note the model in this draft re-uses some attributes defined in ietf-
transport-types.yang, which is specified in
[I-D.sharma-ccamp-otn-service-model].

One of the main augmentations in this model is that it allows to
specify the type of ODU container and the number a link can support
per priority level.  For example, for a ODU3 link, it may advertise
32*ODU0, 16*ODU1, 4*ODU2 available, assuming only a single priority
level is supported.  If one of ODU2 resource is taken to establish a
ODU path, then the availability of this ODU link is updated as
24*ODU0, 12*ODU1, 3*ODU2 available.  If there are equipment hardware
limitations, then a subset of potential ODU type SHALL be advertised.
For instance, an ODU3 link may only support 4*ODU2.

## 3.3.  The YANG Code

<CODE BEGINS> file " ietf-otn-topology@2016-10-26.yang"

module ietf-otn-topology {

```
     yang-version 1;

     namespace "urn:ietf:params:xml:ns:yang:ietf-otn-topology";
     prefix "otntopo";

     import ietf-network {
      prefix "nd";
     }

     import ietf-network-topology {
      prefix "lnk";
     }

     import ietf-te-topology {
      prefix "tet";
     }

     import ietf-transport-types {
      prefix "tran-types";
     }

     organization
      "Internet Engineering Task Force (IETF) CCAMP WG";
     contact
      "
      WG List: <mailto:ccamp@ietf.org>

      ID-draft editor:
       Xian ZHANG (zhang.xian@huawei.com);
       Anurag Sharma (AnSharma@infinera.com);
      ";

     description
      "This module defines a protocol independent Layer 1/ODU
      topology data model.";

     revision 2016-10-26 {
      description
       "Initial version.";
      reference
       "draft-zhang-ccamp-l1-topo-yang-04.txt";
     }

     /*
     typedef
     */

     typedef adaptation-type {
```

```
     type enumeration {
      enum CBR {
       description "Constant Bit Rate.";
      }
      enum ATMvp {
       description "ATM VP.";
      }
      enum GFP {
       description "Generic Framing Procedure.";
      }
      enum NULL {
       description "NULL";
      }
      enum PRBS {
       description "Pseudo Random Binary Sequence";
      }
      enum RSn {
       description "SDH/SONET section";
      }
      enum ODUj-21 {
       description "ODU payload type 21";
      }
      enum ETHERNET_PP-OS {
       description "ETHERNET_PP-OS, for ODU 2 only";
      }
      enum CBRx {
       description "CBRx(0.. 1.25G), for ODU0 only";
      }
      enum ODU {
       description "Optical Data Unit";
      }
     }

     description
      "Defines a type representing the adaptation type
      on the termination point.";
    }


    /*
    Groupings
    */

    grouping otn-topology-type {
     container otn-topology {
      presence "indicates a topology type of Optical
        Transport Network (OTN)-electrical layer.";
       description "otn topology type";
```

```
    }
     description "otn-topology-type";
    }

    grouping otn-topology-attributes {
     leaf name {
      type string;
      description "the topology name";
     }
     description "name attribute for otn topology";
    }

    grouping otn-node-attributes {
     description "otn-node-attributes";
     leaf name {
      type string;
      description "a name for this node.";
     }
    }

    grouping otn-link-attributes {
     description "otn link attributes";

     list available-odu-info{
      key "priority";
      max-elements "8";

      description
       "List of ODU type and number on this link";
      leaf priority {
       type uint8 {
        range "0..7";
       }
       description "priority";
      }

      list odulist {
       key "odu-type";

       description
        "the list of available ODUs per priority level";

       leaf odu-type {
        type identityref{
         base tran-types:tributary-protocol-type;
        }
        description "the type of ODU";
       }
```

```
       leaf number {
        type uint16;
        description "the number of odu type supported";
       }
      }
     }

     leaf distance {
      type uint32;
      description "distance in the unit of kilometers";
     }
   }

   grouping otn-tp-attributes {
    description "otn-tp-attributes";

    leaf client-facing {
     type empty;
     description
       "if present, it means this tp is a client-facing tp.
        adding/dropping client signal flow.";
    }

    leaf tpn {
     type uint16 {
      range "0..4095";
     }
     description
       "Tributary Port Number. Applicable in case of mux services.";
      reference
      "RFC7139: GMPLS Signaling Extensions for Control of Evolving
      G.709 Optical Transport Networks.";
    }

    leaf tsg {
     type identityref {
      base tran-types:tributary-slot-granularity;
     }
     description "Tributary slot granularity.";
     reference
      "G.709/Y.1331, February 2012: Interfaces for the Optical
      Transport Network (OTN)";
    }

    leaf protocol-type {
     type identityref {
      base tran-types:tributary-protocol-type;
     }
```

```
      description "Protocol type for the Termination Point.";
     }

     leaf adaptation-type {
      type adaptation-type;
      description
      "This attribute indicates the type of the supported
      adaptation function at the termination point.";
      reference
      "G.874.1, January 2002: Optical transport network (OTN):
      Protocol-neutral management information model for the
      network element view.";
     }

     leaf sink-adapt-active {
      type boolean;
      description
      "This attribute allows for activation or deactivation of
      the sink adaptation function. The value of TRUE means active.";

      reference
      "G.874.1, January 2002: Optical transport network (OTN):
      Protocol-neutral management information model for the
      network element view ";
     }

     leaf source-adapt-active {
      type boolean;
      description
      "This attribute allows for activation or deactivation of
      the sink adaptation function. The value of TRUE
      means active.";
      reference
      "G.874.1, January 2002: Optical transport network (OTN):
      Protocol-neutral management information model for
      the network element view ";
     }

      container tributaryslots {
       description
       "A list of tributary slots used by the ODU
       Termination Point.";
       leaf-list values {
        type uint8;
        description
        "Tributary slot value.";
        reference
        "G.709/Y.1331, February 2012: Interfaces for the
```

```
     Optical Transport Network (OTN)";
    }
  }

 list supported-payload-types{
  key "index";

  description "supported payload types of a TP";

  leaf index {
   type uint16;
   description "payload type index";
  }

  leaf payload-type {
   type string;
   description "the payload type supported by this client
   tp";
   reference
   "http://www.iana.org/assignments/gmpls-sig-parameters
   /gmpls-sig-parameters.xhtml
   not: the payload type is defined as the generalized PIDs
   in GMPLS.";
  }
 }
}


/*
* Data nodes
*/
augment "/nd:networks/nd:network/nd:network-types/tet:te-topology" {
 uses otn-topology-type;
 description "augment network types to include otn newtork";
}

augment "/nd:networks/nd:network" {
 when "nd:network-types/tet:te-topology/otn-topology" {
  description "Augment only for otn network";
 }
 uses otn-topology-attributes;
 description "Augment network configuration";
}

augment "/nd:networks/nd:network/nd:node" {
 when "../nd:network-types/tet:te-topology/otn-topology" {
  description "Augment only for otn network";
 }
```

```
      description "Augment node configuration";
      uses otn-node-attributes;
     }

    augment "/nd:networks/nd:network/lnk:link/tet:te/tet:config" {
     when "../../../nd:network-types/tet:te-topology/otn-topology" {
      description "Augment only for otn network.";
     }
     description "Augment link configuration";

     uses otn-link-attributes;
    }

    augment "/nd:networks/nd:network/lnk:link/tet:te/tet:state" {
     when "../../../nd:network-types/tet:te-topology/otn-topology" {
      description "Augment only for otn network.";
     }
     description "Augment link state";

     uses otn-link-attributes;
    }

    augment "/nd:networks/nd:network/nd:node/"
     +"lnk:termination-point/tet:te/tet:config" {
     when "../../../../nd:network-types/tet:te-topology/otn-topology" {
      description "Augment only for otn network";
     }
     description "OTN TP attributes config in a ODU topology.";
     uses otn-tp-attributes;
    }

    augment "/nd:networks/nd:network/nd:node/"
     +"lnk:termination-point/tet:te/tet:state" {
     when "../../../../nd:network-types/tet:te-topology/otn-topology" {
      description "Augment only for otn network";
     }
     description "OTN TP attributes state in a ODU topology.";
     uses otn-tp-attributes;
    }
   }

  <CODE ENDS>
```

4.  IANA Considerations

    TBD.

5.  Manageability Considerations

    TBD.

6.  Security Considerations

    The data following the model defined in this draft is exchanged via,
    for example, the interface between an orchestrator and a transport
    network controller.  The security concerns mentioned in
    [I-D.ietf-teas-yang-te-topo] for using ietf-te-topology.yang model
    also applies to this draft.

    The YANG module defined in this document can be accessed via the
    RESTCONF protocol defined in [I-D.ietf-netconf-restconf], or maybe
    via the NETCONF protocol [RFC6241].

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

    Editors note: to list specific subtrees and data nodes and their
    sensitivity/vulnerability.

7.  Acknowledgements

    We would like to thank Igor Bryskin, Zhe Liu, Dieter Beller and
    Daniele Ceccarelli for their comments and discussions.

8.  Contributors

    Baoquan Rao
    Huawei Technologies
    Email: raobaoquan@huawei.com

    Sergio Belotti
    Alcatel Lucent
    Email: Sergio.belotti@alcatel-lucent.com

    Huub van Helvoort
    Hai Gaoming BV
    the Netherlands

   Email: huubatwork@gmail.com

9.  References

9.1.  Normative References

   [I-D.ietf-netconf-restconf]
             Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
             Protocol", draft-ietf-netconf-restconf-17 (work in
             progress), September 2016.

   [I-D.ietf-netmod-rfc6087bis]
             Bierman, A., "Guidelines for Authors and Reviewers of YANG
             Data Model Documents", draft-ietf-netmod-rfc6087bis-09
             (work in progress), October 2016.

   [I-D.ietf-teas-yang-te-topo]
             Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and
             O. Dios, "YANG Data Model for TE Topologies", draft-ietf-
             teas-yang-te-topo-05 (work in progress), July 2016.

   [I-D.sharma-ccamp-otn-service-model]
             ansharma@infinera.com, a., Rao, R., and X. Zhang, "OTN
             Service YANG Model", draft-sharma-ccamp-otn-service-
             model-00 (work in progress), July 2016.

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

   [RFC6020]  Bjorklund, M., Ed., "YANG - A Data Modeling Language for
             the Network Configuration Protocol (NETCONF)", RFC 6020,
             DOI 10.17487/RFC6020, October 2010,
             <http://www.rfc-editor.org/info/rfc6020>.

   [RFC6241]  Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
             and A. Bierman, Ed., "Network Configuration Protocol
             (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
             <http://www.rfc-editor.org/info/rfc6241>.

   [RFC7062]  Zhang, F., Ed., Li, D., Li, H., Belotti, S., and D.
             Ceccarelli, "Framework for GMPLS and PCE Control of G.709
             Optical Transport Networks", RFC 7062,
             DOI 10.17487/RFC7062, November 2013,
             <http://www.rfc-editor.org/info/rfc7062>.

   [RFC7138]  Ceccarelli, D., Ed., Zhang, F., Belotti, S., Rao, R., and
              J. Drake, "Traffic Engineering Extensions to OSPF for
              GMPLS Control of Evolving G.709 Optical Transport
              Networks", RFC 7138, DOI 10.17487/RFC7138, March 2014,
              <http://www.rfc-editor.org/info/rfc7138>.

9.2.  Informative References

   [I-D.ietf-ccamp-wson-yang]
              Lee, Y., Dhody, D., Zhang, X., Guo, A., Lopezalvarez, V.,
              King, D., and B. Yoon, "A Yang Data Model for WSON Optical
              Networks", draft-ietf-ccamp-wson-yang-03 (work in
              progress), July 2016.

   [I-D.vergara-ccamp-flexigrid-yang]
              Madrid, U., Lopezalvarez, V., Dios, O., King, D., Lee, Y.,
              and Z. Ali, "YANG data model for Flexi-Grid Optical
              Networks", draft-vergara-ccamp-flexigrid-yang-03 (work in
              progress), July 2016.

Authors' Addresses

   Xian Zhang
   Huawei Technologies
   F3-5-B R&D Center, Huawei Industrial Base, Bantian, Longgang District
   Shenzhen, Guangdong  518129
   P.R.China

   Email: zhang.xian@huawei.com


   Anurag Sharma
   Infinera

   Email: ansharma@infinera.com


   Xufeng Liu
   Ericsson

   Email: xufeng.liu@ericsson.com