Network Working Group
Internet Draft
Intended status: Informational

Vishnu Pavan Beeram
Juniper Networks
Ina Minei
Google, Inc
Yakov Rekhter
Juniper Networks
Ebben Aries
Facebook
Dante Pacella
Verizon

Expires: September 08, 2015
March 08, 2015

RSVP-TE Scalability - Recommendations
draft-beeram-mpls-rsvp-te-scaling-01

Status of this Memo

Copyright Notice

Abstract

RSVP-TE [RFC3209] describes the use of standard RSVP [RFC2205] to establish Label Switched Paths (LSPs). As such, RSVP-TE inherited some properties of RSVP that adversely affect its control plane scalability. Specifically these properties are (a) reliance on periodic refreshes for state synchronization between RSVP neighbors and for recovery from lost RSVP messages, (b) reliance on refresh timeout for stale state cleanup, and (c) lack of any mechanisms by which a receiver of RSVP messages can apply back pressure to the sender(s) of these messages.

Subsequent to [RFC2205] and [RFC3209] further enhancements to RSVP and RSVP-TE have been developed. In this document we describe how an implementation of RSVP-TE can use these enhancements to address the above mentioned properties to improve RSVP-TE control plane scalability.

Conventions used in this document

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

Table of Contents

1. Introduction

RSVP-TE [RFC3209] describes the use of standard RSVP [RFC2205] to establish Label Switched Paths (LSPs). As such, RSVP-TE inherited some properties of RSVP that adversely affect its control plane scalability. Specifically these properties are (a) reliance on periodic refreshes for state synchronization between RSVP neighbors and for recovery from lost RSVP messages, (b) reliance on refresh timeout for stale state cleanup, and (c) lack of any mechanisms by which a receiver of RSVP messages can apply back pressure to the sender(s) of these messages. The following elaborates on this.

1.1. Reliance on refreshes and refresh timeouts

Standard RSVP [RFC2205] maintains state via the generation of RSVP Path/Resv refresh messages. Refresh messages are used to both synchronize state between RSVP neighbors and to recover from lost RSVP messages. The use of Refresh messages to cover many possible failures has resulted in two operational problems. The first relates to scaling, the second relates to the reliability and latency of RSVP signaling.

The scaling problem is linked to the control plane resource requirements of running RSVP-TE. The resource requirements increase proportionally with the number of LSPs established by RSVP-TE. Each such LSP requires the generation, transmission, reception and processing of RSVP Path and Resv messages per refresh period. Supporting a large number of LSPs and the corresponding volume of refresh messages, presents a scaling problem for the RSVP-TE control plane.

The reliability and latency problem occurs when a triggered (non-refresh) RSVP message such as Path, Resv, or PathTear is lost in transmission. Standard RSVP [RFC2205] recovers from a lost message via RSVP refresh messages. In the face of transmission loss of RSVP messages, the end-to-end latency of RSVP signaling, and thus the end-to-end latency of RSVP-TE signaled LSP establishment, is tied to the refresh interval of the Label Switch Router(s) experiencing the loss. When end-to-end signaling is limited by the refresh interval, the delay incurred in the establishment or the change of an RSVP-TE signaled LSP may be beyond the range of what is acceptable in practice. This is because RSVP-TE ultimately controls establishment of the forwarding state required to realize RSVP-TE signaled LSPs. Thus delay incurred in the establishment or the change of such LSPs results in delaying the data plane convergence, which in turn adversely impacts the services that rely on the data plane.

One way to address the scaling problem caused by the refresh volume is to increase the refresh period, "R" as defined in Section 3.7 of [RFC2205]. Increasing the value of R provides linear improvement on RSVP-TE signaling overhead, but at the cost of increasing the time it takes to synchronize state. For the reasons mentioned in the previous paragraph, in the context of RSVP-TE signaled LSPs, increasing the time to synchronize state is not an acceptable option.

One way to address the reliability and latency of RSVP signaling is to decrease the refresh period R. Decreasing the value of R increases the probability that state will be installed in the face of message loss, but at the cost of increasing refresh message rate and associated processing requirements, which in turn adversely affects RSVP-TE control plane scalability.

An additional problem is the time to clean up the stale state after a tear message is lost. RSVP does not retransmit ResvTear or PathTear messages. If the sole tear message transmitted is lost, the stale state will only be cleaned up once the refresh timeout has expired. This may result in resources associated with the stale state being allocated for an unnecessary period of time. Note that even when the refresh period is adjusted, the refresh timeout must still expire since tear messages are not retransmitted. Decreasing the refresh timeout by decreasing the refresh interval will speed up timely stale state cleanup, but at the cost of increasing refresh message rate, which in turn adversely affects RSVP-TE control plane scalability.

1.2. Lack of back pressure

In standard RSVP, an RSVP speaker sends RSVP messages to a peer with no regard for whether the peer's RSVP control plane is busy. There is no control plane mechanism by which an RSVP speaker may apply back pressure to the peer by asking the peer to reduce the rate of RSVP messages that the peer sends to the speaker. RSVP-TE inherited this from standard RSVP. Lack of such a mechanism could result in RSVP-TE control plane congestion.

RSVP-TE control plane is especially susceptible to congestion during link/node failures, as such failures produce bursts of RSVP-TE messages: Path/Resv for re-routing LSPs affected by the failures, Path/Resv for setup of new backup LSPs (as required by RSVP-TE Fast Reroute [RFC4090]), Tear/Error messages for the affected LSPs. Note that the load on the RSVP-TE control plane caused by these bursts is in addition to the load due to the periodic refreshes of Path/Resv messages for the LSPs not affected by the failures.

RSVP-TE control plane congestion may result in loss of RSVP messages, which in turn have detrimental effects on the overall system behavior. Path/Resv refreshes lost by a peer’s busy control plane will cause refresh timeout for some or all of its existing RSVP-TE state on the peer, thus inadvertently deleting existing LSPs and disrupting traffic carried over these LSPs. Triggered Path/Resv lost by a peer’s busy control plane may result in failure to establish new backup LSPs used by RSVP-TE Fast Reroute [RFC4090] before the state for the corresponding protected primary LSPs times out, thus defeating the whole purpose of RSVP-TE Fast Reroute.

2. Recommendations

Subsequent to the publication of [RFC2205] and [RFC3209] further enhancements to RSVP and RSVP-TE have been developed. In this section we describe how these enhancements could be used to address the problems listed in Section 1.

2.1. Eliminating reliance on refreshes and refresh timeouts

To eliminate reliance on refreshes for both state synchronization between RSVP neighbors and for recovery from lost RSVP messages, as well as to address both the refresh volume and the reliability issues with RSVP mechanisms other than adjusting refresh rate, this document RECOMMENDS the following:

- Implement reliable delivery of Path/Resv messages using the procedures specified in [RFC2961].

- Indicate support for RSVP Refresh Overhead Reduction Extensions (as specified in Section 2 of [RFC2961] by default, with the ability to override the default via configuration.

- Make the value of the refresh interval configurable with the default value of 20 minutes.

To eliminate reliance on refresh timeouts, in addition to the above, this document RECOMMENDS the following:

- Implement reliable delivery of Tear/Err messages using the procedures specified in [RFC2961]

- Implement coupling the state of individual LSPs with the state of the corresponding RSVP-TE signaling adjacency. When an RSVP-TE speaker detects RSVP-TE signaling adjacency failure, the speaker MUST clean up the LSP state for all LSPs affected by the failed adjacency. The LSP state is the combination of "path state" maintained as Path State Block and "reservation state" maintained as Reservation State Block (see Section 2.1 of [RFC2205]).

- Use of Node-ID based Hello session ([RFC3209], [RFC4558]) for detection of RSVP-TE signaling adjacency failures. Make the value of the node hello_interval [RFC3209] configurable; increase the default value from 5 ms (as specified in Section 5.3 of [RFC3209]) to 9 seconds.

- Implement procedures specified in [draft-chandra-mpls-enhanced-frr-bypass] which describes methods to facilitate FRR that works independently of the refresh-interval.

2.2. Providing the ability to apply back pressure

To provide an RSVP speaker with the ability to apply back pressure to its peer(s) to reduce/eliminate RSVP-TE control plane congestion, in addition to the above, this document RECOMMENDS the following:

- Use lack of ACKs from a peer as an indication of peer's RSVP-TE control plane congestion, in which case the local system SHOULD throttle RSVP-TE messages to the affected peer. This has to be done on a per-peer basis.

- Retransmit of all RSVP-TE messages using exponential backoff, as specified in Section 6 of [RFC2961].

- Increase the Retry Limit (Rl), as defined in Section 6.2 of [RFC2961], from 3 to 7.

- Prioritize Tear/Error over trigger Path/Resv sent to a peer when the local system detects RSVP-TE control plane congestion in the peer.

2.3. Making Acknowledgements mandatory

The reliable message delivery mechanism specified in [RFC2961] states that "Nodes receiving a non-out of order message containing a

MESSAGE_ID object with the ACK_Desired flag set, SHOULD respond with a MESSAGE_ID_ACK object." To improve predictability of the system in terms of reliable message delivery this document RECOMMENDS that nodes receiving a non-out of order message containing a MESSAGE_ID object with the ACK_Desired flag set, MUST respond with a MESSAGE_ID_ACK object.

2.4. Clarifications on reaching Rapid Retry Limit (Rl)

According to section 6 of [RFC2961] "The staged retransmission will continue until either an appropriate MESSAGE_ID_ACK object is received, or the rapid retry limit, Rl, has been reached." The following clarifies what actions, if any, a router should take once Rl has been reached.

If it is the retransmission of Tear/Err messages and Rl has been reached, the router need not take any further actions.

If it is the retransmission of Path/Resv messages and Rl has been reached, then the router starts periodic retransmission of these messages every 30 seconds. The retransmitted messages MUST carry MESSAGE_ID object with ACK_Desired flag set. This periodic retransmission SHOULD continue until an appropriate MESSAGE_ID ACK object is received indicating acknowledgement of the (retransmitted) Path/Resv message.

2.5. Avoiding use of Router Alert IP Option

In RSVP-TE the Path message is carried in an IP packet that is addressed to the tail end of the LSP that is signaled using this message. To make all the intermediate/transit LSRs process this message, the IP packet carrying the message includes the Router Alert IP option. The same applies to the PathTear message.

An alternative to relying on the Router Alert IP option is to carry the Path or PathTear message as a sub-message of a Bundle message [RFC2961], as Bundle messages are "addressed directly to RSVP neighbors" and "SHOULD NOT be sent with the Router Alert IP option in their IP headers" [RFC2961]. Notice that since a Bundle message could contain only a single sub-message, this approach could be used to send just a single Path or PathTear message. This document RECOMMENDS implementing support for Bundle messages [RFC2961], and carrying Path and PathTear message(s) as sub-message(s) of a Bundle message.

2.6. Checking Data Plane readiness

In certain scenarios, like Make-Before-Break (MBB), a router needs to move traffic from an existing LSP to a new LSP in the least disruptive fashion. To accomplish this the data plane of the new LSP must be operational before the router moves the traffic.

A possible mechanism by which the router can determine whether the data plane of the new LSP is operational is specified in [draft-bonica-mpls-self-ping]. This document RECOMMENDS implementing this mechanism and using it whenever the ingress of an LSP needs to check whether the data plane of the LSP is operational.

3. Security Considerations

This document does not introduce new security issues. The security considerations pertaining to the original RSVP protocol [RFC2205] and RSVP-TE [RFC3209] remain relevant.

4. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC

5. Normative References

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

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

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

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

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

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

[draft-bonica-mpls-self-ping] Ron Bonica, et al., "LSP Self-Ping", draft-bonica-mpls-self-ping, (work in progress)

[draft-chandra-mpls-enhanced-frr-bypass] Chandra Ramachandran, et al., "Refresh Interval Independent FRR Facility Protection", draft-chandra-mpls-enhanced-frr-bypass, (work in progress)

6. Acknowledgments

Most of the text in Section 1.1 has been taken almost verbatim from [RFC2961].

Authors' Addresses

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

Ina Minei
Google, Inc
Email: inaminei@google.com

Yakov Rekhter
Juniper Networks
Email: yakov@juniper.net

Ebben Aries
Facebook
Email: exa@fb.com

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

Markus Jork
Juniper Networks
Email: mjork@juniper.net

Network Working Group   Daniele Ceccarelli (Editor)
Internet Draft   Ericsson

Intended status: Informational   Young Lee (Editor)
Expires: September 2015   Huawei

March 9, 2015

Framework for Abstraction and Control of Transport Networks

draft-ceccarelli-actn-framework-07.txt

Status of this Memo

Copyright Notice

Abstract

This draft provides a framework for abstraction and control of transport networks.

Table of Contents

1. Introduction

Transport networks have a variety of mechanisms to facilitate separation of data plane and control plane including distributed signaling for path setup and protection, centralized path computation for planning and traffic engineering, and a range of management and provisioning protocols to configure and activate network resources. These mechanisms represent key technologies for enabling flexible and dynamic networking.

Transport networks in this draft refer to a set of different type of connection-oriented networks, primarily Connection-Oriented Circuit Switched (CO-CS) networks and Connection-Oriented Packet Switched (CO-PS) networks. This implies that at least the following transport networks are in scope of the discussion of this draft: Layer 1(L1) and Layer 0 (L0) optical networks (e.g., Optical Transport Network (OTN), Optical Channel Data Unit (ODU), Optical Channel (OCh)/Wavelength Switched Optical Network (WSON)), Multi-Protocol Label Switching - Transport Profile (MPLS-TP), Multi-Protocol Label Switching - Traffic Engineering (MPLS-TE), as well as other emerging technologies with connection-oriented behavior. One of the characteristics of these network types is the ability of dynamic provisioning and traffic engineering such that resource guarantees can be provided to their clients.

One of the main drivers for Software Defined Networking (SDN) is a decoupling of the network control plane from the data plane. This separation of the control plane from the data plane has been already achieved with the development of MPLS/GMPLS [GMPLS] and PCE [PCE] for TE-based transport networks. One of the advantages of SDN is its logically centralized control regime that allows a global view of the underlying network under its control. Centralized control in SDN helps improve network resources utilization from a distributed network control. For TE-based transport network control, PCE is essentially equivalent to a logically centralized control for path computation function.

Two key aspects that need to be solved by SDN are:

. Network and service abstraction
. End to end coordination of multiple SDN and pre-SDN domains e.g. NMS, MPLS-TE or GMPLS.

As transport networks evolve, the need to provide network and service abstraction has emerged as a key requirement for operators; this implies in effect the virtualization of network resources so that the network is "sliced" for different tenants shown as a dedicated portion of the network resources

Particular attention needs to be paid to the multi-domain case, where Abstraction and Control of Transport Networks (ACTN) can facilitate virtual network operation via the creation of a single virtualized network or a seamless service. This supports operators in viewing and controlling different domains (at any dimension: applied technology, administrative zones, or vendor-specific technology islands) as a single virtualized network.

Network virtualization, in general, refers to allowing the customers to utilize a certain amount of network resources as if they own them and thus control their allocated resources in a way most optimal with higher layer or application processes. This empowerment of customer control facilitates introduction of new services and applications as the customers are permitted to create, modify, and delete their virtual network services. More flexible, dynamic customer control capabilities are added to the traditional VPN along with a customer specific virtual network view. Customers control a view of virtual network resources, specifically allocated to each one of them. This view is called an abstracted network topology. Such a view may be specific to the set of consumed services as well as to a particular customer. As the Customer Network Controller is envisioned to support a plethora of distinct applications, there would be another level of virtualization from the customer to individual applications.

The framework described in this draft is named Abstraction and Control of Transport Network (ACTN) and facilitates:

- Abstraction of the underlying network resources to higher-layer applications and users (customers); abstraction for a specific application or customer is referred to as virtualization in the ONF SDN architecture. [ONF-ARCH]

- Slicing infrastructure to connect multiple customers to meet specific customer’s service requirements;

- Creation of a virtualized environment allowing operators to view and control multi-subnet multi-technology networks into a single virtualized network;

- Possibility of providing a customer with abstracted network or abstracted services (totally hiding the network).

- A virtualization/mapping network function that adapts customer requests to the virtual resources (allocated to them) to the supporting physical network control and performs the necessary mapping, translation, isolation and security/policy enforcement, etc.; This function is often referred to as orchestration.

- The multi-domain coordination of the underlying transport domains, presenting it as an abstracted topology to the customers via open and programmable interfaces. This allows for the recursion of controllers in a customer-provider relationship.

The organization of this draft is as follows. Section 2 provides a discussion for a Business Model, Section 3 ACTN Architecture, Section 4 ACTN Applicability, and Section 5 ACTN Interface requirements.

2. Business Model of ACTN

The traditional Virtual Private Network (VPN) and Overlay Network (ON) models are built on the premise that one single network provider provides all virtual private or overlay networks to its customers. This model is simple to operate but has some disadvantages in accommodating the increasing need for flexible and dynamic network virtualization capabilities.

The ACTN model is built upon entities that reflect the current landscape of network virtualization environments. There are three key entities in the ACTN model [ACTN-PS]:

- Customers
- Service Providers
- Network Providers

2.1. Customers

Within the ACTN framework, different types of customers may be taken into account depending on the type of their resource needs, on their number and type of access. As example, it is possible to group them into two main categories:

Basic Customer: Basic customers include fixed residential users, mobile users and small enterprises. Usually the number of basic customers is high; they require small amounts of resources and are characterized by steady requests (relatively time invariant). A typical request for a basic customer is for a bundle of voice services and internet access. Moreover basic customers do not modify their services themselves; if a service change is needed, it is performed by the provider as proxy and they generally have very few dedicated resources (subscriber drop), with everything else shared on the basis of some SLA, which is usually best-efforts.

Advanced Customer: Advanced customers typically include enterprises, governments and utilities. Such customers can ask for both point to point and multipoint connectivity with high resource demand significantly varying in time and from customer to customer. This is one of the reasons why a bundled services offer is not enough but it is desirable to provide each of them with customized virtual network services. Advanced customers may own dedicated virtual resources, or share resources, but shared resources are likely to be governed by more complex SLA agreements; moreover they may have the ability to modify their service parameters directly (within the scope of their virtualized environments. As customers are geographically spread over multiple network provider domains, the necessary control and data interfaces to support such customer needs is no longer a single interface between the customer and one single network provider. With this premise, customers have to interface multiple providers to get their end-to-end network connectivity service and the associated topology information. Customers may have to support multiple virtual network services with different service objectives and QoS requirements. For flexible and dynamic applications, customers may want to control their allocated virtual network resources in a dynamic fashion. To allow that, customers should be given an abstracted view of topology on which they can perform the necessary control decisions and take the corresponding actions. ACTN's primary focus is Advanced Customers.

Customers of a given service provider can in turn offer a service to other customers in a recursive way. An example of recursiveness with 2 service providers is shown below.

- Customer (of service B)
- Customer (of service A) & Service Provider (of service B)
- Service Provider (of service A)
- Network Provider

```
   +------------------------------------------------------------+   ---
   |                                                            |    ^
   |                                     Customer (of service B)|    .
   | +--------------------------------------------------------+ |    B
   | |                                                        | |--- .
   | |Customer (of service A) & Service Provider(of service B)| | ^  .
   | | +---------------------------------------------------+  | | .  .
   | | |                                                   |  | | .  .
   | | |                    Service Provider (of service A)|  | | A  .
   | | |+------------------------------------------+       |  | | .  .
   | | ||                                          |       |  | | .  .
   | | ||                          Network provider|       |  | | v  v
   | | |+------------------------------------------+       |  | |------
   | | +---------------------------------------------------+  | |
   | +--------------------------------------------------------+ |
   +------------------------------------------------------------+
```

Figure 1: Network Recursiveness.

2.2. Service Providers

Service providers are the providers of virtual network services to their customers. Service providers may or may not own physical network resources. When a service provider is the same as the network provider, this is similar to traditional VPN models. This model works well when the customer maintains a single interface with a single provider. When customer location spans across multiple independent network provider domains, then it becomes hard to facilitate the creation of end-to-end virtual network services with this model.

A more interesting case arises when network providers only provide infrastructure while service providers directly interface their customers. In this case, service providers themselves are customers of the network infrastructure providers. One service provider may need to keep multiple independent network providers as its end-users span geographically across multiple network provider domains.

```
Customer            X -----------------------------------X

Service Provider A  X -----------------------------------X

Network Provider B                     X-----------------X

Network Provider A  X------------------X
```

The ACTN network model is predicated upon this three tier model and is summarized in figure below:

```
                    +----------------------+
                    |       customer       |
                    +----------------------+
                              |
                              |   /\  Service/Customer specific
                              |   ||  Abstract Topology
                              |   ||
                    +----------------------+  E2E abstract
                    |  Service Provider    | topology creation
                    +----------------------+
                    /         |            \
```

```
                 /          |            \  Network Topology
                /           |             \ (raw or abstract)
               /            |              \
   +------------------+   +------------------+   +------------------+
   |Network Provider 1|   |Network Provider 2|   |Network Provider 3|
   +------------------+   +------------------+   +------------------+
```

Figure 2: Three tier model.

There can be multiple types of service providers.

- Data Center providers: can be viewed as a service provider type as they own and operate data center resources to various WAN clients, they can lease physical network resources from network providers.
- Internet Service Providers (ISP): can be a service provider of internet services to their customers while leasing physical network resources from network providers.
- Mobile Virtual Network Operators (MVNO): provide mobile services to their end-users without owning the physical network infrastructure.

The network provider space is the one where recursiveness occurs. A customer-provider relationship between multiple service providers can be established leading to a hierarchical architecture of controllers within service provider network.

2.3. Network Providers

Network Providers are the infrastructure providers that own the physical network resources and provide network resources to their customers. The layered model proposed by this draft separates the concerns of network providers and customers, with service providers acting as aggregators of customer requests.

3. ACTN architecture

This section provides a high-level control and interface model of ACTN.

The ACTN architecture, while being aligned with the ONF SDN architecture [ONF-ARCH], is presenting a 3-tiers reference model. It allows for hierarchy and recursiveness not only of SDN controllers but also of traditionally controlled domains. It defines three types of controllers depending on the functionalities they implement. The main functionalities that are identified are:

. Multi domain coordination function: With the definition of domain being "everything that is under the control of the same controller",it is needed to have a control entity that oversees the specific aspects of the different domains and to build a single abstracted end-to-end network topology in order to coordinate end-to-end path computation and path/service provisioning.

. Virtualization/Abstraction function: To provide an abstracted view of the underlying network resources towards customer, being it the client or a higher level controller entity. It includes computation of customer resource requests into virtual network paths based on the global network-wide abstracted topology and the creation of an abstracted view of network slices allocated to each customer, according to customer-specific virtual network objective functions, and to the customer traffic profile.

. Customer mapping function: In charge of mapping customer VN setup commands into network provisioning requests to the Physical Network Controller (PNC) according to business OSS/NMS provisioned static or dynamic policy. Moreover it provides mapping and translation of customer virtual network slices into physical network resources

. Virtual service coordination: Virtual service coordination function in ACTN incorporates customer service-related knowledge into the virtual network operations in order to seamlessly operate virtual networks while meeting customer's service requirements.

The functionality is covering two types of services:

- Service-aware Connectivity Services: This category includes all the network service operations used to provide connectivity between customer end-points while meeting policies and service related constraints. The data model for this category would include topology entities such as virtual nodes, virtual links, adaptation and termination points and service-related entities such as policies and service related constraints. (See Section 4.2.2)

- Network Function Virtualization Services: These kinds of services are usually setup between customers' premises and service provider premises and are provided mostly by cloud providers or content delivery providers. The context may include, but not limited to a security function like firewall, a traffic optimizer, the provisioning of storage or computation capacity where the customer does not care whether the service is implemented in a given data center or another. These services may be hosted virtually by the provider or physically part of the network. This allows the service provider to hide his own resources (both network and data centers) and divert customer requests where most suitable. This is also known as "end points mobility" case and introduces new concepts of traffic and service provisioning and resiliency. (e.g. Virtual Machine mobility)." (See Section 4.2.3)

About the Customer service-related knowledge it includes:

- VN Service Requirements: The end customer would have specific service requirements for the VN including the customer endpoints access profile as well as the E2E customer service objectives. The ACTN framework architectural "entities" would monitor the E2E service during the lifetime of VN by focusing on both the connectivity provided by the network as well as the customer service objectives. These E2E service requirements go beyond the VN service requirements and include customer infrastructure as well.

- Application Service Policy: Apart for network connectivity, the customer may also require some policies for application specific features or services. The ACTN framework would take these application service policies and requirements into consideration while coordinating the virtual network operations, which require end customer connectivity for these advanced services.

While the "types" of controller defined are shown in Figure 3 below and are the following:

. CNC - Customer Network Controller

- . MDSC - Multi Domain Service Coordinator
- . PNC - Physical Network Controller

```
   VPN customer         NW Mobile Customer     ISP NW service Customer
       |                         |                           |
   +-------+                 +-------+                   +-------+
   | CNC-A |                 | CNC-B |                   | CNC-C |
   +-------+                 +-------+                   +-------+
         \___________            |             _____________/
          ----------             |             ------------
                     \           |            /
                      +-----------------------+
                      |         MDSC          |
                      +-----------------------+
            __________/          |            \__________
            ----------           |             ------------____
           /                     |                         \
   +-------+                 +-------+                   +-------+
   |  PNC  |                 |  PNC  |                   |  PNC  |
   +-------+                 +-------+                   +-------+
        | GMPLS          /      |                      /    \
        | trigger       /       |                     /      \
       --------       __----      +-----+  __         +-----+     \
      (        )       (    )_    | PNC |__           | PCE |      \
      -        -      ( Phys )    +-----+            +-----+    -----
     (  GMPLS   )      (Netw)       |                /        (     )
    (  Physical  )      ----         |               /        ( Phys. )
     (  Network )                 -----        -----          ( Net )
      -        -                 (     )      (     )           -----
      (        )                ( Phys. )    ( Phys  )
       --------                  ( Net )      ( Net )
                                  -----        -----
```

Figure 3: ACTN Control Hierarchy

3.1. Customer Network Controller

A Virtual Network Service is instantiated by the Customer Network Controller via the CMI (CNC-MDSC Interface). As the Customer Network Controller directly interfaces the application stratum, it understands multiple application requirements and their service needs. It is assumed that the Customer Network Controller and the MDSC have a common knowledge on the end-point interfaces based on their business negotiation prior to service instantiation. End-point interfaces refer to customer-network physical interfaces that connect customer premise equipment to network provider equipment. Figure 10 in Appendix shows an example physical network topology that supports multiple customers. In this example, customer A has three end-points A.1, A.2 and A.3. The interfaces between customers and transport networks are assumed to be 40G OTU links.

In addition to abstract networks, ACTN allows to provide the CNC with services. Example of services include connectivity between one of the customer’s end points with a given set of resources in a data center from the service provider.

3.2. Multi Domain Service Coordinator

The MDSC (Multi Domain Service Coordinator) sits between the CNC (the one issuing connectivity requests) and the PNCs (Physical Network Controllersr - the ones managing the physical network resources). The MDSC can be collocated with the PNC, especially in those cases where the service provider and the network provider are the same entity.

The internal system architecture and building blocks of the MDSC are out of the scope of ACTN. Some examples can be found in the Application Based Network Operations (ABNO) architecture [ABNO] and the ONF SDN architecture [ONF-ARCH].

The MDSC is the only building block of the architecture that is able to implement all the four ACTN main functionalities, i.e. multi domain coordination function, virtualization/abstraction function, customer mapping function and virtual service coordination.
A hierarchy of MDSCs can be foreseen for scalability and administrative choices.

```
   +-------+                 +-------+                 +-------+
   | CNC-A |                 | CNC-B |                 | CNC-C |
   +-------+                 +-------+                 +-------+
         \____________           |             ___________/
          ----------             |             ----------
                     \           |            /
                      +-----------------------+
                      |         MDSC          |
                      +-----------------------+
            __________/          |            \__________
            ----------           |             -----------____
           /                     |                        \
   +----------+              +----------+             +--------+
   |   MDSC   |              |   MDSC   |             |  MDSC  |
   +----------+              +----------+             +--------+
        |                     /    |                   /    \
        |                    /     |                  /      \
     +-----+           +-----+  +-----+            +-----+  +-----+
     | PNC |           | PNC |  | PNC |            | PNC |  | PNC |
     +-----+           +-----+  +-----+            +-----+  +-----+
```

Figure 4: Controller recursiveness

A key requirement for allowing recursion of MDSCs is that a single interface needs to be defined both for the north and the south bounds.

In order to allow for multi-domain coordination a 1:N relationship must be allowed between MDSCs and between MDSCs and PNCs (i.e. 1 parent MDSC and N child MDSC or 1 MDSC and N PNCs). In addition to that it could be possible to have also a M:1 relationship between MDSC and PNC to allow for network resource partitioning/sharing among different customers not necessarily connected to the same MDSC (e.g. different service providers).

It should be noted that the interface between the parent MDSC and a child MDSC does not introduce any complexity as it is "internal" and "transparent" from the perspective of the CNCs and the PNCs and it makes use of the same interface model and its primitives as the CMI and MPI.

3.3. Physical Network Controller

The physical network controller is the one in charge of configuring the network elements, monitoring the physical topology of the network and passing it, either raw or abstracted, to the MDSC.

The internal architecture of the PNC, his building blocks and the way it controls its domain, are out of the scope of ACTN. Some examples can be found in the Application Based Network Operations (ABNO) architecture [ABNO] and the ONF SDN architecture [ONF-ARCH]

The PNC, in addition to being in charge of controlling the physical network, is able to implement two of the four ACTN main functionalities: multi domain coordination function and virtualization/abstraction function
A hierarchy of PNCs can be foreseen for scalability and administrative choices.

3.4. ACTN interfaces

To allow virtualization and multi domain coordination, the network has to provide open, programmable interfaces, in which customer applications can create, replace and modify virtual network resources and services in an interactive, flexible and dynamic fashion while having no impact on other customers. Direct customer control of transport network elements and virtualized services is not perceived as a viable proposition for transport network providers due to security and policy concerns among other reasons. In addition, as discussed in the previous section, the network control plane for transport networks has been separated from data plane and as such it is not viable for the customer to directly interface with transport network elements.

While the current network control plane is well suited for control of physical network resources via dynamic provisioning, path computation, etc., a multi service domain controller needs to be built on top of physical network controller to support network virtualization. On a high-level, virtual network control refers to a mediation layer that performs several functions:

Figure 4 depicts a high-level control and interface architecture for ACTN. A number of key ACTN interfaces exist for deployment and operation of ACTN-based networks. These are highlighted in Figure 4 (ACTN Interfaces) below:

```
                .--------------
               -------------   |
              | Application |--
               -------------
                     ^
                     | I/F A                 --------
                     v                      (        )
                --------------             -          -
               | Customer     |           (  Customer  )
               |  Network     |--------->(    Network   )
               |   Controller |           (            )
                --------------             -          -
                     ^                      (        )
                     | I/F B                 --------
                     v                        ^    ^
                --------------                :    :
               | MultiDomain  |               :     .
               |  Service     |               :      .
               |   Coordinator|            --------   . I/F E
                --------------            (        )   .
                     ^                   -          -   .
                     | I/F C            (  Physical  )   .
                     v                 (    Network   )   .
                  ---------------       (            )     --------
                 |               |<----> -          -     (        )
                --------------   |        (        )     -          -
               | Physical     |--          --------     (  Physical  )
               |  Network     |<---------------------->(    Network   )
               |   Controller |         I/F D           (            )
                --------------                           -          -
                                                          (        )
                                                           --------
```

Figure 4: ACTN Interfaces

The interfaces and functions are described below:

. Interface A: A north-bound interface (NBI) that will communicate the service request or application demand. A request will include specific service properties, including: services, topology, bandwidth and constraint information.

- Interface B: The CNC-MDSC Interface (CMI) is an interface between a Customer Network Controller and a Multi Service Domain Controller. It requests the creation of the network resources, topology or services for the applications. The Virtual Network Controller may also report potential network topology availability if queried for current capability from the Customer Network Controller.

- Interface C: The MDSC-PNC Interface (MPI) is an interface between a Multi Domain Service Coordinator and a Physical Network Controller. It communicates the creation request, if required, of new connectivity of bandwidth changes in the physical network, via the PNC. In multi-domain environments, the MDSC needs to establish multiple MPIs, one for each PNC, as there are multiple PNCs responsible for its domain control.

- Interface D: The provisioning interface for creating forwarding state in the physical network, requested via the Physical Network Controller.

- Interface E: A mapping of physical resources to overlay resources.

The interfaces within the ACTN scope are B and C.

3.5. Work in Scope of ACTN

This section provides a summary of use-cases in terms of two categories: (i) service-specific requirements; (ii) network-related requirements.

Service-specific requirements listed below are uniquely applied to the work scope of ACTN. Service-specific requirements are related to virtual service coordination function defined in Section 3. These requirements are related to customer’s VNs in terms of service policy associated with VNs such as service performance objectives, VN endpoint location information for certain required service-specific functions (e.g., security and others), VN survivability requirement, or dynamic service control policy, etc.

Network-related requirements are related to virtual network operation function defined in Section 3. These requirements are related to multi-domain and multi-layer signaling, routing, protection/restoration and synergy, re-optimization/re-grooming, etc. These requirements are not inherently unique for the scope of ACTN but some of these requirements are in scope of ACTN, especially for coherent/seamless operation aspect of multiple controller hierarchy.

The following table gives an overview of service-specific requirements and network-related requirements respectively for each ACTN use-case and identifies the work in scope of ACTN.

Details on these requirements will be developed into the information model in [ACTN-Info].

Use-case	Service-specific Requirements	Network-related Requirements	ACTN Work Scope
[Cheng]	- E2E service provisioning - Performance monitoring - Resource utilization abstraction	- Multi-layer (L2/L2.5) coordination - VNO for multi-domain transport networks	- Dynamic multi-layer coordination based on utilization is in scope of ACTN - YANG for utilization abstraction
[Dhody]	- Service awareness/ coordination between P/O.	- POI Performance monitoring - Protection/ Restoration synergy	- Performance related data model may be in scope of ACTN - Customer’s VN survivability policy enforcement for protection/restoration is unique to ACTN.
[Fang]	- Dynamic VM migration (service), Global load balancing (utilization efficiency), Disaster recovery - Service-aware network	- On-demand virtual circuit request - Network Path Connection request	- Multi-destination service selection policy enforcement and its related primitives/information are unique to

	query - Service Policy Enforcement		ACTN. - Service-aware network query and its data model can be extended by ACTN.
[Klee]		- Two stage path computation E2E signaling coordination - Abstraction of inter-domain info - Enforcement of network policy (peering, domain preference) - Network capability exchange (pull/push, abstraction level, etc.)	- Multi-domain service policy coordination to network primitives is in scope of ACTN
[Kumaki]	- On-demand VN creation - Multi-service level for VN - VN survivability /diversity/confidentiality		- All of the service-specific lists in the left column is unique to ACTN.
[Lopez]	- E2E accounting and resource usage data	- E2E connection management, path provisioning - E2E network	- Escalation of performance and fault management

	- E2E service policy enforcement	monitoring and fault management	data to CNC and the policy enforcement for this area is unique to ACTN.
[Shin]	- Current network resource abstraction Endpoint/DC dynamic selection (for VM migration)	- LB for recovery - Multi-layer routing and optimization coordination	- Multi-layer routing and optimization are related to VN's dynamic endpoint selection policy.
[Xu]	- Dynamic service control policy enforcement - Dynamic service control	- Traffic monitoring - SLA monitoring	- Dynamic service control policy enforcement and its control primitives are in scope of ACTN - Data model to support traffic monitoring data is an extension of YANG model ACTN can extend.

4. References

4.1. Informative References

[PCE] Farrel, A., Vasseur, J.-P., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", IETF RFC 4655, August 2006.

[PCE-S] Crabbe, E, et. al., "PCEP extension for stateful PCE",draft-ietf-pce-stateful-pce, work in progress.

[GMPLS] Manning, E., et al., "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", RFC 3945, October 2004.

[NFV-AF] "Network Functions Virtualization (NFV); Architectural Framework", ETSI GS NFV 002 v1.1.1, October 2013.

[ACTN-PS] Y. Lee, D. King, M. Boucadair, R. Jing, L. Contreras Murillo, "Problem Statement for Abstraction and Control of Transport Networks", draft-leeking-actn-problem-statement, work in progress.

[ONF] Open Networking Foundation, "OpenFlow Switch Specification Version 1.4.0 (Wire Protocol 0x05)", October 2013.

[ABNO] King, D., and Farrel, A., "A PCE-based Architecture for Application-based Network Operations", draft-farrkingel-pce-abno-architecture, work in progress.

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

[Cheng] W. Cheng, et. al., "ACTN Use-cases for Packet Transport Networks in Mobile Backhaul Networks", draft-cheng-actn-ptn-requirements, work in progress.

[Dhody] D. Dhody, et. al., "Packet Optical Integration (POI) Use Cases for Abstraction and Control of Transport Networks (ACTN)", draft-dhody-actn-poi-use-case, work in progress.

[Fang] L. Fang, "ACTN Use Case for Multi-domain Data Center Interconnect", draft-fang-actn-multidomain-dci, work in progress.

[Klee] K. Lee, H. Lee, R. Vilata, V. Lopez, "ACTN Use-case for On-demand E2E Connectivity Services in Multiple Vendor Domain Transport Networks", draft-klee-actn-connectivity-multi-vendor-domains, work in progress.

[Kumaki] K. Kumaki, T. Miyasaka, "ACTN : Use case for Multi Tenant VNO ", draft-kumaki-actn-multitenant-vno, work in progress.

[Lopez] D. Lopez (Ed), "ACTN Use-case for Virtual Network Operation for Multiple Domains in a Single Operator Network", draft-lopez-actn-vno-multidomains, work in progress.

[Shin] J. Shin, R. Hwang, J. Lee, "ACTN Use-case for Mobile Virtual Network Operation for Multiple Domains in a Single Operator Network", draft-shin-actn-mvno-multi-domain, work in progress.

[Xu] Y. Xu, et. al., "Use Cases and Requirements of Dynamic Service Control based on Performance Monitoring in ACTN Architecture", draft-xu-actn-perf-dynamic-service-control, work in progress.

5. Contributors

Authors' Addresses

Daniele Ceccarelli (Editor)
Ericsson
Torshamnsgatan,48
Stockholm, Sweden
Email: daniele.ceccarelli@ericsson.com

Young Lee (Editor)
Huawei Technologies
5340 Legacy Drive
Plano, TX 75023, USA
Phone: (469)277-5838
Email: leeyoung@huawei.com

Luyuan Fang
Email: luyuanf@gmail.com

Diego Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
28006 Madrid, Spain
Email: diego@tid.es

Sergio Belotti
Alcatel Lucent
Via Trento, 30
Vimercate, Italy
Email: sergio.belotti@alcatel-lucent.com

Daniel King
Lancaster University
Email: d.king@lancaster.ac.uk

Dhruv Dhoddy
Huawei Technologies
dhruv.ietf@gmail.com

Internet Engineering Task Force H. Chen
Internet-Draft R. Li
Intended status: Standards Track Huawei Technologies
Expires: December 3, 2017 G. Cauchie

A. Retana
Cisco Systems, Inc.
N. So
Tata Communications
F. Xu
M. Toy
Verizon
V. Liu
China Mobile
L. Liu
Fijitsu
June 1, 2017

TE LSP Crossing Topology-Transparent Zones
draft-chen-teas-rsvp-ttz-02.txt

Abstract

A topology-transparent zone is virtualized as the edges of the zone fully connected. This document presents the procedures for the establishment of Traffic Engineering (TE) LSPs crossing Topology-Transparent Zones.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The number of routers in a network becomes larger and larger as the Internet traffic keeps growing. Through splitting the network into multiple areas, we can extend the network further. However, there are a number of issues when a network is split further into more areas.

At first, dividing a network into more areas is a very challenging and time consuming since it is involved in significant network architecture changes.

Secondly, the services carried by the network may be interrupted while the network is being split into more areas.

Furthermore, it is complex for a TE LSP crossing areas to be setup. In one option, a TE path crossing areas is computed by using collaborating PCEs [RFC5441] through PCEP[RFC5440], which is not easy to configure.

Topology-transparent zone (TTZ) resolves these issues. This document briefs TTZ and presents the procedures for the establishment of TE LSPs crossing TTZs.

2. Conventions Used in This Document

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

3. Overview of Topology-Transparent Zone (TTZ)

A Topology-Transparent Zone (TTZ) is identified by an Identifier (ID) called TTZ ID, and it includes a group of routers and a number of links connecting the routers. A TTZ is in an IGP area.

In addition to having the functions of an IGP area, an IGP TTZ makes some improvements on an IGP area, which include:

- o An IGP TTZ is virtualized as the TTZ edge routers connected.
- o An IGP TTZ receives the link state information about the topology outside of the TTZ, stores the information in the TTZ and floods the information through the TTZ to the routers outside of the TTZ.

The figure below shows an area containing a TTZ: TTZ 600.

```
                 TTZ 600
                   \
                    \ ^~^~^~^~^~^~^~^~^~^~^~^~
                     (                        )
    ===[R15]========(==[R61]------------[R63]==)======[R29]===
        ||         (   |    \          /    |   )       ||
        ||         (   |     \        /     |   )       ||
        ||         (   |      \      /      |   )       ||
        ||         (   |    ___\    /       |   )       ||
        ||         (   |   /   [R71]        |   )       ||
        ||         (   | [R73] /    \       |   )       ||
        ||         (   |      /      \      |   )       ||
        ||         (   |     /        \     |   )       ||
        ||         (   |    /          \    |   )       ||
    ===[R17]========(==[R65]------------[R67]==)======[R31]===
         \\          (//                    \\)       //
          ||         //v~v~v~v~v~v~v~v~v~v~v~\\      ||
          ||        //                        \\     ||
          ||       //                          \\    ||
           \\     //                            \\   //
       ======[R23]==============================[R25]=====
             //                                     \\
            //                                       \\
```

Figure 1: An Example of TTZ

The area comprises routers R15, R17, R23, R25, R29 and R31. It also contains TTZ 600, which comprises routers R61, R63, R65, R67, R71 and R73, and the links connecting them.

There are two types of routers in a TTZ: TTZ internal routers and TTZ edge routers. A TTZ internal router is a router inside the TTZ and its adjacent routers are in the TTZ. A TTZ edge router is a router inside the TTZ and has at least one adjacent router that is outside of the TTZ.

The TTZ in the figure above comprises four TTZ edge routers R61, R63, R65 and R67. Each TTZ edge router is connected to at least one router outside of the TTZ. For instance, router R61 is a TTZ edge router since it is connected to router R15, which is outside of the TTZ.

In addition, the TTZ comprises two TTZ internal routers R71 and R73. A TTZ internal router is not connected to any router outside of the TTZ. For instance, router R71 is a TTZ internal router since it is just connected to routers R61, R63, R65, R67 and R73 in the TTZ.

A TTZ hides the information inside the TTZ from the outside. It does not directly distribute any internal information about the TTZ to a router outside of the TTZ.

For instance, the TTZ in the figure above does not send the information about TTZ internal router R71 to any router outside of the TTZ in the routing domain; it does not send the information about the link between TTZ router R61 and R65 to any router outside of the TTZ.

From a router outside of the TTZ, a TTZ is seen as a group of routers fully connected. For instance, router R15 in the figure above, which is outside of TTZ 600, sees TTZ 600 as a group of TTZ edge routers: R61, R63, R65 and R67. These four TTZ edge routers are fully connected.

The cost of the "link" from one edge router to another edge router is the cost of the shortest path between these two routers. The bandwidth of the "link" is the maximum bandwidth of a path between the two routers.

In addition, a router outside of the TTZ sees TTZ edge routers having normal connections to the routers outside of the TTZ. For example, router R15 sees four TTZ edge routers R61, R63, R65 and R67, which have the normal connections to R15, R29, R17 and R23, R25 and R31 respectively.

4. Set up TE LSPs crossing TTZs

On a source node, we can configure a TE LSP from the source to a destination crossing TTZs in the same way as we configure it without any TTZs. This is because the source node is not aware of any TTZs.

For example, on node R15 in Figure 1, to set up a TE LSP from R15 to R31, we just configure the TE LSP by giving its source R15, its destination R31, and some constraints such as bandwidth as needed.

On the source node, it computes the path to the destination based on the configuration of the TE LSP. It just sees a full mess connection of edge nodes for every TTZ. Thus the computation of the path is done in the same way as it is done without any TTZ. After the path is computed, the source node starts to signal the LSP automatically along the path.

For example, on node R15 in Figure 1, it computes the path to the destination R31. It sees the full mess connection of four TTZ edge nodes R61, R63, R65 and R67 in its topology. It computes the path in the same way as before and may get the path: R15 - R61 - R67 - R31. And then it signals the TE LSP along this path. It sends a RSVP-TE PATH message to R61.

When an edge node of a TTZ receives a PATH message, it checks if the next hop in the ERO in the message is another edge node of the TTZ. If so, it computes the path segment to the other edge node and continues to signal the TE LSP along the path segment computed.

For instance, when R61, which is an edge node of a TTZ, receives the PATH message, it computes the path segment to the other edge node R67 (Supposed that the path segment is: R61 - R71 - R67) and continues to signal the TE LSP to R67 along the path segment computed. It sends a PATH message to R71, which sends a PATH message to R67, which sends a PATH message to R31.

```
                   TTZ 600
                   \
                    \ ^~^~^~^~^~^~^~^~^~^~^~^~
      Source    51   (                        )
    ===[R15]========(==[R61]------------[R63]==)======[R29]===
        ||         (   |    \          /    |   )       ||
        ||         (   |     \        /     |   )       ||
        ||         (   |      \11    /      |   )       ||
        ||         (   |    ___\    /       |   )       ||
        ||         (   |   /   [R71]        |   )       ||
        ||         (   | [R73] /    \       |   )       ||
        ||         (   |      /      \      |   )       ||
        ||         (   |     /        \17   |   )       ||
        ||         (   |    /          \    |   )  71   ||
    ===[R17]========(==[R65]------------[R67]==)======[R31]===
         \\          (//                    \\)       //Destination
          ||         //v~v~v~v~v~v~v~v~v~v~v~\\      ||
          ||        //                        \\     ||
          ||       //                          \\    ||
           \\     //                            \\  //
       ======[R23]==============================[R25]=====
             //                                     \\
            //                                       \\
```

Figure 2: LSP from R15 to R31

When R31 receives the PATH message from R67, it allocates a label (e.g., 71), reserves the bandwidth as needed, and sends a RESV message with the label (71) to R67. It sets the forwarding entry for the TE LSP using label 71 as inbound label.

When R67 receives the RESV message from R31, it allocates a label (e.g., 17), and sends a RESV message with the label (17) to R71. It also sets the cross connect for the TE LSP using labels 17 and 71 as inbound label and outbound label respectively.

When R71 receives the RESV message with the label (17) from R67, it allocates a label (e.g., 11), and sends a RESV message with the label (11) to R61. It sets the cross connect for the TE LSP using labels 11 and 17 as inbound label and outbound label respectively.

When R61 receives the RESV message with the label (11) from R71, it allocates a label (e.g., 51), and sends a RESV message with the label (51) to R15. It sets the cross connect for the TE LSP using labels 51 and 11 as inbound label and outbound label respectively.

When R15 receives the RESV message with the label (51) from R61, it sets the forwarding entry for the TE LSP using label 51 as outbound label. At this point, the set up of TE LSP from R15 to R31 is done.

The source node (i.e., head-end LSR) sets the "label recording requested" flag in the SESSION_ATTRIBUTE object for LSPs requesting local protection. This will cause all LSRs to record their INBOUND labels.

For a TE LSP crossing a TTZ, we may assume that it goes into the TTZ through an in edge node of the TTZ and goes out of the TTZ from a out edge node of the TTZ.

For example, the TE LSP crossing TTZ 600 in Figure 2 is from R15 to R61 to R71 to R67 to R31. The LSP goes into TTZ 600 through the edge node R61, which is the in edge node. The LSP goes out of TTZ 600 from the edge node R67, which is the out edge node.

On the in edge node of the TTZ for the TE LSP, it does not record all the INBOUND labels inside the TTZ in the RESV message to be sent to its previous hop node. It just records the INBOUND label of the out edge node.

For example, R61 (the in edge node of TTZ 600 for the TE LSP in Figure 2) just keeps the INBOUND label 17 of R67 (the out edge node). It does not record any other INBOUND labels inside TTZ 600. It will remove the INBOUND label 11 of the TTZ internal node R71. Thus the RESV message sent by R61 to its previous hop node R15 records two INBOUND labels 17 and 71, which are the INBOUND labels of R67 and R31 respectively.

On the out edge node of the TTZ for the TE LSP, it does not record all the hops inside the TTZ in the PATH message to its next hop node.

It just records one hop from the in edge to the out edge.

5. Security Considerations

The mechanism described in this document does not raise any new security issues for the RSVP-TE protocols.

6. IANA Considerations

There is not any requirement for IANA to assign a code point.

7. Normative References

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

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <http://www.rfc-editor.org/info/rfc3031>.

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

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

[RFC5441] Vasseur, JP., Ed., Zhang, R., Bitar, N., and JL. Le Roux, "A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths", RFC 5441, DOI 10.17487/RFC5441, April 2009, <http://www.rfc-editor.org/info/rfc5441>.

[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>.

Authors' Addresses

Huaimo Chen
Huawei Technologies
Boston, MA
USA

Email: huaimo.chen@huawei.com

Renwei Li
Huawei Technologies
2330 Central Expressway
Santa Clara, CA
USA

Email: renwei.li@huawei.com

Gregory Cauchie
FRANCE

Email: greg.cauchie@gmail.com

Alvaro Retana
Cisco Systems, Inc.
7025 Kit Creek Rd.
Raleigh, NC 27709
USA

Email: aretana@cisco.com

Ning So
Tata Communications
USA

Email: ningso01@gmail.com

Fengman Xu
Verizon
2400 N. Glenville Dr
Richardson, TX 75082
USA

Email: fengman.xu@verizon.com

Mehmet Toy
Verizon
USA

Email: mehmet.toy@verizon.com

Vic Liu
China Mobile
No.32 Xuanwumen West Street, Xicheng District
Beijing, 100053
China

Email: liu.cmri@gmail.com

Lei Liu
Fijitsu
USA

Email: liulei.kddi@gmail.com

TEAS Working Group
Internet-Draft
Intended Status: Informational
Expires: July 30, 2017

X. Zhang
H. Zheng, Ed.
Huawei Technologies
R. Gandhi, Ed.
Z. Ali
Cisco Systems, Inc.
P. Brzozowski
ADVA Optical
January 26, 2017

RSVP-TE Signaling Procedure for End-to-End GMPLS Restoration and Resource Sharing

draft-ietf-teas-gmpls-resource-sharing-proc-08

Abstract

In non-packet transport networks, there are requirements where Generalized Multi-Protocol Label Switching (GMPLS) end-to-end recovery scheme needs to employ restoration Label Switched Path (LSP) while keeping resources for the working and/or protecting LSPs reserved in the network after the failure occurs.

This document reviews how the LSP association is to be provided using Resource Reservation Protocol - Traffic Engineering (RSVP-TE) signaling in the context of GMPLS end-to-end recovery scheme when using restoration LSP where failed LSP is not torn down. In addition, this document discusses resource sharing-based setup and teardown of LSPs as well as LSP reversion procedures. No new signaling extensions are defined by this document, and it is strictly informative in nature.

Status of this Memo

Table of Contents

1. Introduction

Generalized Multi-Protocol Label Switching (GMPLS) [RFC3945] defines a set of protocols, including Open Shortest Path First - Traffic Engineering (OSPF-TE) [RFC4203] and Resource ReserVation Protocol - Traffic Engineering (RSVP-TE) [RFC3473]. These protocols can be used to set up Label Switched Paths (LSPs) in non-packet transport networks. The GMPLS protocol extends MPLS to support interfaces capable of Time Division Multiplexing (TDM), Lambda Switching and Fiber Switching. These switching technologies provide several protection schemes [RFC4426][RFC4427] (e.g., 1+1, 1:N and M:N).

Resource Reservation Protocol - Traffic Engineering (RSVP-TE) signaling has been extended to support various GMPLS recovery schemes, such as end-to-end recovery [RFC4872] and segment recovery [RFC4873]. As described in [RFC6689], an ASSOCIATION object with Association Type "Recovery" [RFC4872] can be signaled in the RSVP Path message to identify the LSPs for restoration. Also, an ASSOCIATION object with Association Type "Resource Sharing" [RFC4873] can be signaled in the RSVP Path message to identify the LSPs for resource sharing. [RFC6689] Section 2.2 reviews the procedure for providing LSP associations for GMPLS end-to-end recovery and Section 2.4 reviews the procedure for providing LSP associations for sharing resources.

Generally GMPLS end-to-end recovery schemes have the restoration LSP set up after the failure has been detected and notified on the working LSP. For recovery scheme with revertive behavior, a restoration LSP is set up while working LSP and/or protecting LSP are not torn down in control plane due to a failure. In non-packet transport networks, as working LSPs are typically set up over preferred paths, service providers would like to keep resources associated with the working LSPs reserved. This is to make sure that the service can be reverted to the preferred path (working LSP) when the failure is repaired to provide deterministic behavior and guaranteed Service Level Agreement (SLA).

In this document, we review procedures for GMPLS LSP associations, resource sharing based LSP setup, teardown, and LSP reversion for non-packet transport networks, including the following:

o Review the procedure for providing LSP associations for the GMPLS end-to-end recovery using restoration LSP where working and protecting LSPs are not torn down and resources are kept reserved in the network after the failure.

o In [RFC3209], the make-before-break (MBB) method assumes the old and new LSPs share the SESSION object and signal Shared Explicit

(SE) flag in SESSION_ATTRIBUTE object for sharing resources. According to [RFC6689], an ASSOCIATION object with Association Type "Resource Sharing" in the Path message enables the sharing of resources across LSPs with different SESSION objects. The procedure for resource sharing using the SE flag in conjunction with an ASSOCIATION object is discussed in this document.

o When using end-to-end recovery scheme with revertive behavior, methods for LSP reversion and resource sharing are summarized in this document.

This document is strictly informative in nature and does not define any RSVP-TE signaling extensions.

2. Conventions Used in This Document

2.1. Terminology

The reader is assumed to be familiar with the terminology in [RFC3209], [RFC3473], [RFC4872] and [RFC4873]. The terminology for GMPLS recovery is defined in [RFC4427].

2.2. Acronyms and Abbreviations

GMPLS: Generalized Multi-Protocol Label Switching

LSP: An MPLS Label Switched Path

MBB: Make Before Break

MPLS: Multi-Protocol Label Switching

RSVP: Resource ReSerVation Protocol

SE: Shared Explicit flag

TDM: Time Division Multiplexing

TE: Traffic Engineering

3. Overview

The GMPLS end-to-end recovery scheme, as defined in [RFC4872] and being considered in this document, switches normal traffic to an alternate LSP that is not even partially established only after the working LSP failure occurs. The new alternate route is selected at the LSP head-end node, it may reuse resources of the failed LSP at intermediate nodes and may include additional intermediate nodes and/or links.

3.1. Examples of Restoration Schemes

Two forms of end-to-end recovery schemes, 1+R restoration and 1+1+R restoration are described in the following sections. Other forms of end-to-end recovery schemes also exist and they can use these signaling techniques.

3.1.1. 1+R Restoration

One example of the recovery scheme considered in this document is 1+R recovery. The 1+R recovery scheme is exemplified in Figure 1. In this example, a working LSP on path A-B-C-Z is pre-established. Typically after a failure detection and notification on the working LSP, a second LSP on path A-H-I-J-Z is established as a restoration LSP. Unlike a protecting LSP which is set up before the failure, a restoration LSP is set up when needed, after the failure.

```
+-----+    +-----+     +-----+     +-----+
|  A  +----+  B  +-----+  C  +-----+  Z  |
+--+--+    +-----+     +-----+     +--+--+
    \                                /
     \                              /
   +--+--+      +-----+        +--+--+
   |  H  +------+  I  +--------+  J  |
   +-----+      +-----+        +-----+
```

Figure 1: An Example of 1+R Recovery Scheme

During failure switchover with 1+R recovery scheme, in general, working LSP resources are not released so that working and restoration LSPs coexist in the network. Nonetheless, working and restoration LSPs can share network resources. Typically when the failure has recovered on the working LSP, the restoration LSP is no longer required and is torn down while the traffic is reverted to the original working LSP.

3.1.2. 1+1+R Restoration

Another example of the recovery scheme considered in this document is 1+1+R. In 1+1+R, a restoration LSP is set up for the working LSP and/or the protecting LSP after the failure has been detected, and this recovery scheme is exemplified in Figure 2.

```
              +-----+       +-----+        +-----+
              |  D  +-------+  E  +--------+  F  |
              +--+--+       +-----+        +--+--+
                /                              \
               /                                \
           +--+--+    +-----+     +-----+     +--+--+
           |  A  +----+  B  +-----+  C  +-----+  Z  |
           +--+--+    +-----+     +-----+     +--+--+
               \                                /
                \                              /
              +--+--+       +-----+        +--+--+
              |  H  +-------+  I  +--------+  J  |
              +-----+       +-----+        +-----+
```

Figure 2: An Example of 1+1+R Recovery Scheme

In this example, a working LSP on path A-B-C-Z and a protecting LSP on path A-D-E-F-Z are pre-established. After a failure detection and notification on the working LSP or protecting LSP, a third LSP on path A-H-I-J-Z is established as a restoration LSP. The restoration LSP in this case provides protection against failure of both the working and protecting LSPs. During failure switchover with 1+1+R recovery scheme, in general, failed LSP resources are not released so that working, protecting and restoration LSPs coexist in the network. The restoration LSP can share network resources with the working LSP, and it can share network resources with the protecting LSP. Typically, the restoration LSP is torn down when the traffic is reverted to the original LSP and it is no longer needed.

There are two possible models when using a restoration LSP with 1+1+R recovery scheme:

o A restoration LSP is set up after either a working or protecting LSP fails. Only one restoration LSP is present at a time.

o A restoration LSP is set up after both working and protecting LSPs fail. Only one restoration LSP is present at a time.

3.1.2.1. 1+1+R Restoration - Variants

Two other possible variants exist when using a restoration LSP with 1+1+R recovery scheme:

- o A restoration LSP is set up after either a working or protecting LSP fails. Two different restoration LSPs may be present, one for the working LSP and one for the protecting LSP.

- o Two different restoration LSPs are set up after both working and protecting LSPs fail, one for the working LSP and one for the protecting LSP.

In all these models, if a restoration LSP also fails, it is torn down and a new restoration LSP is set up.

3.2. Resource Sharing by Restoration LSP

```
                           +-----+      +-----+
                           |  F  +------+  G  +--------+
                           +--+--+      +-----+        |
                              |                        |
                              |                        |
    +-----+    +-----+     +--+--+      +-----+     +--+--+
    |  A  +----+  B  +-----+  C  +--X---+  D  +-----+  E  |
    +-----+    +-----+     +-----+      +-----+     +-----+
```

Figure 3: Resource Sharing in 1+R Recovery Scheme

Using the network shown in Figure 3 as an example using 1+R recovery scheme, LSP1 (A-B-C-D-E) is the working LSP, and assume it allows for resource sharing when the LSP traffic is dynamically restored. Upon detecting the failure of a link along the LSP1, e.g. Link C-D, node A needs to decide which alternative path it will use to signal restoration LSP and reroute traffic. In this case, A-B-C-F-G-E is chosen as the restoration LSP path and the resources on the path segment A-B-C are re-used by this LSP. The working LSP is not torn down and co-exists with the restoration LSP. When the head-end node A signals the restoration LSP, nodes C, F, G and E reconfigure the resources (as listed in Table 1 of this document) to set up the LSP by sending cross-connection command to the data plane.

In the recovery scheme employing revertive behavior, after the failure is repaired, the resources on nodes C and E need to be reconfigured to set up the working LSP (using a procedure described in Section 4.3 of this document) by sending cross-connection command to the data plane. The traffic is then reverted back to the original working LSP.

4. RSVP-TE Signaling Procedure

4.1. Restoration LSP Association

Where GMPLS end-to-end recovery scheme needs to employ a restoration LSP while keeping resources for the working and/or protecting LSPs reserved in the network after the failure, the restoration LSP is set up with an ASSOCIATION object that has Association Type set to "Recovery" [RFC4872], the Association ID and the Association Source set to the corresponding Association ID and the Association Source signaled in the Path message of the LSP it is restoring. For example, when a restoration LSP is signaled for a failed working LSP, the ASSOCIATION object in the Path message of the restoration LSP contains the Association ID and Association Source set to the Association ID and Association Source signaled in the working LSP for the "Recovery" Association Type. Similarly, when a restoration LSP is set up for a failed protecting LSP, the ASSOCIATION object in the Path message of the restoration LSP contains the Association ID and Association Source set to the Association ID and Association Source signaled in the protecting LSP for the "Recovery" Association Type.

The procedure for signaling the PROTECTION object is specified in [RFC4872]. Specifically, the restoration LSP used for a working LSP is set up with P bit cleared in the PROTECTION object in the Path message of the restoration LSP and the restoration LSP used for a protecting LSP is set up with P bit set in the PROTECTION object in the Path message of the restoration LSP.

4.2. Resource Sharing-based Restoration LSP Setup

GMPLS LSPs can share resources during LSP setup if they have Shared Explicit (SE) flag set in the SESSION_ATTRIBUTE objects [RFC3209] in the Path messages that create them and:

- o As defined in [RFC3209], LSPs have identical SESSION objects and/or

- o As defined in [RFC6689], LSPs have matching ASSOCIATION object with Association Type set to "Resource Sharing" signaled in their Path messages. LSPs in this case can have different SESSION objects i.e. different Tunnel ID, Source and/or Destination signaled in their Path messages.

As described in [RFC3209], Section 2.5, the purpose of make-before-break is not to disrupt traffic, or adversely impact network operations while TE tunnel rerouting is in progress. In non-packet transport networks during the RSVP-TE signaling procedure, the nodes set up cross-connections along the LSP accordingly. Because the cross-connection cannot simultaneously connect a shared resource to different resources in two alternative LSPs, nodes may not be able to fulfill this request when LSPs share resources.

For LSP restoration upon failure, as explained in Section 11 of [RFC4872], the reroute procedure may re-use existing resources. The action of the intermediate nodes during the rerouting process to reconfigure cross-connections does not further impact the traffic since it has been interrupted due to the already failed LSP.

The node actions for setting up the restoration LSP can be categorized into the following:

Category	Action
Reusing existing resource on both input and output interfaces (nodes A & B in Figure 3).	This type of node needs to reserve the existing resources and no cross-connection command is needed.
Reusing existing resource only on one of the interfaces, either input or output interfaces and using new resource on the other interfaces. (nodes C & E in Figure 3).	This type of node needs to reserve the resources and send the re-configuration cross-connection command to its corresponding data plane node on the interfaces where new resources are needed and it needs to re-use the existing resources on the other interfaces.
Using new resources on both interfaces. (nodes F & G in Figure 3).	This type of node needs to reserve the new resources and send the cross-connection command on both interfaces.

Table 1: Node Actions During Restoration LSP Setup

Depending on whether the resource is re-used or not, the node actions differ. This deviates from normal LSP setup since some nodes do not need to re-configure the cross-connection. Also, the judgment whether the control plane node needs to send a cross-connection setup or modification command to its corresponding data plane node(s) relies on the check whether the LSPs are sharing resources.

4.3. LSP Reversion

If the end-to-end LSP recovery scheme employs the revertive behavior, as described in Section 3 of this document, traffic can be reverted from the restoration LSP to the working or protecting LSP after its failure is recovered. The LSP reversion can be achieved using two methods:

1. Make-while-break Reversion, where resources associated with a working or protecting LSP are reconfigured while removing reservations for the restoration LSP.

2. Make-before-break Reversion, where resources associated with a working or protecting LSP are reconfigured before removing reservations for the restoration LSP.

In non-packet transport networks, both of the above reversion methods will result in some traffic disruption when the restoration LSP and the LSP being restored are sharing resources and the cross-connections need to be reconfigured on intermediate nodes.

4.3.1. Make-while-break Reversion

In this reversion method, restoration LSP is simply requested to be deleted by the head-end. Removing reservations for restoration LSP triggers reconfiguration of resources associated with a working or protecting LSP on every node where resources are shared. The working or protecting LSP state was not removed from the nodes when the failure occurred. Whenever reservation for restoration LSP is removed from a node, data plane configuration changes to reflect reservations of working or protecting LSP as signaling progresses. Eventually, after the whole restoration LSP is deleted, data plane configuration will fully match working or protecting LSP reservations on the whole path. Thus reversion is complete.

Make-while-break, while being relatively simple in its logic, has a few limitations as follows which may not be acceptable in some networks:

- No rollback

If for some reason reconfiguration of data plane on one of the nodes to match working or protecting LSP reservations fails, falling back to restoration LSP is no longer an option, as its state might have already been removed from other nodes.

- No completion guarantee

Deletion of an LSP provides no guarantees of completion. In particular, if RSVP packets are lost due to a node or link failure it is possible for an LSP to be only partially deleted. To mitigate this, RSVP could maintain soft state reservations and hence eventually remove remaining reservations due to refresh timeouts. This approach is not feasible in non-packet transport networks however, where control and data channels are often separated and hence soft state reservations are not useful.

Finally, one could argue that graceful LSP deletion [RFC3473] would provide guarantee of completion. While this is true for most cases, many implementations will time out graceful deletion if LSP is not removed within certain amount of time, e.g. due to a transit node fault. After that, deletion procedures which provide no completion guarantees will be attempted. Hence, in corner cases a completion guarantee cannot be provided.

o No explicit notification of completion to head-end node

In some cases, it may be useful for a head-end node to know when the data plane has been reconfigured to match working or protecting LSP reservations. This knowledge could be used for initiating operations like enabling alarm monitoring, power equalization and others. Unfortunately, for the reasons mentioned above, make-while-break reversion lacks such explicit notification.

4.3.2. Make-before-break Reversion

This reversion method can be used to overcome limitations of make-while-break reversion. It is similar in spirit to MBB concept used for re-optimization. Instead of relying on deletion of the restoration LSP, the head-end chooses to establish a new reversion LSP that duplicates the configuration of the resources on the working or protecting LSP, and uses identical ASSOCIATION and PROTECTION objects in the Path message of that LSP. Only if setup of this LSP is successful will other (restoration and working or protecting) LSPs be deleted by the head-end. MBB reversion consists of two parts:

A) Make part:

Creating a new reversion LSP following working or protecting LSP’s path. The reversion LSP shares all of the resources of the working or protecting LSP and may share resources with the restoration LSP. As reversion LSP is created, resources are reconfigured to match its reservations. Hence, after reversion LSP is created, data plane configuration reflects working or protecting LSP reservations.

B) Break part:

After "make" part is finished, the original working or protecting and restoration LSPs are torn down, and the reversion LSP becomes the new working or protecting LSP. Removing reservations for working or restoration LSPs does not cause any resource reconfiguration on reversion LSP’s path - nodes follow same procedures as for "break" part of any MBB operation. Hence, after working or protecting and restoration LSPs are removed, data plane configuration is exactly the same as before starting restoration. Thus, reversion is complete.

MBB reversion uses make-before-break characteristics to overcome challenges related to make-while-break reversion as follow:

o Rollback

If "make" part fails, (existing) restoration LSP will still be used to carry existing traffic as the restoration LSP state was not removed. Same logic applies here as for any MBB operation failure.

o Completion guarantee

LSP setup is resilient against RSVP message loss, as Path and Resv messages are refreshed periodically. Hence, given that network recovers from node and link failures eventually, reversion LSP setup is guaranteed to finish with either success or failure.

o Explicit notification of completion to head-end node

Head-end knows that data plane has been reconfigured to match working or protecting LSP reservations on intermediate nodes when it receives Resv for the reversion LSP.

5. Security Considerations

This document reviews procedures defined in [RFC3209] [RFC4872] [RFC4873] and [RFC6689] and does not define any new procedure. This document does not introduce any new security issues other than those already covered in [RFC3209] [RFC4872] [RFC4873] and [RFC6689].

6. IANA Considerations

This informational document does not make any request for IANA action.

7. References

7.1. Normative 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, December 2001.

[RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, January 2003.

[RFC4872] Lang, J., Ed., Rekhter, Y., Ed., and D. Papadimitriou, Ed., "RSVP-TE Extensions in Support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery", RFC 4872, May 2007.

[RFC4873] Berger, L., Bryskin, I., Papadimitriou, D., and A. Farrel, "GMPLS Segment Recovery", RFC 4873, May 2007.

[RFC6689] L. Berger, "Usage of the RSVP ASSOCIATION Object", RFC 6689, July 2012.

7.2. Informative References

[RFC3945] Mannie, E., "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", RFC 3945, October 2004.

[RFC4203] Kompella, K., and Rekhter, Y., "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4203, October 2005.

[RFC4426] Lang, J., Rajagopalan, B., and Papadimitriou, D., "Generalized Multiprotocol Label Switching (GMPLS) Recovery Functional Specification", RFC 4426, March 2006.

[RFC4427] Mannie, E., and Papadimitriou, D., "Recovery (Protection and Restoration) Terminology for Generalized Multi-Protocol Label Switching", RFC 4427, March 2006.

Acknowledgements

The authors would like to thank George Swallow for the discussions on the GMPLS restoration. The authors would like to thank Lou Berger for the guidance on this work. The authors would also like to thank Lou Berger, Vishnu Pavan Beeram and Christian Hopps for reviewing this document and providing valuable comments. A special thanks to Dale Worley for his thorough review of this document.

Contributors

Gabriele Maria Galimberti
Cisco Systems, Inc.

EMail: ggalimbe@cisco.com

Authors' Addresses

Xian Zhang
Huawei Technologies
F3-1-B R&D Center, Huawei Base
Bantian, Longgang District
Shenzhen 518129 P.R.China

EMail: zhang.xian@huawei.com

Haomian Zheng (editor)
Huawei Technologies
F3-1-B R&D Center, Huawei Base
Bantian, Longgang District
Shenzhen 518129 P.R.China

EMail: zhenghaomian@huawei.com

Rakesh Gandhi (editor)
Cisco Systems, Inc.

EMail: rgandhi@cisco.com

Zafar Ali
Cisco Systems, Inc.

EMail: zali@cisco.com

Pawel Brzozowski
ADVA Optical

EMail: PBrzozowski@advaoptical.com

TEAS Working Group
Internet Draft
Intended status: Standard Track
Updates RFC4874
Expires: September 03, 2018

Zafar Ali, Ed.
George Swallow, Ed.
Cisco Systems
F. Zhang, Ed.
Huawei
D. Beller, Ed.
Nokia

March 02, 2018

Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Path Diversity using Exclude Route

draft-ietf-teas-lsp-diversity-10.txt

Status of this Memo

Copyright Notice

Abstract

Resource ReSerVation Protocol-Traffic Engineering provides support for the communication of exclusion information during label switched path (LSP) setup. A typical LSP diversity use case is for protection, where two LSPs should follow different paths through the network in order to avoid single points of failure, thus greatly improving service availability. This document specifies an approach which can be used for network scenarios where full knowledge of the path(s) is not necessarily known by use of an abstract identifier for the path. Three types of abstract identifiers are specified: client-based, Path Computation Engine (PCE)-based, network-based. This document specifies two new diversity subobjects for the RSVP eXclude Route Object (XRO) and the Explicit Exclusion Route Subobject (EXRS).

For the protection use case, LSPs are typically created at a slow rate and exist for a long time, so that it is reasonable to assume that a given (reference) path currently existing, with a well-known identifier, will continue to exist and can be used as a reference when creating the new diverse path. Re-routing of the existing (reference)LSP, before the new path is established, is not considered.

Conventions used in this document

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

Table of Contents

Terms and Abbreviations

Diverse LSP: a diverse Label-Switched Path (LSP) is an LSP that has a path that does not have any link or SRLG in common with the path of a given LSP. Diverse LSPs are meaningful in the context of protection or restoration.

ERO: Explicit Route Object as defined in [RFC3209]

EXRS: Explicit eXclusion Route Subobject as defined in [RFC4874]

SRLG: Shared Risk Link Group as defined in [RFC4202]

Reference Path: the reference path is the path of an existing LSP, to which the path of a diverse LSP shall be diverse.

XRO: eXclude Route Object as defined in [RFC4874]

1. Introduction

Path diversity for multiple connections is a well-known operational requirement. Diversity constraints ensure that Label-Switched Paths (LSPs) can be established without sharing network resources, thus greatly reducing the probability of simultaneous connection failures.

The source node can compute diverse paths for LSPs when it has full knowledge of the network topology and is permitted to signal an Explicit Route Object (ERO). However, there are scenarios where different nodes perform path computations, and therefore there is a need for relevant diversity constraints to be signaled to those nodes. These include (but are not limited to):

- LSPs with loose hops in the Explicit Route Object, e.g. inter-domain LSPs.

- Generalized Multi-Protocol Label Switching (GMPLS) User-Network Interface (UNI), where the core node may perform path computation [RFC4208].

[RFC4874] introduced a means of specifying nodes and resources to be excluded from a route, using the eXclude Route Object (XRO) and Explicit Exclusion Route Subobject (EXRS). It facilitates the calculation of diverse paths for LSPs based on known properties of those paths including addresses of links and nodes traversed, and Shared Risk Link Groups (SRLGs) of traversed links. Employing these mechanisms requires that the source node that initiates signaling knows the relevant properties of the path(s) from which diversity is desired. However, there are circumstances under which this may not be possible or desirable, including (but not limited to):

. Exclusion of a path which does not originate, terminate or traverse the source node of the diverse LSP, in which case the addresses of links and SRLGs of the path from which diversity is required are unknown to the source node.

. Exclusion of a path which is known to the source node of the diverse LSP for which the node has incomplete or no path information, e.g. due to operator policy. In this case, the source node is aware of the existence of the reference path but the information required to construct an XRO object to guarantee diversity from the reference path is not fully known. Inter-domain and GMPLS overlay networks can impose such restrictions.

This is illustrated in the Figure 1, where the overlay reference model from [RFC4208] is shown.

```
   Overlay                                                  Overlay
   Network       +----------------------------------+       Network
 +---------+     |                                  |     +---------+
 |  +----+ |     |  +-----+    +-----+    +-----+   |     | +----+  |
 |  |    | | UNI |  |     |    |     |    |     |   | UNI | |    |  |
 | -+ EN1+-+-----+--+ CN1 +----+ CN2 +----+ CN3 +---+-----+-+ EN3+- |
 |  |    | |  +--+--+     |    |     |    |     |   | +---+-|    |  |
 |  +----+ |  |  |  +--+--+    +--+--+    +--+--+   | |   | +----+  |
 +---------+  |  |     |          |          |      | |   +---------+
              |  |     |          |          |      | |
 +---------+  |  |  +--+--+       |       +--+--+   | |   +---------+
 |  +----+ |  |  |  |     |       +-------+     +-----+   | +----+  |
 |  |    +-+--+  |  | CN4 +---------------+ CN5 |   |     | |    |  |
 | -+ EN2+-+-----+--+     |               |     +---+-----+-+ EN4+- |
 |  |    | | UNI |  +-----+               +-----+   | UNI | |    |  |
 |  +----+ |     |                                  |     | +----+  |
 +---------+     +----------------------------------+     +---------+
   Overlay                 Core Network                     Overlay
   Network                                                  Network

                     Legend:   EN  -  Edge Node
                               CN  -  Core Node
```

Figure 1: Overlay Reference Model [RFC4208]

Figure 1 depicts two types of UNI connectivity: single-homed and dual-homed ENs (which also applies to higher order multi-homed connectivity). Single-homed EN devices are connected to a single CN device via a single UNI link. This single UNI link may constitute a single point of failure. UNI connection between EN1 and CN1 is an example of singled-homed UNI connectivity.

Such a single point of failure can be avoided when the EN device is connected to two different CN devices, as depicted for EN2 in Figure 1. For the dual-homing case, it is possible to establish two different UNI connections from the same source EN device to the same destination EN device. For example, two connections from EN2 to EN3 may use the two UNI links EN2-CN1 and EN2-CN4. To avoid single points of failure within the provider network, it is necessary to also ensure path (LSP) diversity within the core network.

In a network providing a set of UNI interfaces between ENs and CNs such as that shown in Figure 1, the CNs typically perform path computation. Information sharing across the UNI boundary is restricted based on the policy rules imposed by the core network. Typically, the core network topology information as well as LSP path information is not exposed to the ENs. In the network shown in Figure 1, consider a use case where an LSP from EN2 to EN4 needs to be SRLG diverse from an LSP from EN1 to EN3. In this case, EN2 may not know SRLG attributes of the EN1- EN3 LSP and hence cannot construct an XRO to exclude these SRLGs. In this example EN2 cannot use the procedures described in [RFC4874]. Similarly, an LSP from EN2 to EN3 traversing CN1 needs to be diverse from an LSP from EN2 to EN3 going via CN4. Again, in this case, exclusions based on [RFC4874] cannot be used.

This document addresses these diversity requirements by introducing an approach of excluding the path taken by these particular LSP(s). The reference LSP(s) or route(s) from which diversity is required is/are identified by an abstract "identifier". The type of identifier to use is highly dependent on the core network operator’s networking deployment scenario; it could be client-initiated (provided by the EN), provided by a PCE or allocated by the (core) network. This document defines three different types of identifiers corresponding to these three cases: a client-initiated identifier, a PCE allocated identifier and CN ingress node (UNI-N) allocated identifier (= network-assigned identifier).

1.1. Client-Initiated Identifier

The following fields MUST be used to represent the client-initiated identifier: IPv4/IPv6 tunnel sender address, IPv4/IPv6 tunnel endpoint address, Tunnel ID, and Extended Tunnel ID. Based on local policy, the client MAY also include the LSP ID to identify a specific LSP within the tunnel. These fields are defined in [RFC3209], sections 4.6.1.1 and 4.6.2.1.

The usage of the client-initiated identifier is illustrated by Figure 1. Suppose a LSP from EN2 to EN4 needs to be diverse with respect to a LSP from EN1 to EN3. The LSP identifier of the EN1-EN3 LSP is LSP-IDENTIFIER1, where LSP-IDENTIFIER1 is defined by the tuple (tunnel-id = T1, LSP ID = L1, source address = EN1.RID (ROUTE Identifier), destination address = EN3.RID, extended tunnel-id = EN1.RID). Similarly, LSP identifier of the EN2-EN4 LSP is LSP-IDENTIFIER2, where LSP-IDENTIFIER2 is defined by the tuple (tunnel-id = T2, LSP ID = L2, source address = EN2.RID, destination address = EN4.RID, extended tunnel-id = EN2.RID). The

EN1-EN3 LSP is signaled with an exclusion requirement from LSP-IDENTIFIER2, and the EN2-EN4 LSP is signaled with an exclusion requirement from LSP-IDENTIFIER1. In order to maintain diversity between these two connections within the core network, the core network SHOULD implement Crankback Signaling Extensions as defined in [RFC4920]. Note that crankback signaling is known to lead to slower setup times and sub-optimal paths under some circumstances as described by [RFC4920].

1.2. PCE-allocated Identifier

In scenarios where a PCE is deployed and used to perform path computation, the core edge node (e.g., node CN1 in Figure 1) could consult a PCE to allocate identifiers, which are used to signal path diversity constraints. In other deployment scenarios, a PCE is deployed at a network node(s) or a PCE is part of a Network Management System (NMS). In all these cases, the PCE is consulted and the Path-Key as defined in [RFC5520] can be used in RSVP signaling as the identifier to ensure diversity.

An example of specifying LSP diversity using a Path-Key is shown in Figure 2, where a simple network with two domains is shown. It is desired to set up a pair of path-disjoint LSPs from the source in Domain 1 to the destination in Domain 2, but the domains keep strict confidentiality about all path and topology information.

The first LSP is signaled by the source with ERO {A, B, loose Dst} and is set up with the path {Src, A, B, U, V, W, Dst}. However, when sending the Record Route Object (RRO) out of Domain 2, node U would normally strip the path and replace it with a loose hop to the destination. With this limited information, the source is unable to include enough detail in the ERO of the second LSP to avoid it taking, for example, the path {Src, C, D, X, V, W, Dst} for path-disjointness.

```
       ---------------------    -----------------------------
      | Domain 1            |  |                    Domain 2 |
      |                     |  |                             |
      |        ---    ---   |  |   ---    ---     ---        |
      |       | A |--| B |--+--+--| U |--| V |---| W |       |
      |      / ---    ---   |  |   ---    ---     --- \      |
      |  ---/               |  |          /       /    \---  |
      | |Src|               |  |         /       /     |Dst| |
      |  ---\               |  |        /       /      /---  |
      |      \ ---    ---   |  |   --- /   --- /  --- /      |
      |       | C |--| D |--+--+--| X |---| Y |--| Z |       |
      |        ---    ---   |  |   ---     ---    ---        |
      |                     |  |                             |
       ---------------------    -----------------------------
```

Figure 2: A Simple Multi-Domain Network

In order to support LSP diversity, node U consults the PCE and replaces the path segment {U, V, W} in the RRO with a Path Key subobject. The PCE function assigns an "identifier" and puts it into the Path Key field of the Path Key subobject. The PCE ID in the message indicates that this replacement operation was performed by node U.

With this additional information, the source node is able to signal the subsequent LSPs with the ERO set to {C, D, exclude Path Key(EXRS), loose Dst}. When the signaling message reaches node X, it can consult the PCE function associated with node U to expand the Path Key in order to calculate a path that is diverse with respect to the first LSP. Alternatively, the source node could use an ERO of {C, D, loose Dst} and include an XRO containing the Path Key.

This mechanism can work with all the Path Key resolution mechanisms, as detailed in [RFC5553] section 3.1. A PCE, co-located or not, may be used to resolve the Path Key, but the node (i.e., a Label Switching Router (LSR)) can also use the Path Key information to index a Path Segment previously supplied to it by the entity that originated the Path Key, for example the LSR that inserted the Path Key in the RRO or a management system.

1.3. Network-Assigned Identifier

There are scenarios in which the network provides diversity-related information for a service that allows the client device to include this information in the signaling message. If the

Shared Resource Link Group (SRLG) identifier information is both available and shareable (by policy) with the ENs, the procedure defined in [RFC8001] can be used to collect SRLG identifiers associated with an LSP (LSP1). When a second LSP (LSP2) needs to be diverse with respect to LSP1, the EN constructing the RSVP signaling message for setting up LSP2 can insert the SRLG identifiers associated with LSP1 as diversity constraints into the XRO using the procedure described in [RFC4874]. However, if the core network SRLG identifiers are either not available or not shareable with the ENs based on policies enforced by core network, existing mechanisms cannot be used.

In this draft, a signaling mechanism is defined where information signaled to the CN via the UNI does not require shared knowledge of core network SRLG information. For this purpose, the concept of a Path Affinity Set (PAS) is defined for abstracting SRLG information. The motive behind the introduction of the PAS is to minimize the exchange of diversity information between the core network (CNs) and the client devices (ENs). The PAS contains an abstract SRLG identifier associated with a given path rather than a detailed SRLG list. The PAS is a single identifier that can be used to request diversity and associate diversity. The means by which the processing node determines the path corresponding to the PAS is beyond the scope of this document.

A CN on the core network boundary interprets the specific PAS identifier (e.g. "123") as meaning to exclude the core network SRLG information (or equivalent) that has been allocated by LSPs associated with this PAS identifier value. For example, if a Path exists for the LSP with the PAS identifier "123", the CN would use local knowledge of the core network SRLGs associated with the LSPs tagged with PAS attribute "123" and use those SRLGs as constraints for path computation. If a PAS identifier is used as an exclusion identifier in the connection request, the CN (UNI-N) in the core network is assumed to be able to determine the existing core network SRLG information and calculate a path that meets the determined diversity constraints.

When a CN satisfies a connection setup for a (SRLG) diverse signaled path, the CN may optionally record the core network SRLG information for that connection in terms of CN based parameters and associates that with the EN addresses in the Path message. Specifically, for Layer 1 Virtual Private Networks (L1VPNs), Port Information Tables (PIT) [RFC5251] can be leveraged to translate between client (EN) addresses and core network addresses.

The means to distribute the PAS information within the core network is beyond the scope of this document. For example, the PAS and the associated SRLG information can be distributed within the core network by an Interior Gateway Protocol (IGP) or by other means such as configuration. Regardless of means used to distribute the PAS information, the information is kept inside the core network and is not shared with the overlay network (see Figure 1).

2. RSVP-TE signaling extensions

This section describes the signaling extensions required to address the aforementioned requirements and use cases.

2.1. Diversity XRO Subobject

New Diversity XRO subobjects are defined below for the IPv4 and IPv6 address families. Most of the fields in the IPv4 and IPv6 Diversity XRO subobjects are common and are described following the definition of the two subobjects.

IPv4 Diversity XRO subobject is defined 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|L|  XRO Type   |     Length    |DI Type|A-Flags|E-Flags| Resvd |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           IPv4 Diversity Identifier Source Address            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Diversity Identifier Value                   |
//                             ...                             //
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Similarly, the IPv6 Diversity XRO subobject is defined 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|L|  XRO Type   |     Length    |DI Type|A-Flags|E-Flags| Resvd |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           IPv6 Diversity Identifier source address            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         IPv6 Diversity Identifier source address (cont.)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         IPv6 Diversity Identifier source address (cont.)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         IPv6 Diversity Identifier source address (cont.)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Diversity Identifier Value                   |
//                             ...                             //
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

L:

The L-flag is used in the same way as for the XRO subobjects defined in [RFC4874], i.e.,

0 indicates that the diversity constraints MUST be satisfied.

1 indicates that the diversity constraints SHOULD be satisfied.

XRO Type

The value is set to TBA1 for the IPv4 Diversity XRO subobject (value to be assigned by IANA). The value is set to TBA2 for the IPv6 Diversity XRO subobject (value to be assigned by IANA).

Length

Per [RFC4874], the Length contains the total length of the IPv4/IPv6 subobject in bytes, including the XRO Type and Length fields. The Length is variable, depending on the diversity identifier value.

Diversity Identifier Type (DI Type)

Diversity Identifier Type (DI Type) indicates the way the reference LSP(s) or route(s) with which diversity is required is identified in the IPv4/IPv6 Diversity subobjects. The following three DI type values are defined in this document:

DI Type value	Definition
1	Client Initiated Identifier
2	PCE Allocated Identifier
3	Network Assigned Identifier

Attribute Flags (A-Flags):

The Attribute Flags (A-Flags) are used to communicate desirable attributes of the LSP being signaled in the IPv4/IPv6 Diversity subobjects. Each flag acts independently. Any combination of flags is permitted.

0x01 = Destination node exception

Indicates that the exclusion does not apply to the destination node of the LSP being signaled.

0x02 = Processing node exception

Indicates that the exclusion does not apply to the node(s) performing ERO expansion for the LSP being signaled. An ingress UNI-N node is an example of such a node.

0x04 = Penultimate node exception

Indicates that the penultimate node of the LSP being signaled MAY be shared with the excluded path even when this violates the exclusion flags. This flag is useful, for example, when an EN is not dual-homed (like EN4 in Figure 1 where all LSPs have to go through CN5).

The penultimate node exception flag is typically set when the destination node is single homed (e.g. EN1 or EN4 in Figure 1). In such a case, LSP diversity can only be accomplished inside the core network up to the egress node and the penultimate hop must be the same for the LSPs.

0x08 = LSP ID to be ignored

This flag is used to indicate tunnel level exclusion. Specifically, this flag is used to indicate that if diversity identifier contains LSP ID field, the LSP ID is to be ignored and the exclusion applies to any LSP matching the rest of the diversity identifier.

Exclusion Flags (E-Flags):

The Exclusion Flags are used to communicate the desired type(s) of exclusion requested in the IPv4/IPv6 diversity subobjects. The following flags are defined. Any combination of these flags is permitted. Please note that the exclusion specified by these flags may be modified by the value of the Attribute-flags. For example, node exclusion flag is ignored for the "Penultimate node" if the "Penultimate node exception" flag of the Attribute-flags is set.

0x01 = SRLG exclusion

Indicates that the path of the LSP being signaled is requested to be SRLG disjoint with respect to the excluded path specified by the IPv4/IPv6 Diversity XRO subobject.

0x02 = Node exclusion

Indicates that the path of the LSP being signaled is requested to be node-diverse from the excluded path specified by the IPv4/IPv6 Diversity XRO subobject.

0x04 = Link exclusion

Indicates that the path of the LSP being signaled is requested to be link-diverse from the path specified by the IPv4/IPv6 Diversity XRO subobject.

0x08 = reserved

This flag is reserved. It MUST be set to zero on transmission, and MUST be ignored on receipt for both IPv4/IPv6 Diversity XRO subobjects.

Resvd

This field is reserved. It MUST be set to zero on transmission, and MUST be ignored on receipt for both IPv4/IPv6 Diversity XRO subobjects.

IPv4 / IPv6 Diversity Identifier source address:

This field MUST be set to the IPv4/IPv6 address of the node that assigns the diversity identifier. Depending on the diversity identifier type, the diversity identifier source may be a client node, PCE entity or network node. Specifically:

- o When the diversity identifier type is set to "IPv4/IPv6 Client Initiated Identifier", the value MUST be set to IPv4/IPv6 tunnel sender address of the reference LSP against which diversity is desired. IPv4/IPv6 tunnel sender address is as defined in [RFC3209].

- o When the diversity identifier type is set to "IPv4/IPv6 PCE Allocated Identifier", the value MUST be set to the IPv4/IPv6 address of the node that assigned the Path Key identifier and that can return an expansion of the Path Key or use the Path Key as exclusion in a path computation. The Path Key is defined in [RFC5553]. The PCE-ID is carried in the Diversity Identifier Source Address field of the subobject.

- o When the diversity identifier type is set to "IPv4/IPv6 Network Assigned Identifier", the value MUST be set to the

IPv4/IPv6 address of the node allocating the Path Affinity Set (PAS).

Diversity Identifier Value:

Encoding for this field depends on the diversity identifier type, as defined in the following.

When the diversity identifier type is set to "Client Initiated Identifier" in the IPv4 Diversity XRO subobject, the diversity identifier value MUST be encoded 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 IPv4 tunnel end point address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Must Be Zero         |     Tunnel ID                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Extended Tunnel ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Must Be Zero         |            LSP ID             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The IPv4 tunnel end point address, Tunnel ID, Extended Tunnel ID and LSP ID are as defined in [RFC3209].

When the diversity identifier type is set to "Client Initiated Identifier" in the IPv6 Diversity XRO subobject, the diversity identifier value MUST be encoded 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 IPv6 tunnel end point address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             IPv6 tunnel end point address (cont.)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             IPv6 tunnel end point address (cont.)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             IPv6 tunnel end point address (cont.)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Must Be Zero         |     Tunnel ID                 |
```

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Extended Tunnel ID                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Extended Tunnel ID (cont.)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Extended Tunnel ID (cont.)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Extended Tunnel ID (cont.)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Must Be Zero         |             LSP ID            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The IPv6 tunnel end point address, Tunnel ID, IPv6 Extended Tunnel ID and LSP ID are as defined in [RFC3209].

When the diversity identifier type is set to "PCE Allocated Identifier" in IPv4 or IPv6 Diversity XRO subobject, the diversity identifier value MUST be encoded 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Must Be Zero          |            Path Key           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Path Key is defined in [RFC5553].

When the diversity identifier type is set to "Network Assigned Identifier" in IPv4 or IPv6 Diversity XRO subobject, the diversity identifier value MUST be encoded 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Path Affinity Set (PAS) identifier                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Path Affinity Set (PAS) identifier field is a 32-bit value that is scoped by, i.e., is only meaningful when used in combination with, the Diversity Identifier source address field. There are no restrictions on how a node selects a PAS identifier value. Section 1.3 defines the PAS term and provides context on how values may be selected.

2.2. Diversity EXRS Subobject

[RFC4874] defines the EXRS ERO subobject. An EXRS is used to identify abstract nodes or resources that must not or should not be used on the path between two inclusive abstract nodes or resources in the explicit route. An EXRS contains one or more subobjects of its own, called EXRS subobjects [RFC4874].

An EXRS MAY include a Diversity subobject as specified in this document. The same type values TBA1 and TBA2 MUST be used.

2.3. Processing rules for the Diversity XRO and EXRS subobjects

The procedure defined in [RFC4874] for processing the XRO and EXRS is not changed by this document. The processing rules for the Diversity XRO and EXRS subobjects are similar unless the differences are explicitly described. Similarly, IPv4 and IPv6 Diversity XRO subobjects and IPv4 and IPv6 Diversity EXRS subobjects follow the same processing rules.

If the processing node cannot recognize the Diversity XRO/EXRS subobject, the node is expected to follow the procedure defined in [RFC4874].

An XRO/EXRS object MAY contain multiple Diversity subobjects of the same DI Type. E.g., in order to exclude multiple Path Keys, a node MAY include multiple Diversity XRO subobjects each with a different Path Key. Similarly, in order to exclude the routes taken by multiple LSPs, a node MAY include multiple Diversity XRO/EXRS subobjects each with a different LSP identifier. Likewise, to exclude multiple PAS identifiers, a node MAY include multiple Diversity XRO/EXRS subobjects each with a different PAS identifier. However, all Diversity subobjects in an XRO/EXRS MUST contain the same Diversity Identifier Type. If a Path message contains an XRO/EXRS with multiple Diversity subobjects of different DI Types, the processing node MUST return a PathErr with the error code "Routing Problem" (24) and error sub-code "XRO/EXRS Too Complex" (68/69).

If the processing node recognizes the Diversity XRO/EXRS subobject but does not support the DI type, it MUST return a

PathErr with the error code "Routing Problem" (24) and error sub-code "Unsupported Diversity Identifier Type" (TBA3).

In case of DI type "Client Initiated Identifier", all nodes along the path SHOULD process the diversity information signaled in the XRO/EXRS Diversity subobjects to verify that the signaled diversity constraint is satisfied. If a diversity violation is detected, crankback signaling MAY be initiated.

In case of DI type "PCE Allocated Identifier" and "Network Assigned Identifier", the nodes in the domain that perform path computation SHOULD process the diversity information signaled in the XRO/EXRS Diversity subobjects as follows. In the PCE case, the ingress node of a domain sends a path computation request for a path from ingress node to egress node including diversity constraints to a PCE. Or,in the PAS case, the ingress node is capable to calculate the path for the new LSP from ingress node to the egress node taking the diversity constraints into account. The calculated path is then carried in the explicit route object (ERO). Hence, the transit nodes in a domain and the domain egress node SHOULD NOT process the signaled diversity information unless path computation is performed.

While processing EXRS object, if a loose hop expansion results in the creation of another loose hop in the outgoing ERO, the processing node MAY include the EXRS in the newly created loose hop for further processing by downstream nodes.

The Attribute-flags affect the processing of the Diversity XRO/EXRS subobject as follows:

- o When the "Processing node exception" flag is set, the exclusion MUST be ignored for the node processing the XRO or EXRS subobject.

- o When the "Destination node exception" flag is set, the exclusion MUST be ignored for the destination node in processing the XRO subobject. The destination node exception for the EXRS subobject applies to the explicit node identified by the ERO subobject that identifies the next abstract node. When the "destination node exception" flag is set in the EXRS subobject, exclusion MUST be ignored for the said node (i.e., the next abstract node).

- o When the "Penultimate node exception" flag is set in the XRO subobject, the exclusion MUST be ignored for the penultimate node on the path of the LSP being established.

The penultimate node exception for the EXRS subobject applies to the node before the explicit node identified by the ERO subobject that identifies the next abstract node. When the "penultimate node exception" flag is set in the EXRS subobject, the exclusion MUST be ignored for the said node (i.e., the node before the next abstract node).

If the L-flag of the Diversity XRO subobject or Diversity EXRS subobject is not set, the processing node proceeds as follows.

- If the Diversity Identifier Type is set to "Client Initiated Identifier", the processing node MUST ensure that the path calculated/expanded for the signaled LSP is diverse from the route taken by the LSP identified in the Diversity Identifier Value field.

- If the Diversity Identifier Type is set to "PCE Allocated Identifier", the processing node MUST ensure that any path calculated for the signaled LSP is diverse from the route identified by the Path Key. The processing node MAY use the PCE identified by the Diversity Identifier Source Address in the subobject for route computation. The processing node MAY use the Path Key resolution mechanisms described in [RFC5553].

- If the Diversity Identifier Type is set to "Network Assigned Identifier", the processing node MUST ensure that the path calculated for the signaled LSP is diverse with respect to the values associated with the PAS identifier and Diversity Identifier source address fields.

- Regardless of whether the path computation is performed locally or at a remote node (e.g., PCE), the processing node MUST ensure that any path calculated for the signaled LSP is diverse from the requested Exclusion Flags.

- If the excluded path referenced in the XRO subobject is unknown to the processing node, the processing node SHOULD ignore the Diversity XRO subobject and SHOULD proceed with the signaling request. After sending the Resv for the signaled LSP, the processing node MUST return a PathErr with the error code "Notify Error" (25) and error sub-code TBA4 "Route of XRO LSP identifier unknown" (value to be assigned by IANA) for the signaled LSP.

- If the processing node fails to find a path that meets the requested constraint, the processing node MUST return a PathErr with the error code "Routing Problem" (24) and error sub-code "Route blocked by Exclude Route" (67).

If the L-flag of the Diversity XRO subobject or Diversity EXRS subobject is set, the processing node proceeds as follows:

- If the Diversity Identifier Type is set to " Client Initiated Identifiers", the processing node SHOULD ensure that the path calculated/ expended for the signaled LSP is diverse from the route taken by the LSP identified in the Diversity Identifier Value field.

- If the Diversity Identifier Type is set to " PCE Allocated Identifiers", the processing node SHOULD ensure that the path calculated for the signaled LSP is diverse from the route identified by the Path Key.

- If the Diversity Identifier Type is set to "IPv4/IPv6 Network Assigned Identifiers", the processing node SHOULD ensure that the path calculated for the signaled LSP is diverse with respect to the values associated with the PAS identifier and Diversity Identifier source address fields.

- If the processing node fails to find a path that meets the requested constraint, it SHOULD proceed with signaling using a suitable path that meets the constraint as far as possible. After sending the Resv for the signaled LSP, it MUST return a PathErr message with error code "Notify Error" (25) and error sub-code TBA5 "Failed to satisfy Exclude Route" (value: to be assigned by IANA) to the source node.

If, subsequent to the initial signaling of a diverse LSP, an excluded path referenced in the XRO subobject becomes known to the processing node, or a change in the excluded path becomes known to the processing node, the processing node MUST re-evaluate the exclusion and diversity constraints requested by the diverse LSP to determine whether they are still satisfied.

- In case the L-flag was not set in the initial setup message, the exclusion and diversity constraints were satisfied at the time of the initial setup. If the processing node re-evaluating the exclusion and diversity constraints for a diverse LSP detects that the exclusion and diversity constraints are no longer met, it MUST send a PathErr message for the diverse LSP with the error code "Routing Problem" (24) and error sub-code "Route blocked by Exclude Route" (67). The Path_State_Removed flag (PSR) [RFC3473] MUST NOT be set. A source node receiving a

PathErr message with this error code and sub-code combination SHOULD take appropriate actions and move the diverse LSP to a new path that meets the original constraints.

- In case the L-flag was set in the initial setup message, the exclusion and diversity constraints may or may not be satisfied at any given time. If the exclusion constraints for a diverse LSP were satisfied before and if the processing node re-evaluating the exclusion and diversity constraints for a diverse LSP detects that exclusion and diversity constraints are no longer met, it MUST send a PathErr message for the diverse LSP with the error code error code "Notify Error" (25) and error sub-code TBA5 "Failed to satisfy Exclude Route" (value: to be assigned by IANA). The PSR flag MUST NOT be set. The source node MAY take no consequent action and keep the LSP along the path that does not meet the original constraints. Similarly, if the exclusion constraints for a diverse LSP were not satisfied before and if the processing node re-evaluating the exclusion and diversity constraints for a diverse LSP detects that the exclusion constraints are met, it MUST send a PathErr message for the diverse LSP with the error code "Notify Error" (25) and a new error sub- code TBA6 "Compliant path exists" (value: to be assigned by IANA). The PSR flag MUST NOT be set. A source node receiving a PathErr message with this error code and sub-code combination MAY move the diverse LSP to a new path that meets the original constraints.

3. Security Considerations

This document does not introduce any additional security issues in addition to those identified in [RFC5920], [RFC2205], [RFC3209], [RFC3473], [RFC2747], [RFC4874], [RFC5520], and [RFC5553].

The diversity mechanisms defined in this document, rely on the new diversity subobject that is carried in the XRO or EXRS, respectively. In section 7 of [RFC4874], it is noted some administrative boundaries may remove the XRO due to security concerns on explicit route information exchange. However, when the diversity subobjects specified in this document are used, removing at the administrative boundary an XRO containing these diversity subobjects would result in the request for diversity being dropped at the boundary, and path computation would be unlikely to produce the requested diverse path. As such, diversity subobjects MUST be retained in an XRO crossing an administrative boundary, even if other subobjects are removed. This retention would be based on operator policy. The use of diversity subobjects are based on mutual agreement. This avoids the need to share the identity of network resources when supporting diversity.

4. IANA Considerations

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

4.1. New XRO subobject types

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

This document defines two new subobjects for the EXCLUDE_ROUTE object [RFC4874], C-Type 1. (see: http://www.iana.org/assignments/rsvp-parameters/rsvp-parameters.xhtml#rsvp-parameters-94)

Subobject Description	Subobject Type
IPv4 Diversity subobject	TBA1
IPv6 Diversity subobject	TBA2

4.2. New EXRS subobject types

The Diversity XRO subobjects are also defined as new EXRS subobjects. (EXPLICIT_ROUTE see: http://www.iana.org/assignments/rsvp-parameters/rsvp-parameters.xhtml#rsvp-parameters-24). The same numeric subobject type values TBA1 and TBA2 are being requested for the two new EXRS subobjects.

4.3. New RSVP error sub-codes

IANA registry: RSVP PARAMETERS
Subsection: Error Codes and Globally Defined Error Value Sub-Codes.

For Error Code "Routing Problem" (24) (see [RFC3209]) the following sub-codes are defined. (see: http://www.iana.org/assignments/rsvp-parameters/rsvp-parameters.xhtml#rsvp-parameters-105)

Error Value Sub-codes	Description	Reference
TBA3	Unsupported Diversity Identifier Type	This document

For Error Code "Notify Error" (25) (see [RFC3209]) the following sub-codes are defined. (see: http://www.iana.org/assignments/rsvp-parameters/rsvp-parameters.xhtml#rsvp-parameters-105)

Error Value Sub-codes	Description	Reference
TBA4	Route of XRO LSP identifier unknown	This document
TBA5	Failed to satisfy Exclude Route	This document
TBA6	Compliant path exists	This document

5. Acknowledgements

The authors would like to thank Xihua Fu for his contributions. The authors also would like to thank Luyuan Fang and Walid Wakim for their review comments.

6. References

6.1. Normative References

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

[RFC2205] Braden, R. (Ed.), Zhang, L., Berson, S., Herzog, S. and S. Jamin, "Resource ReserVation Protocol -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2747] Baker, F., Lindell, B. and M. Talwar, "RSVP Cryptographic Authentication", RFC 2747, January 2000.

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

[RFC3473] Berger, L., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, January 2003.

[RFC4202] Kompella, Ed., K, Rekhter, Y, Ed., "Routing Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4202, October 2005.

[RFC4874] Lee, CY., Farrel, A., and S. De Cnodder, "Exclude Routes - Extension to Resource ReserVation Protocol-Traffic Engineering (RSVP-TE)", RFC 4874, April 2007.

[RFC4920] Farrel, A., Ed., Satyanarayana, A., Iwata, A., Fujita, N., and G. Ash, "Crankback Signaling Extensions for MPLS and GMPLS RSVP-TE", RFC 4920, July 2007.

[RFC5553] Farrel, A., Ed., Bradford, R., and JP. Vasseur, "Resource Reservation Protocol (RSVP) Extensions for Path Key Support", RFC 5553, May 2009.

6.2. Informative References

[RFC4208] Swallow, G., Drake, J., Ishimatsu, H., and Y. Rekhter, "Generalized Multiprotocol Label Switching (GMPLS) User-Network Interface (UNI): Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Support for the Overlay Model", RFC 4208, October 2005.

[RFC5520] Bradford, R., Ed., Vasseur, JP., and A. Farrel, "Preserving Topology Confidentiality in Inter-Domain Path Computation Using a Path-Key-Based Mechanism", RFC 5520, April 2009.

[RFC8001] F. Zhang, D. Li, O. Gonzalez de Dios, C. Margaria, "RSVP-TE Extensions for Collecting SRLG Information", RFC 8001, January 2017.

[RFC2205] Braden, R. (Ed.), Zhang, L., Berson, S., Herzog, S. and S. Jamin, "Resource ReserVation Protocol -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC5251] Fedyk, D. (Ed.), Rekhter, Y. (Ed.), Papadimitriou, D., Rabbat, R., and Berger, L., "Layer 1 VPN Basic Mode", RFC 5251, July 2008.

[RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, July 2010.

Contributors' Addresses

Igor Bryskin
Huawei Technologies
Email: Igor.Bryskin@huawei.com

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

Dhruv Dhody
Huawei Technologies
Email: dhruv.ietf@gmail.com

Oscar Gonzalez de Dios
Telefonica I+D
Email: ogondio@tid.es

Don Fedyk
Hewlett-Packard Enterprise
Email: don.fedyk@hpe.com

Clarence Filsfils
Cisco Systems, Inc.
Email: cfilsfil@cisco.com

Gabriele Maria Galimberti
Cisco Systems
Email: ggalimbe@cisco.com

Ori Gerstel
SDN Solutions Ltd.
Email: origerstel@gmail.com

Matt Hartley
Cisco Systems
Email: mhartley@cisco.com

Kenji Kumaki
KDDI Corporation
Email: ke-kumaki@kddi.com

Ruediger Kunze
Deutsche Telekom AG
Email: Ruediger.Kunze@telekom.de

Lieven Levrau
Nokia
Email: Lieven.Levrau@nokia.com

Cyril Margaria
cyril.margaria@gmail.com

Julien Meuric
France Telecom Orange
Email: julien.meuric@orange.com

Yuji Tochio
Fujitsu
Email: tochio@jp.fujitsu.com

Xian Zhang
Huawei Technologies
Email: zhang.xian@huawei.com

Authors' Addresses

Zafar Ali
Cisco Systems.
Email: zali@cisco.com

Dieter Beller

Nokia
Email: Dieter.Beller@nokia.com

George Swallow
Cisco Systems
Email: swallow@cisco.com

Fatai Zhang
Huawei Technologies
Email: zhangfatai@huawei.com

TEAS Working Group K. Lam
Internet Draft E. Varma
Intended status: Informational Nokia
Intended status: Informational P. Doolan
Expires: April 2017 Coriant
N. Davis
Ciena
B. Zeuner
Deutsche Telekom
M. Betts
ZTE
I. Busi
Huawei
S. Mansfield
Ericsson
R. Vilalta
CTTC
V. Lopez
Telefonica
October 27, 2016

Usage of IM for network topology to support TE Topology YANG Module Development
draft-lam-teas-usage-info-model-net-topology-04.txt

Status of this Memo

Abstract

The benefits of using a common Information Model (IM) as a foundation for deriving purpose and protocol specific interfaces, particularly

for complex networking domains, has been described in draft-betts-netmod-framework-data-schema-uml. This draft describes existing information model relevant to Network Topology and illustrates how it can be used to help ensure the consistency and completeness of the YANG data modeling for TE topologies solutions work in TEAS.

Table of Contents

1. Introduction

This draft describes existing information modeling (IM) relevant to Network Topology [ONF TR-512] [OSSDN SNOWMASS] and illustrates how it can be used to help ensure the consistency and completeness of the YANG data model (DM) for TE topologies solutions development work in TEAS.

2. Background and Motivation

Information Models (IM) and Data Models (DM) are related but different. An IM provides an abstract, conceptual view of the system being modeled in terms of its constituent parts (objects), independent of any specific implementations or protocols used to transport the data; it hides all protocol and implementation details (RFC 3444, TM Forum/NGCOR, ITU-T SG 15). A DM is a concrete specification in a particular language of an interface to, in this case, a controlled/managed system. The intention of the distinction between IMs and DMs has been to separate the modeling of problem space semantics from the modeling of the implementation of those semantics (though the dividing line has not always been clearly articulated).

A DM may be derived from an IM though it is often created without (explicit or obviously implicit) reference to one. When a DM is derived from an IM, the DM and the components of the system it provides control/management access to are traceable to the definitions provided in the IM. There is no ambiguity between designer, developer, user or operator regarding the name, function, and information elements that are associated with a particular managed object.

As described in [I-D.betts], when DMs are created "in isolation" solely for the purpose of encoding specific interfaces, they may do that job adequately for any particular interface but in complex domains may create opportunities for confusion, duplication of effort, lack of interoperability, and lack of extensibility. In the past, ad-hoc development of DMs has caused significant operational and implementation inefficiencies in our industry.

Since March 2014, upon IESG recommendation that SNMP no longer be used for new work re configuration and that NETCONF/YANG be used instead, there has been an explosion of YANG DM development in IETF. It has consequently been recognized as essential to assure proper coordination of YANG DM development (including reaching out to different SDOs/consortia), as well as to assure that the YANG modules themselves provide a good representation of what is being modeled, to meet expectations of functionality, quality, and interoperability. In order to facilitate this objective, guidance from available pertinent IMs can be valuable.

This draft first describes an existing information model relevant to Network Topology [ONF TR-512], which is part of the Common Information Model (ONF-CIM) of network resources (as described in [I-D.betts]), that can be leveraged to assess the consistency and completeness of related YANG modules under development. It also describes an transport application-specific IM [OSSDN SNOWMASS], derived from CIM pruning and refactoring as explained in [I-D.betts], that is intended to enable further clarity in understanding the modeling. Being part of a Common Information Model, it will not lead to development of incompatible/uncoordinated models that can be difficult to maintain as other purpose-specific interfaces are developed.

3. The Common Information Model

This section provides a high level introduction to the ONF Common Information Model (ONF-CIM), and in particular its Core Model (see [ONF TR-512]), to provide an overall context for the topology relevant subset. The ONF-CIM has been developed through collaboration among several SDOs, including ITU-T, TM Forum, and ONF, and also published as ITU-T Recommendation G.7711 [G.7711].

An information model describes the things in a domain in terms of objects, their properties (represented as attributes), and their relationships.

The ONF-CIM is expressed in a formal language called UML (Unified Modeling Language). UML has a number of basic model elements, called UML artifacts. In order to assure a consistent and harmonized modeling approach, only a selected subset of these UML artifacts were used in the development of the ONF-CIM according to guidelines for creating an information model expressed in UML (see the UML Guidelines document in the ONF TR-514 [ONF TR-514]).

The ONF-CIM has been developed using the Papyrus open source UML Tool, for which a detailed guidelines document is available (see the Papyrus Guidelines document in the ONF TR-515 [ONF TR-515]). This guidelines document also describes how the modelers constructing the ONF-CIM can cooperate in the GitHub environment to allow for separate and still coordinated development of the ONF-CIM fragments.

The OMF-CIM includes all of the artifacts (objects, attributes, associations, etc.) that are necessary to describe the domain for the applications being developed.

It will be necessary to continually expand and refine the ONF-CIM over time as, for example to add, new applications, capabilities or forwarding technologies, or to refine the ONF-CIM as new insights are gained. To allow these extensions to be made in a seamless manner, the ONF-CIM is structured into a number of sub-models. This modeling approach enables application specific and forwarding technology specific extensions to be developed by domain experts with appropriate independence. This approach is further articulated in ONF TR-513 [ONF TR-513] and [I-D.betts].

3.1. Core Model

The Core Model of the ONF-CIM consists of model artifacts that are intended for use by multiple applications and/or forwarding technologies.

For navigability, the Core Model is further sub-structured into sub-models. Currently, these consist of the Core Network Model (CNM), Core Foundation Model, Core Physical Model, and the Core Specification Model. The following sub-sections provide an overview of these sub-models. A detailed description is contained in ONF TR-512 [ONF TR-512].

3.1.1. Core Network Model

The Core Network Model (CNM) consists of artifacts that model the essential network aspects that are neutral to the forwarding technology of the network. The CNM currently encompasses Topology, Termination, and Forwarding aspects (subsets of the CNM) as described below:

- Topology Subset of CNM

  The Topology subset of the CNM supports the modeling of network topology information, which can be used to build the topology database and depict the topology. Object classes representing topological entities include:

  o Forwarding Domain (FD): Offers the potential to enable forwarding of information.

  o Link (L): Models the adjacency between two or more FDs. A Link has LinkPorts.

  o Logical Termination Point (LTP): Models the ports of a link. It encapsulates the termination, adaptation, and OAM functions of one or more transport layers.

  o Network Element (NE): While not actually part of topology, a NE brings meaning to the FD and the LTP contexts (and hence the links). A NE represents physical equipment "bundling" to provide a view of management scope, management access, and session.

The Topology subset of the CNM supports network topology abstraction and virtualization. FD abstraction is supported via recursive aggregation and virtualization via partitioning of resources according to the resource dedication criterion.

- Forwarding Subset of CNM

  The Forwarding subset of the CNM (not covered in detail in this draft) supports configuration of forwarding entities, including their setup, modification, and tear down. Artifacts representing the forwarding construct include:

  o ForwardingConstruct (FC): Also known as SNC. In conjunction with the FcPort, FC models the enabled forwarding between two FcPorts across a FD.

  o FcPort: Models the access to the FC, and associates the FC to the LTP. When the FC supports protection, the FcPort also indicates its role in the protection scheme, i.e., whether it is a working or protection FcPort.

  o FcRoute: Also known as SncRoute. It models the individual routes of an FC.

  o FcSwitch: Also known as SncSwitch. It models the switched forwarding of traffic (traffic flow) between EPs and is present where there is protection functionality in the FD.

- Termination Subset of CNM

  The Termination subset of the CNM (not covered in detail in this draft) supports modeling of the processing of transport characteristic information, such as termination, adaptation, OAM, etc. Artifacts representing the termination and adaptation and OAM construct include:

  o Logical Termination Point (LTP): See the LTP description in the Topology Subset o Layer Protocol (LP): This identifies the type of signal and is the anchor for transport layer protocol specific definitions, which are modeled in, e.g., [G.874.1] for OTN, [G.8052] for transport Ethernet, and [G.8152] for MPLS-TP.

- Resilience Subset of CNM

  The Resilience subset provides a view of the model for resilience (including protection and restoration) and encompasses:

  o The basic resilience model structure

  o The key attributes relevant to resilience

  o The application of the resilience model to various cases

3.1.2. Core Foundation Model

To communicate about an entity, it is important to have some way of referring to that entity, i.e., to have some way of referencing it. The Core Foundation model defines the artifacts for referencing entities; i.e.:

- Global Unique ID (GUID):

  An identifier that is globally unique where an identifier is a property of an entity/role with a value that is unique within an identifier space, where the identifier space is itself unique, and immutable. The identifier therefore represents the identity of the entity/role. An identifier carries no semantics with respect to the purpose of the entity.)

- Local ID:

  An identifier that is unique in the context of some scope that is less than the global scope (where an identifier is as defined in GUID above).

- Name:

A property of an entity with a value that is unique in some namespace but may change during the life of the entity. A name carries no semantics with respect to the purpose of the entity.

- Label:

  A property of an entity with a value that is not expected to be unique and is allowed to change. A label carries no semantics with respect to the purpose of the entity and has no effect on the entity behavior or state.

The Core Foundation model also provides the opportunity to extend any entity using the Extension structure.

The model also defines two foundation object classes:

- GlobalClass:

  Super class of object classes for which their instances can exist on their own right, e.g. NE, LTP, FD, Link, and FC. Global classes shall have one and only one globally unique identifier (GUID) and may have zero or more local identifiers, zero or more names, zero or more labels, zero or more extensions.

- LocalClass:

  Super class of object classes for which the existence of their instances depends on instances of global classes; e.g., LP (of LTP), EP (of FC), and LE (of Link). Local classes shall have at least one local identifier, may have zero or more names, zero or more labels, zero or more extensions.

```
--------------------------------------------------------------------
I                                                                  I
I                                                                  I
I                                                                  I
I                                                                  I
I             Artifacts for Referencing of Entities                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 3-1 Artifacts for Referencing of Entities

The Core Foundation model also defines a State_Pac artifact, which is a package of state attributes. The State_Pac is inherited by GlobalClass and LocalClass object classes. The State_Pac consists of the following state-related attributes:

- Operational State:

  Read-only with values: DISABLED, ENABLED

- Administrative State:

  Read-write with values: LOCKED, UNLOCKED

- Lifecycle State:

  Read-write with values: PLANNED, POTENTIAL, INSTALLED, PENDING_REMOVAL

```
--------------------------------------------------------------------
I                                                                   I
I                                                                   I
```

```
   I                                                                  I
   I                       States of Objects                          I
   I                                                                  I
   I                                                                  I
   I                     (only in PDF version)                        I
   I                                                                  I
   I                                                                  I
   I                                                                  I
   --------------------------------------------------------------------
```

Figure 3-2 States of Objects

3.1.3. Core Physical Model

The Physical model provides a view of the model for physical entities (including equipment, holders and connectors). This model also specifies the relationship between the connector and the LTP, and the relationship between physical and functional views.

```
   --------------------------------------------------------------------
   I                                                                  I
   I                                                                  I
   I                                                                  I
   I                                                                  I
   I                       Basic Equipment Pattern                    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 3-3 Basic Equipment Pattern

3.1.4. Core Specification Model

There are several related needs that have given rise to the Specification model:

- Provide machine readable form of specific localized behavior:

  o Representing rules related to restrictions of specific cases of use of the model

  o Representing capabilities of specific cases of use

- Enable the introduction of run time schema where the essential structure of the model is known up front (at compile time) but the details are not

- Reduce the clutter in a representation where a set of details take the same values for all instances that are related to a specific case

- Allow leverage of existing standards definitions (e.g., technology/application specific) in a machine readable language

The combination of the above resulted in a separation in the model of definitions of structure and content such that an instance of a class from one model fragment could have an association instance to another model fragment to enable the provision of a fragment of definition of the class and of subordinates.

The aim of all specification definitions is that they be rigorous definitions of specific cases of usage and enable machine interpretation where traditional interface designs would only allow human interpretation.

The following dedicated spec structures have been considered:

- FC spec: Main focus to provide a representation of the effective internal structure of a ForwardingConstruct (FC)

- LTP and LP spec: Main focus to provide a representation of Layer Protocol (LP) specific parameters for the Logical Termination Point (LTP)

- FD and Link spec: Main focus on capacity and forwarding enablement restrictions

- Equipment spec: Main focus to provide a representation of equipping constraints

```
--------------------------------------------------------------------
I                                                                  I
I                                                                  I
I                                                                  I
I                                                                  I
I          Class Diagram of the Spec Model of LTP and LP           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 3-4 Class Diagram of the Spec Model of LTP and LP

3.2. Other Models

In addition to the Core Model, the ONF-CIM includes forwarding technology and application specific models. The forwarding technology models of the ONF-CIM (see [ONF TR-512]) encompasses transport technology layers 0, 1, and 2.

4. High Level Description of the Topology Subset of the CNM

This section provides a high-level overview of the Topology Subset of the CNM. Figure 4-1 below is a skeleton class diagram illustrating the key object classes. To avoid cluttering the figure, not all associations have been shown and all of the attributes were omitted.

```
--------------------------------------------------------------------
I                                                                  I
I                                                                  I
I                                                                  I
I                                                                  I
I               Overview of the CNM Topology Subset                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 4-1 Overview of the CNM Topology Subset

4.1. Object Classes of the CNM Topology Subset

This section describes the object classes of the Topology Subset of the CNM. Relationships between these classes are described in section 4.2 below

4.1.1. LogicalTerminationPoint (LTP) and LayerProtocol (LP)

The LogicalTerminationPoint (LTP) object class encapsulates the termination, adaptation and OAM functions of one or more transport protocol layers. The structure of the LTP supports all transport protocols including circuit and packet forms. Each transport layer is represented by a LayerProtocol (LP) instance. The LayerProtocol instances of the LTP can be used for controlling the termination and OAM functionality of that layer. It can also be used for controlling the adaptation (i.e. encapsulation and/or multiplexing of client signal). Where the client/server relationship is fixed 1:1 and immutable, the different layers can be encapsulated in a single LTP instance. Where there is a n:1 relationship between client and server, the layers must be split over separate instances of LTP.

The LP object class is defined with generic attributes "layerProtocolName" for indicating the supported transport layer protocol.

Transport layer specific properties (such as layer-specific termination and adaptation properties) are modeled as attributes of conditional packages (called "_Pacs" in the UML notation of the ONF-CIM) associated with the LP object class.

4.1.2. ForwardingDomain (FD)

The ForwardingDomain (FD) object class models the switching and routing capabilities (see "subnetwork" topological component in [G.852.2] and [TMF612]), which is used to effect forwarding of transport characteristic information and offers the potential to enable forwarding. It represents the resource that supports flows across the FD. The FD object can hold zero or more instances of ForwardingConstruct (FC) (representing constrained forwarding, not discussed further in this document, covering connections, VLANs etc) of one or more layer networks; e.g., OCh, ODU, ETH, and MPLS-TP. The

FD object provides the context for operations that create/modify/delete FCs.

The FD object class supports a recursive aggregation relationship such that the internal construction of an FD can be exposed as multiple lower level FDs and associated Links (partitioning) (see section 4.2.1.)

At the lowest level of recursion, a FD (within a network element) could represent a switch matrix (i.e., a fabric).

Note that an NE can encompass multiple switch matrices (FDs), as described in section 4.2.2. An instance of FD is associated with zero or more LTP objects, as described in section 4.2.3.

4.1.3. Link and Link Port

The Link object class models the adjacency between two or more ForwardingDomains (FDs).

In its basic form (i.e., point-to-point Link) it associates a set of LTP clients on one FD with an equivalent set of LTP clients on another FD. Like the FC, the Link has endpoints (LinkPort) which take roles in the context of the function of the Link. A point-to-point Link can be a TE Link and support parameters such as capacity, delay etc. These parameters depend on the type of technology that supports the link.

A Link can be terminated on two or more FDs. This provides support for technologies such as PON and Layer 2 MAC in MAC configurations.

The LinkPort further details the relationship between FD and Link for asymmetric cases.

A FD may aggregate Links (see section 4.2.5).

The Link can support multiple transport layers via the associated LTP object. An instance of Link can be formed with the necessary properties according to the degree of virtualization. For implementation optimization, multiple layer-specific links can be merged and represented as a single Link instance.

4.1.4. Network Element (NE)

The NetworkElement (NE) object class represents a network element (traditional NE) in the data plane or a virtual network element visible in an interface where virtualization is used.

In the direct interface from a SDN controller to a network element in the data plane, the NE object defines the scope of control for the resources within the network element, e.g., internal transfer of user information between the external terminations (ports), encapsulation, multiplexing/demultiplexing, and OAM functions, etc. The NE provides the scope of the naming space for identifying objects representing the resources within the network element.

Where virtualization is employed, the NE object represents a virtual NE (VNE). The mapping of the VNE to the NEs is the internal matter of the SDN controller that offers the view of the VNE. Via the interface between hierarchical SDN controllers, NE instances can be created (or deleted) for providing (or removing) virtual views of the combination of slices of network elements in the data plane.

4.2. Relationships between Object Classes of the Topology Subset

4.2.1. ForwardingDomain Recursive Aggregation (HigherLevelFdEncompassesLowerLevelFds Aggregation)

Figure 4-2 below provides a pictorial example of ForwardingDomain (FD) recursion with Links.

```
--------------------------------------------------------------------
I                                                                  I
I                                                                  I
I                                                                  I
I                                                                  I
```

```
   I              ForwardingDomain recursion with Links            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 4-2 ForwardingDomain recursion with Links

Figure 4-2 shows a UML fragment including the Link and ForwardingDomain (FD). For simplicity it is assumed here that the Links and FDs are for a single LayerProtocol (LP) although it can be seen from the detailed figure earlier in this section that both a FD and link can support a list of LPs.

The pictorial form shows a number of instances of FD interconnected by Links and shows nesting of FDs. The recursive aggregation "HigherLevelFdEncompassesLowerLevelFds" relationship (represented by an open diamond) supports the FD nesting but it should be noted that this is intentionally showing no lifecycle dependency between the lower FDs and the higher ones that nest them (to do this composition, a black diamond would have been used instead of the open diamond). This is to allow for rearrangements of the FD hierarchy (e.g. when regions of a network are split or merged). This emphasizes that the nesting is an abstraction rather than decomposition. The underlying network still operates regardless of how it is perceived in terms of aggregating FDs. The model allows for only one hierarchy.

4.2.2. Network Elements encompassing ForwardingDomains (NeEncompassesFds Aggregation)

Figure 4-3 below provides a pictorial example of ForwardingDomain (FD) recursion with Links and NEs.

```
--------------------------------------------------------------------
I                                                                  I
I                                                                  I
I                                                                  I
I                                                                  I
I          ForwardingDomain recursion with Links and NEs           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 4-3 ForwardingDomain recursion with Links and NEs

Figure 4-3 above shows an overlay of NetworkElement (NE) on the ForwardingDomains and a corresponding fragment of UML showing only the ForwardingDomain and NetworkElement classes.

The figure emphasizes that one level of abstraction of ForwardingDomain is bounded by an NE. This is represented in the UML fragment by the composition association (black diamond) that explains that there is a lifecycle dependency in that the ForwardingDomain at this level that cannot exist without the NE. The figure also shows that a ForwardingDomain need not be bounded by an NE (as explained in the UML fragment by the 0..1 composition) and that a ForwardingDomain may have smaller scope than the whole NE (even when considering only a single LayerProtocol as described below).

In one of the cases depicted (e.g., the right hand side NE encompassing two FDs), the two ForwardingDomains in the NE are completely independent. In the other cases depicted (e.g., the left hand side NE encompassing three FDs) the subordinate ForwardingDomains are themselves joined by Links emphasizing that the NE does not necessarily represent the lowest level of relevant network decomposition.

The figure also emphasizes that just because one ForwardingDomain at a particular level of decomposition of the network happens to be the one bounded by an NE does not mean that all ForwardingDomains at that level are also bounded by NEs.

4.2.3. ForwardingDomain association with LTPs (FdAggregatesLtps Composition)

An instance of FD is associated with zero or more LTP objects via the "FdAggregatesLtps" composition.

4.2.4. ForwardingDomain aggregating Links (FdEncompassesLinks)

A ForwardingDomain can aggregate links. An example of ForwardingDomain Recursive Aggregation with Links is shows in section 4.2.1 above.

However, the FdAggregatesLink association is not modeled because this association can be inferred from the higherLevelFdContainsLowerLevelFd association together with the linkHasAssociatedFds association.

4.2.5. ForwardingDomain aggregating NEs

A ForwardingDomain can aggregate Network Elements. An example of ForwardingDomain Recursive Aggregation with Links and NEs is shown in section 4.2.2 above.

However, the FdAggregatesNe association is not modeled because this association can be inferred from higherLevelFdContainsLowerLevelFd association and together with the NeEncompassesFd association.

5. Detailed Description of the Topology Subset

The two key classes related to Topology are the ForwardingDomain (FD) and the Link. For simple cases the FD represents the switching capability in the network and the Link represents adjacency. These are depicted in the context of other model classes in Figure 5-1.

```
 --------------------------------------------------------------------
 I                                                                  I
 I                                                                  I
 I                                                                  I
 I                                                                  I
 I     Object Classes and Relationships in the Topology Subset      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 5-1 Object Classes and Relationships in the Topology Subset

Figure 5-1 shows a lightweight view of the model omitting the attributes (where appropriate these will be described later in this section).

The FD and Link will be described in detail later in the document. Figure 5-1 focuses on interrelationships and these will be the focus of this section. The figure shows that:

- An FD may be a subordinate part of a NetworkElement (NE) or may be larger than, and independent of, any NE.

- An FD may encompass lower level FDs. This may be such that:

  o A FD directly contained in an NE is divided into smaller parts o A FD not encompassed by an NE is divided into smaller parts some of which may be encompassed by NEs

   o The FD represents the whole network

- An FD encompasses Links that interconnect any FDs encompassed by the FD

- A Link may aggregate Links in several ways

   o In parallel where several links are considered as one

   o In series where Links chain to form a Link of a greater span

      . Note that this case requires further development in the model

- A Link has associated FDs that it interconnects

   o A Link may interconnect 2 or more FDs

      . Note that it is usual for a Link to interconnect 2 FDs but there are cases where many FDs may be interconnected by a Link

- A Link has LinkPorts that represent the ports of the Link itself

   o LinkPorts are especially relevant for multi-ended asymmetric Link

- A LinkPort aggregates LogicalTerminationPoints (LTPs) that bound the Link. The LTP represent a stack LayerProtocol terminations where the details of each is held in the LayerProtocol (LP). The LTP may be:

   o Part of an NE

   o Conceptually independent from any NE

- A LinkPort references LTPs on which the Link associated to the LE terminates

Both the Link and FD are subclasses of ForwardingEntity (an abstract class, i.e. a class that will never be instantiated) and hence they can acquire contents from the conditional packages (_Pacs). The conditional packages provide all key topology properties.

5.1. Forwarding Entity

As noted in the previous section the two key topology classes are Forwarding Domain (FD) and Link (L).

The FD topological component is used to show the potential to enable forwarding. At the lowest level of recursion, an FD (within a network element (NE)) represents a switch matrix (e.g., a fabric). Note that an NE can encompass multiple switch matrices (FDs).

As noted earlier the Link models adjacency between two or more Forwarding Domains (FD).

Both the link and the FD have the potential to handle more than one layerProtocol (both have a layerProtocolNameList attribute).

As shown in Figure 5-1 an object class "ForwardingEntity" has been defined to collect topology-related properties (characteristics etc.) that are common for FD and Link.

A ForwardingEntity is an abstract representation of the emergent effect of the combined functioning of an arrangement of components (running hardware, software running on hardware, etc). The effect can be considered as the realization of the potential for apparent communication adjacency for entities that are bound to the terminations at the boundary of the ForwardingEntity.

The ForwardingEntity enables the creation of constrained forwarding to achieve the apparent adjacency. The apparent adjacency has intended performance degraded from perfect adjacency and a statement of that degradation is conveyed via the attributes of the packages associated with this class. In the model both ForwardingDomain and Link are ForwardingEntities.

This abstract class is used as a modeling approach to apply packages of attributes to both Link and ForwardingDomain. Link and ForwardingDomain are the key ForwardingEntities.

5.2. Characteristics of Topological Entity

As noted above the characteristic of a TopologicalEnity are covered by the conditional packages (_PACs).

```
   --------------------------------------------------------------------
   I                                                                  I
   I                                                                  I
   I                                                                  I
   I                                                                  I
   I           Conditional Packages of Topological Entity             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 5-2 Conditional Packages of Topological Entity

5.2.1. Risk (RiskParameter_Pac)

The risk characteristics of a ForwardingEntity come directly from the underlying physical realization.

The risk characteristics propagate from the physical realization to the client and from the server layer to the client layer, this propagation may be modified by protection.

A ForwardingEntity may suffer degradation or failure as a result of a problem in a part of the underlying realization.

The realization can be partitioned into segments which have some relevant common failure modes.

There is a risk of failure/degradation of each segment of the underlying realization.

Each segment is a part of a larger physical/geographical unit that behaves as one with respect to failure (i.e. a failure will have a high probability of impacting the whole unit (e.g. all fibers in the same cable).

Disruptions to that larger physical/geographical unit will impact (cause failure/errors to) all ForwardingEntities that use any part of that larger physical/geographical entity.

Any ForwardingEntity that uses any part of that larger physical/geographical unit will suffer impact and hence each ForwardingEntity shares risk.

The identifier of each physical/geographical unit that is involved in the realization of each segment of a Topological entity can be listed in the RiskParameter_Pac of that ForwardingEntity.

A segment has one or more risk characteristic.

Shared risk between two ForwardingEntities compromises the integrity of any solution that use one of those ForwardingEntity as a backup for the other.

Where two ForwardingEntities have a common risk characteristic they have an elevated probability of failing simultaneously compared to two ForwardingEntities that do not share risk characteristics.

- riskCharacteristicList: A list of risk characteristics (RiskCharacteristic) for consideration in an analysis of shared risk. Each element of the list represents a specific risk consideration.

- RiskCharacteristic: The information for a particular risk characteristic where there is a list of risk identifiers related to that characteristic. It includes:

  o riskCharacteristicName: The name of the risk characteristic. The characteristic may be related to a specific degree of closeness. For example a particular characteristic may apply to failures that are localized (e.g. to one side of a road) where as another characteristic may relate to failures that have a broader impact (e.g. both sides of a road that crosses a bridge). Depending upon the importance of the traffic being routed different risk characteristics will be evaluated.

  o riskIdentifierList: A list of the identifiers of each physical/geographic unit (with the specific risk characteristic) that is related to a segment of the ForwardingEntity.

5.2.2. TransferCost_Pac

The cost characteristics of a ForwardingEntity not necessarily correlated to the cost of the underlying physical realization.

They may be quite specific to the individual ForwardingEntity e.g. opportunity cost. Relates to layer capacity

There may be many perspectives from which cost may be considered for a particular ForwardingEntity and hence many specifc costs and potentially cost algorithms.

Using an entity will incur a cost.

- costCharcteristicList: The list of costs (CostCharacteristic) where each cost relates to some aspect of the Link

    o CostCharcteristic: The information for a particular cost characteristic

        . costName: The cost characteristic will related to some aspect of the ForwardingEntity (e.g. $ cost, routing weight). This aspect will be conveyed by the costName

        . costValue: The specific cost.

        . costAlgorithm: The cost may vary based upon some properties of the ForwardingEntity. The rules for the variation are conveyed by the costAlgorithm.

5.2.3. TransferTiming_Pac

A link will suffer effects from the underlying physical realization related to the timing of the information passed by the link.

- fixedLatencyCharacteristic: A ForwardingEntity suffers delay caused by the realization of the servers (e.g. distance related; FEC encoding etc.) along with some client specific processing. This is the total average latency effect of the ForwardingEntity

- jitterCharacteristic: High frequency deviation from true periodicity of a signal and therefore a small high rate of change of transfer latency. Applies to TDM systems (i.e., not packet based systems).

- wanderCharacteristics: Low frequency deviation from true periodicity of a signal and therefore a small low rate of change of transfer latency. Applies to TDM systems (i.e., not packet based systems).

- queuingLatencyList: The effect on the latency of a queuing process. This only has significant effect for packet based systems and has a complex characteristic (QueuingLatency).

  o QueuingLatency: Provides information on latency characteristic for a particular stated trafficProperty.

5.2.4. TransferIntegrity_Pac

Transfer integrity characteristic covers expected (specified) error, loss and duplication signal content as well as any damage of any form to total link and to the client signals.

- errorCharacteristic: describes the degree to which the signal propagated can be errored. Applies to TDM systems as the errored signal will be propagated and not packet as errored packets will be discarded.

- lossCharacteristic: Describes the acceptable characteristic of lost packets where loss may result from discard due to errors or overflow. Applies to packet systems and not TDM (as for TDM errored signals are propagated unless grossly errored and overflow/underflow turns into timing slips).

- repeatDeliveryCharacteristic: Primarily applies to packet systems where a packet may be delivered more than once (in fault recovery for example). It can also apply to TDM where several frames may be received twice due to switching in a system with a large differential propagation delay.

- deliveryOrderCharacteristic: Describes the degree to which packets will be delivered out of sequence. Does not apply to TDM as the TDM protocols maintain strict order.

- unavailableTimeCharacteristic: Describes the duration for which there may be no valid signal propagated.

- serverIntegrityProcessCharacteristic: Describes the effect of any server integrity enhancement process on the characteristics of the ForwardingEntity.

5.2.5. TransferCapcity_Pac

The ForwardingEntity derives capacity from the underlying realization.

A ForwardingEntity may be an abstraction and virtualization of a subset of the underlying capability offered in a view or may be directly reflecting the underlying realization.

A ForwardingEntity may be directly used in the view or may be assigned to another view for use.

The clients supported by a multi-layer ForwardingEntity may interact such that the resources used by one client may impact those available to another. This is derived from the LTP spec details.

A ForwardingEntity represents the capacity available to user (client) along with client interaction and usage.

A ForwardingEntity may reflect one or more client protocols and one or more members for each profile.

- totalPotentialCapacity: A "best case" view of the capacity of the ForwardingEntity assuming that any shared capacity is available to be taken.

Note that this area is still under development to cover concepts such as:

- exclusiveCapacityList: The capacity allocated to this ForwardingEntity for its exclusive use

- sharedCapacityList: The capacity allocated to this ForwardingEntity that is not exclusively available as it is shared with others.

- assignedAsExclusiveCapacityList: The capacity assigned from this TopologicalEnity to another ForwardingEntity for its exclusive use

- assignedAsSharedCapacityList: The capacity assigned to one or more other ForwardingEntities for shared use where the interaction follows some stated algorithm.

- Capacity which includes:

  - totalSize

  - numberOfUsageInstances

  - maximumUsageSize

  - numberingRange

5.2.6. Validation_Pac

Validation covers the various adjacenct discovery and reachability verification protocols. Also may cover Information source and degree of integrity.

- validationMechanismList: Provides details of the specific validation mechanism(s) used to confirm the presence of an intended ForwardingEntity.

5.2.7. LayerProtocolTransition_Pac

Relevant for a Link that is formed by abstracting one or more LTPs (in a stack) to focus on the flow and deemphasize the protocol transformation.

This abstraction is relevant when considering multi-layer routing.

The layer protocols of the LTP and the order of their application to the signal is still relevant and need to be accounted for. This is derived from the LTP spec details.

This Pac provides the relevant abstractions of the LTPs and provides the necessary association to the LTPs involved.

Links that included details in this Pac are often referred to as Transitional Links.

- transitionedLayerProtocolList: Provides the ordered structure of layer protocol transitions encapsulated in the ForwardingEntity. The ordering relates to the LinkEnd role.

6. Purpose Specific IM Example - Transport API Topology Service

In order to provide some further clarity, this section provides a high level introduction to a Purpose Specific IM, the Transport API (T-API) Topology service, which has been derived from the ONF Common Information Model (ONF-CIM) according to the principles in [I-D.betts].

The context of the T-API refers to the scope and control and naming that a particular SDN controller, manager or a client application has with respect to the information it operates on internally or exchanges over an interface. The following sections further describe this purpose specific IM and relationship to the ONF-CIM.

6.1. T-API IM Constructs

The T-API IM uses terminology that is considered to be more familiar to the transport network management community and maps to the constructs defined in the ONF-CIM CNM Topology model. The following table provides a high level summary of the mapping of the constructs relevant to the T-API Topology Service.

Mapping of CIM and T-API IM Constructs
ONF-CIM CNM Terminology   T-API IM Terminology
NetworkControlDomain   Context
Topology   ForwardingDomain
(FD)   Node
Link   Link
TransitionalLink
NodeEdgePoint   LogicalTermi
nationPoint (LTP) ServideEndPoint

The following provides a brief description of these T-API IM constructs.

- o Link: A Link is an abstract representation of the effective adjacency between two or more associated Nodes in a Topology. It is terminated by Node-Edge-Points of the associated Nodes.

- o Node: A Node is an abstract representation of the forwarding-capabilities of a particular set of Network Resources. It is described in terms of an aggregation of set of ports (Node-Edge-Point) belonging to those Network Resources and the potential to enable forwarding of information between those edge ports.

- o Node-Edge-Point: A Node-Edge-Point represents the inward network-facing aspects of the edge-port functions that access the forwarding capabilities provided by the Node. Hence it provides an encapsulation of addressing, mapping, termination, adaptation and OAM functions of one or more transport layers (including circuit and packet forms) performed at the entry and exit points of the Node.

- o Topology: A Topology is an abstract representation of the topological-aspects of a particular set of Network Resources. It is described in terms of a network of set of Nodes and Links that enable the forwarding-capabilities of that particular set of Network Resources.

- o Service-End-Point: A Service-End-Point represents the outward customer-facing aspects of the edge-port functions that access the forwarding capabilities provided by the Node. Hence it provides a limited, simplified view of interest to external clients (e.g. shared addressing, capacity, resource availability, etc) that enable the clients to request connectivity without the need to understand the provider network internals.

- o Transitional Link: A topological component that consists of the link port at the edge of one node and a corresponding link port at the edge of another node that operates on different layers or whose layer is the same but with different Layer

Information. A transitional link is supported/implemented by transport processing functions (e.g., adaptation/termination). A transitional link can be partitioned into parallel transitional links, or a concatenation of transitional links. It can also be partitioned into a concatenation of transitional links and zero or more links.

6.2. T-API Topology Service IM

The resultant high-level description for the T-API Topology Service constructs, based upon the pruned and refactored ONF-CIM, and the related Topology Service APIs are provided in Figure 6-1 below.

```
--------------------------------------------------------------------
I                                                                  I
I                                                                  I
I                                                                  I
I                                                                  I
I                    Topology Service Skeleton                       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 6-1 Topology Service Skeleton

The T-API Topology Service API enables the API client to, for example, retrieve Topology, Node, Link, and Edge-Point details.

- o Topology details: returns attributes of the Topology identified by the provided input ID. This includes references to lower-level Nodes and Links encompassed by that Topology. A NULL input value is expected to return the top-most Topology that corresponds to the scope of the entire Context including any Off-Network-Links.

- o Node details: Returns attributes of the Node identified by the provided input ID. Includes references to Node-Edge-Points aggregated by the Node, and attributes representing the identification, naming, states and forwarding capabilities of the Node.

- o Link details: Returns attributes of the Link identified by the provided input ID. Includes references to Node-Edge-Points terminating the Link, and references to the Nodes associated by the Link.

- o Node-Edge-Point details: Returns attributes of the Node-Edge-Point identified by the provided input ID, including references to Service-End-Points mapped to this Node-Edge-Point.

The API supports a retrieve-scope filter: LayerProtocol list. If set, the API call will return output that is relevant to the specified Layer only.

7. Usage of the IM Topology Subset regarding TE Topology DM

As discussed earlier, a data model (DM) may be derived from an IM. Examples of YANG DMs derived according to automated translation tools based upon mapping guidelines are provided in [OSSDN SNOMASS] at https://github.com/OpenNetworkingFoundation/Snowmass-ONFOpenTransport/tree/develop/YANG. It is possible to leverage the IM

Topology Subset to assess the consistency and completeness of related YANG modules under development.

8. Security Considerations

This informational document is intended only to provide a description of an interface-protocol-neutral information model, and the security concerns are therefore out of the scope of this document.

9. IANA Considerations

This document includes no request to IANA.

10. Conclusions

The information modeling described in this draft, which is relevant to Network Topology [ONF TR-512] [OSSDN SNOWMASS], can be leveraged in assessing the consistency and completeness of related YANG modules under development.

11. References

11.1. Normative References

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

11.2. Informative References

[I-D.betts] Betts, M., Davis, N., Lam, K., Zeuner, B., Mansfield, S. and P. Doolan, "Framework for Deriving Interface Data Schema from UML Information Models", draft-betts-netmod-framework-data-schema-uml-04 (work in progress), October 2016

[I-D.mansfield] Mansfield, S., Zeuner, B., Davis, N., Yun, X., Tochio, Y., Lam, K. and E. Varma, "Guidelines for Translation of UML Information Model to YANG Data Model", draft-mansfield-netmod-uml-to-yang-03 (work in progress), October 2016

[ONF TR-512] ONF TR-512 "ONF-CIM Core Model base document 1.2" (https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-512_CIM_(CoreModel)_1.2.zip), September 2016

[ONF TR-513] ONF TR-513 " Common Information Model Overview 1.2" (https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-513_CIM_Overview_1.2.pdf), September 2016

[ONF TR-514] ONF TR-514 "UML Modeling Guidelines 1.2" (https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-514_UML_Modeling_Guidelines_v1.2.pdf), September 2016

[ONF TR-515] ONF TR-515 "Papyrus Guidelines 1.2" (https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-515_Papyrus_Guidelines_v1.2.pdf), September 2016

[OSSDN SNOWMASS] Open Source SDN SNOWMASS/Open Transport API Specifications (https://github.com/OpenNetworkingFoundation/Snowmass-ONFOpenTransport)

[G.7711] Recommendation ITU-T G.7711/Y.17022 "Generic protocol-neutral information model for transport resources", September 2016

[G.874.1] Recommendation ITU-T G.874.1 "Optical transport network: Protocol-neutral management information model for the network element view", September 2016

[G.8052] Recommendation ITU-T G.8052/Y.1346 "Protocol-neutral management information model for the Ethernet Transport capable network element", September 2016

[G.8152] Recommendation ITU-T G.8152/Y.1375 "Protocol-neutral management information model for the MPLS-TP network element", September 2016

[G.852.2] Recommendation ITU-T G.852.2 "Enterprise viewpoint description of transport network resource model", March 1999

[TMF612] TM Forum 612 "MTOSI Information Agreement", October 2014

12. Contributors

Karthik Sethuraman
NEC

Email: karthik.sethuraman@necam.com

13. Acknowledgments

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

Authors' Addresses

Kam Lam
Alcatel-Lucent, USA

Phone: +1 732 331 3476
Email: kam.lam@alcatel-lucent.com

Eve Varma
Alcatel-Lucent, USA

Email: eve.varma@alcatel-lucent.com

Paul Doolan
Coriant, Germany

Phone: +1 972 357 5822
Email: paul.doolan@coriant.com

Malcolm Betts
ZTE, China

Phone: +1 678 534 2542
Email: malcolm.betts@zte.com.cn

Nigel Davis
Ciena, UK

Email: ndavis@ciena.com

Bernd Zeuner
Deutsche Telekom,   Germany

Phone: +49 6151 58 12086
Email: b.zeuner@telekom.de

Italo Busi
Huawei, China

Email: Italo.Busi@huawei.com

Scott Mansfield
Ericsson, Sweden

Phone: 1 724 931 9316
Email: scott.mansfield@ericsson.com

Yuji Tochio
Fujitsu, Japan

Phone: 81 44 754 8829
Email: tochio@jp.fujitsu.com

Ricard Vilalta
CTTC, Spain

Phone:
Email: ricard.vilalta@cttc.es

Victor Lopez
Telefonica, Spain

Phone:
Email: victor.lopezalvarez@telefonica.com

Network Working Group
Internet Draft

Intended status: Informational

Expires: September 2015

Young Lee
Huawei

Sergio Belotti
Alcatel-Lucent

Dhruv Dhody
Huawei

Daniele Ceccarelli
Ericsson

March 9, 2015

Information Model for Abstraction and Control of Transport Networks

draft-leebelotti-actn-info-01.txt

Abstract

This draft provides an information model for abstraction and control of transport networks.

Status of this Memo

This Internet-Draft will expire on September 9, 2015.

Copyright Notice

Table of Contents

1. Introduction

This draft provides information model for the ACTN interfaces identified in the ACTN architecture and framework document [ACTN-Frame].

The ACTN architecture identified a three-tier control hierarchy as depicted in Figure 1:

- Customer Network Controllers (CNC)
- Multi-Domain Service Coordinator (MDSC)
- Physical Network Controllers (PNC).

```
VPN customer         NW Mobile Customer     ISP NW service Customer
    |                        |                            |
+-------+                +-------+                    +-------+
| CNC-A |                | CNC-B |                    | CNC-C |
+-------+                +-------+                    +-------+
      \                      |                           /
       ----------            | CMI         ------------
                 \           |            /
                  +-----------------------+
                  |         MDSC          |
                  +-----------------------+
                 /           |            \
         --------            | MPI         ------------
        /                    |                          \
+-------+                +-------+                    +-------+
|  PNC  |                |  PNC  |                    |  PNC  |
+-------+                +-------+                    +-------+
```

Figure 1. A Three-tier ACTN control hierarchy

The two interfaces with respect to the MDSC, one north of the MDSC and the other south of the MDSC are referred to as CMI (CNC-MDSC Interface) and MPI (MDSC-PNC Interface), respectively. It is intended to model these two interfaces with one common model.

Section 2 provides a high-level applicability of ACTN based on a number of use-cases listed in the following:

- draft-cheng-actn-ptn-requirements-00 (ACTN Use-cases for Packet Transport Networks in Mobile Backhaul Networks)

- draft-dhody-actn-poi-use-case-03 (Packet Optical Integration (POI) Use Cases for Abstraction and Control of Transport Networks (ACTN))

- draft-fang-actn-multidomain-dci-01 (ACTN Use Case for Multi-domain Data Center Interconnect)

- draft-klee-actn-connectivity-multi-vendor-domains-03 (ACTN Use-case for On-demand E2E Connectivity Services in Multiple Vendor Domain Transport Networks)

- draft-kumaki-actn-multitenant-vno-00 (ACTN : Use case for Multi Tenant VNO)

- draft-lopez-actn-vno-multidomains-01 (ACTN Use-case for Virtual Network Operation for Multiple Domains in a Single Operator Network)

- draft-shin-actn-mvno-multi-domain-00 (ACTN Use-case for Mobile Virtual Network Operation for Multiple Domains in a Single Operator Network)

- draft-xu-actn-perf-dynamic-service-control-02 (Use Cases and Requirements of Dynamic Service Control based on Performance Monitoring in ACTN Architecture)

2. ACTN Applications

This section provides the scope of the ACTN applicability to support the following applications.

- Coordination of Multi-destination Service Requirement/Policy (Section 2.2.1)
- Application Service Policy-aware Network Operation (section 2.2.2)
- Network Function Virtualization Service Enabled Connectivity (2.2.3)
- Dynamic Service Control Policy Enforcement for Performance/Fault Management (Section 2.2.4)
- E2E VN Survivability and Multi-Layer (Packet-Optical) Coordination for Protection/Restoration (Section 2.2.5)

2.1.1. Coordination of Multi-destination Service Requirement/Policy

```
                             +----------------+
                             |       CNC      |
                             |   (Global DC   |
                             |   Operation    |
                             |    Control)    |
                             +--------+-------+
                                      | |  Service Requirement/Policy:
                                      | |  - Endpoint/DC location info
                                      | |  - Endpoint/DC dynamic
                                      | |    selection policy
                                      | |    (for VM migration, DR, LB)
                                      | v
                            +---------+--------+
                            |  Multi-domain    | Service policy-driven
                            |Service Controller| dynamic DC selection
                            +-----+---+---+----+
                                  |   |   |
                                  |   |   |
                 +----------------+   |   +----------------+
                 |                    |                    |
           +-----+-----+       +-----+------+      +------+-----+
           |   PNC for |       |  PNC for   |      |  PNC for   |
           | Transport |       | Transport  |      | Transport  |
           | Network A |       | Network B  |      | network C  |
           +-----------+       +------------+      +------------+
                 |                    |                   |
+---+      ------               ------              ------       +---+
|DC1|--////      \\\\       ////      \\\\      ////      \\\\---+DC4|
+---+ |              |     |              |    |              |  +---+
      |     TN A     +-----+     TN B     +----+      TN C    |
      /              |     |              |    |              |
     / \\\\      ////     / \\\\      ////      \\\\      ////
   +---+   ------        /      ------    \         ------ \
   |DC2|                /                  \                \+---+
   +---+               /                    \                |DC6|
                     +---+                   \ +---+         +---+
                     |DC3|                    \|DC4|
                     +---+                     +---+

                                                DR: Disaster Recovery
                                                LB: Load Balancing
```

Figure 2: Service Policy-driven Data Center Selection

Figure 2 shows how VN service policies from the CNC are incorporated by the MDSC to support multi-destination applications. Multi-destination applications refer to applications in which the selection of the destination of a network path for a given source needs to be decided dynamically to support such applications.

Data Center selection problems arise for VM mobility, disaster recovery and load balancing cases. VN’s service policy plays an important role for virtual network operation. Service policy can be static or dynamic. Dynamic service policy for data center selection may be placed as a result of utilization of data center resources supporting VNs. The MSDC would then incorporate this information to meet the service objective of this application.

2.1.2. Application Service Policy-aware Network Operation

```
                           +----------------+
                           |       CNC      |
                           |   (Global DC   |
                           |   Operation    |
                           |    Control)    |
                           +--------+-------+
                                    | | Application Service Policy
                                    | | - VNF requirement (e.g.
                                    | |   security function, etc.)
                                    | | - Location profile for each VNF
                                    | v
                          +---------+--------+
                          |  Multi-domain    |  Dynamically select the
                          |Service Controller|  network destination to
                          +-----+---+---+----+  meet VNF requirement.
                                |   |   |
                                |   |   |
                +---------------+   |   +----------------+
                |                   |                    |
         +------+-----+       +-----+------+      +------+-----+
         |   PNC for  |       |  PNC for   |      |  PNC for   |
         | Transport  |       | Transport  |      | Transport  |
         | Network A  |       | Network B  |      | network C  |
         |            |       |            |      |            |
         +------------+       +------------+      +------------+
                |                   |                    |
{VNF b}         |                   |                    |  {VNF b,c}
+---+      ------               ------              ------     +---+
|DC1|--////      \\\\       ////      \\\\      ////      \\\\-|DC4|
+---+ |              |     |              |    |              |+---+
     |      TN A      +---+     TN B       +--+      TN C      |
      /              |     |              |    |              |
     / \\\\      ////     / \\\\      ////      \\\\      ////
   +---+   ------        /      ------    \         ------ \
   |DC2|                /                  \                \\+---+
   +---+               /                    \                 |DC6|
    {VNF a}         +---+                    +---+            +---+
                    |DC3|                    |DC4|      {VNF a,b,c}
                    +---+                    +---+
            {VNF a, b}                      {VNF a, c}
```

Figure 3: Application Service Policy-aware Network Operation

This scenario is similar to the previous case in that the VN service policy for the application can be met by a set of multiple destinations that provide the required virtual network functions (VNF). Virtual network functions can be, for example, security functions required by the VN application. The VN service policy by the CNC would indicate the locations of a certain VNF that can be fulfilled. This policy information is critical in finding the optimal network path subject to this constraint. As VNFs can be dynamically moved across different DCs, this policy should be dynamically enforced from the CNC to the MDSC and the PNCs.

2.1.3. Network Function Virtualization Service Enabled Connectivity

```
                           +----------------+
                           |       CNC      |
                           |   (Global DC   |
                           |   Operation    |
                           |    Control)    |
                           +--------+-------+
                                    | | Service Policy
                                    | | (e.g., firewall, traffic
                                    | | optimizer)
                                    | |
                                    | v
                          +---------+--------+
                          |  Multi-domain    | Select network
                          |Service Controller| connectivity subject to
                          +-----+---+---+----+ meeting service policy
                                |   |   |
                                |   |   |
                +---------------+   |   +----------------+
                |                   |                    |
         +------+-----+       +-----+------+      +------+-----+
         |   PNC for  |       |  PNC for   |      |  PNC for   |
         | Transport  |       | Transport  |      | Transport  |
         | Network A  |       | Network B  |      | network C  |
         |            |       |            |      |            |
         +------------+       +------------+      +------------+
                |                   |                    |
                |                   |                    |
+---+      ------               ------              ------     +---+
|DC1|--////      \\\\       ////      \\\\      ////      \\\\-|DC4|
+---+ |              |     |              |    |              |+---+
     |      TN A      +---+     TN B       +--+      TN C      |
      /              |     |              |    |              |
     / \\\\      ////     / \\\\      ////      \\\\      ////
   +---+   ------        /      ------    \         ------ \
   |DC2|                /                  \                \\+---+
   +---+               /                    \                 |DC6|
                    +---+                    +---+            +---+
                    |DC3|                    |DC4|
                    +---+                    +---+
```

Figure 4: Network Function Virtualization Service Enabled Connectivity

Network Function Virtualization Services are usually setup between customers' premises and service provider premises and are provided mostly by cloud providers or content delivery providers. The context may include, but not limited to a security function like firewall, a traffic optimizer, the provisioning of storage or computation capacity where the customer does not care whether the service is implemented in a given data center or another.

These services may be hosted virtually by the provider or physically part of the network. This allows the service provider to hide his own resources (both network and data centers) and divert customer requests where most suitable. This is also known as "end points mobility" case and introduces new concepts of traffic and service provisioning and resiliency (e.g., Virtual Machine mobility).

2.1.4. Dynamic Service Control Policy Enforcement for Performance and Fault Management

```
+------------------------------------------------+
|            Customer Network Controller         |
+------------------------------------------------+
1.Traffic|  /|\4.Traffic           | /|\
Monitor& |   |  Monitor            |  | 8.Traffic
Optimize |   |  Result   5.Service |  | modify &
Policy   |   |             modify& |  | optimize
        \|/  |       optimize Req.\|/ | result
+------------------------------------------------+
|          Mult-domain Service Controller        |
+------------------------------------------------+
2. Path  |  /|\3.Traffic           |  |
Monitor  |   | Monitor             |  |7.Path
Request  |   | Result     6.Path   |  | modify &
         |   |            modify&  |  | optimize
        \|/  |       optimize Req.\|/ | result
+------------------------------------------------+
|           Physical Network Controller          |
+------------------------------------------------+
```

Figure 5: Dynamic Service Control for Performance and Fault Management

Figure 5 shows the flow of dynamic service control policy enforcement for performance and fault management initiated by customer per their VN. The feedback loop and filtering mechanism tailored for VNs performed by the MDSC differentiates this ACTN scope from traditional network management paradigm. VN level dynamic OAM data model is a building block to support this capability.

2.1.5. E2E VN Survivability and Multi-Layer (Packet-Optical) Coordination for Protection/Restoration

```
                        +----------------+
                        |   Customer     |
                        |   Network      |
                        |   Controller   |
                        +--------*-------+
                                 *  |     E2E VN Survivability Req.
                                 *  |     - VN Protection/Restoration
                                 *  v        - 1+1, Restoration, etc.
                          +------*-----+   - End Point (EP) info.
                          |            |
                          |    MDSC    | MDSC enforces VN survivability
                          |            | requirement, determining the
                          |            | optimal combination of Packet/
                          +------*-----+ Opticalprotection/restoration,
                                 *       Optical bypass, etc.
                                 *
                                 *
              **********************************************
              *               *             *              *
         +----*-----+    +----*----+   +----*-----+   +----*----+
         |PNC for   |    |PNC for  |   |PNC for   |   |PNC for  |
         |Access N. |    |Packet C.|   |Optical C.|   |Access N.|
         +----*-----+    +----*----+   +----*-----+   +---*-----+
              *             --*---          *             *
              *          ///      \\\       *             *
            --*---      |   Packet   |      *         ----*-
         ///      \\\   |    Core    +------+------///      \\\
        |  Access    +----\\      ///       *     |   Access   |
        |  Network   |      ---+--          *     |   Network  |  +---+
        |\\\      ///         |             *      \\\      ///---+EP6|
        |   +---+-  |         |       -----*          -+---+      +---+
      +-+-+     |   |         +----///      \\\      |   |
      |EP1|     |   +--------------+  Optical   |      |   |  +---+
      +---+     |                  |    Core    +-------+  +--+EP5|
              +-+-+                 \\\      ///              +---+
              |EP2|                   ------ |
              +---+                    |     |
                                     +--++   ++--+
                                     |EP3|   |EP4|
                                     +---+   +---+
```

Figure 6: E2E VN Survivability and Multi-layer Coordination for Protection and Restoration

Figure 6 shows the need for E2E protection/restoration control coordination that involves CNC, MDSC and PNCs to meet the VN survivability requirement. VN survivability requirement and its policy need to be translated into multi-domain and multi-layer network protection and restoration scenarios across different controller types. After an E2E path is setup successfully, the MSDC has a unique role to enforce policy-based flexible VN survivability requirement by coordinating all PNC domains.

As seen in Figure 6, multi-layer (i.e., packet/optical) coordination is a subset of this E2E protection/restoration control operation. The MDSC has a role to play in determining an optimal protection/restoration level based on the customer’s VN survivability requirement. For instance, the MDSC needs to interface the PNC for packet core as well as the PNC for optical core and enforce protection/restoration policy as part of the E2E protection/restoration. Neither the PNC for packet core nor the PNC for optical core is in a position to be aware of the E2E path and its protection/restoration situation. This role of the MSDC is unique for this reason. In some cases, the MDSC will have to determine and enforce optical bypass to find a feasible reroute path upon packet core network failure which cannot be resolved the packet core network itself.

To coordinate this operation, the PNCs will need to update its domain level abstract topology upon resource changes due to a network failure or other factors. The MSDC will incorporate all these update to determine if an alternate E2E reroute path is necessary or not based on the changes reported from the PNCs. It will need to update the E2E abstract topology and the affected CN’s VN topology in real-time. This refers to dynamic synchronization of topology from Physical topology to abstract topology to VN topology.

MDSC will also need to perform the path restoration signaling to the affected PNCs whenever necessary.

3. ACTN common interfaces information model

This section provides ACTN common interface information model to support primitives between controllers: CNC-MDSC and MDSC-PNC.

The basic primitives are required between the controllers. It is described between a client controller and a server controller. A client-server relationship is recursive between a CNC and a MDSC and between a MDSC and a PNC. In the CMI interface, the client is a CNC while the server is a MDSC. In the MPI interface, the client is a MDSC and the server is a PNC. At a minimum, the following primitives should be supported:

- Virtual Network (VN) Instantiate/Modify/Delete
- VN Topology Update (Push Model)

```
<VN> ::= <VN Identifier>

         <VN Action>

         <End-Point List>

         <VN Topology Metric>

         <Traffic-Matrix>

         <VN Survivability>

         <VN Status>

         <VN Topology>
```

Where

<VN Identifier> is an identifier that identifies a particular VN.

<VN Action> is an indicator if this <VN> is for (i) instantiate, (ii) modify; (iii) delete. There may be a case where a query of a VN is necessary before an instantiate request. This is subject to further investigation.

```
<End-Point List> ::= (<Interface Identifier>

                     [<Client Capability>])...

                     <Location Service Profile>

                     <End-Point Dynamic Selection Policy>
```

Where

It is assumed that a list of interface identifiers has been known to the server prior to the VN Query message flow.

<Client Capability> ::= <Client Interface Capability>

[<Client Service Policy>]

The Client Capability comprises the client interface capability (e.g., maximum interface bandwidth, etc.) and other Service policy information of the client.

<Client Service Policy> ::= <Customer-Level |

<Network-Level>

Where

<Customer-Level> pertains to end-client service policies which specify the end-client related service/operational policies. Details of this field will be supplied in a later revision.

<Network-Level> pertains to the policies related to multi-domain network operation assumed by the MDSC. For example, domain selection preference in the context of multi-domain networks is a network-level service policy. Details of this field will be supplied in a later version.

<Location Service Profile> describes the End-Point Location's support for certain Virtual Network Functions (VNFs) (e.g., security function, firewall capability, etc.).

<End-Point Dynamic Selection Policy> describes if the End-Point can support load balancing, disaster recovery or VM migration.

<VN Topology Metric> ::= <VN Topology Type>

<VN Topology Cost>

```
                              [<VN Topology Preference>]

                              [<VN Topology Objective Function>]

         Where

         <VN Topology Type> ::= <Path> | <Graph>

         <VN Topology Cost> describes a particular cost associated with
         the VN Topology link/path such as reservable bandwidth,
         maximum link/path capacity, latency, etc.

         <VN Topology Preference> describes if the request is

           .  a single vs. a bulk request,
           .  VN diversity preference (in case of a bulk request,
               whether VNs should be disjoint or not),
           .  SRLG is required in describing link/path topology, or
           .  Others TDB.


         <VN Topology Objective Function> indicates a specific
         objective function for computing a path. This only
         applies when the VN Topology Type is a path vector.




   <Traffic-Matrix> ::= <End-Point List>

                        <Connectivity Type>

                        <Connectivity Metric>

   Where

      <Connectivity Type> ::= <P2P> | <P2MP> | <MP|MP> | <MP|P>

                                  <Multi-destination>


      <Connectivity Matric> ::= <Bandwidth>
```

```
                             [<Latency>]


<VN Survivability> ::= <VN Protection Level>

                       <VN Survivability Policy>


Where

   <VN Protection Level> ::= <No Protection> | <1+1> | <1:N> |
                             <restoration>

   <VN Survivability Policy> ::= <Local Reroute Allowed>

                                 [<Domain Preference>]

                                 <Push Allowed>

                                 <Incremental Update>

   Where
```

<Local Reroute Allowed> is a delegation policy to the Server to allow or not a local reroute fix upon a failure of the primary LSP.

<Domain Preference> is only applied on the MPI where the MDSC (client) provides a domain preference to each PNC (server).

<Push Allowed> is a policy that allows a server to trigger an updated VN topology upon failure without an explicit request from the client.

<Increment Update> is another policy that triggers an increment update from the server.

<VN Status> is the status indicator whether the VN has been successfully instantiated/modified/deleted in the server network or not in response to <VN Action>.

<VN Topology> describes the resulting VN topology. Details of <VN Topology> are TDB.

4. References

4.1. Informative References

[ACTN-Frame] D. Ceccarelli, et al., "Framework for Abstraction and Control of Transport Networks", draft-ceccarelli-actn-framework, work in progress.

5. Contributors

Contributors' Addresses

Authors' Addresses

Young Lee
Huawei Technologies
5340 Legacy Drive
Plano, TX 75023, USA
Phone: (469)277-5838
Email: leeyoung@huawei.com

Sergio Belotti
Alcatel Lucent
Via Trento, 30
Vimercate, Italy
Email: sergio.belotti@alcatel-lucent.com

Dhruv Dhoddy
Huawei Technologies
Email: dhruv.ietf@gmail.com

Daniele Ceccarelli
Ericsson
Torshamnsgatan,48
Stockholm, Sweden
Email: daniele.ceccarelli@ericsson.com

TEAS Working Group X. Liu
Internet-Draft Ericsson
Intended status: Standards Track I. Bryskin
Expires: September 26, 2015 ADVA Optical Networking
V. Beeram
Juniper Networks
T. Saad
Cisco Systems Inc.
H. Shah
Ciena
March 25, 2015

YANG Data Model for TE Topologies
draft-liu-teas-yang-te-topo-01

Abstract

This document defines a YANG data model for representing and manipulating TE Topologies.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC6020] a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interface, such as CLI and programmatic APIs. This document defines a YANG data model for representing and manipulating TE Topologies. This model contains technology agnostic TE Topology building blocks that can be augmented and used by other technology-specific TE Topology models.

1.1. Tree Structure - Legend

A simplified graphical representation of the data model is presented in Section 3 of this document. The following notations are used for the YANG model data tree representation.

<status> <flags> <name> <opts> <type>

<status> is one of:
+ for current
x for deprecated
o for obsolete

<flags> is one of:
rw for read-write configuration data
ro for read-only non-configuration data
-x for execution rpcs
-n for notifications

<name> is the name of the node

If the node is augmented into the tree from another module, its name is printed as <prefix>:<name>

<opts> is one of:
? for an optional leaf or node
! for a presence container
* for a leaf-list or list
Brackets [<keys>] for a list’s keys
Curly braces {<condition>} for optional feature that make node conditional
Colon : for marking case nodes
Ellipses ("...") subtree contents not shown

Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

<type> is the name of the type for leafs and leaf-lists.

1.2. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are prefixed using the standard prefix associated with the corresponding YANG imported modules, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and corresponding YANG modules

1.3. Terminology

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

2. Design Considerations

2.1. Generic extensible Model

The TE Topology model proposed in this document is meant to be technology agnostic. Other technology specific TE Topology models can augment and use the building blocks provided by the proposed model.

2.1.1. Generic TE Link Attributes

The model covers the definitions for generic TE Link attributes - bandwidth, admin groups, SRLGs, switching capabilities, TE metric extensions etc.

2.1.2. Generic TE Node Attributes

The model covers the definitions for generic TE Node attributes like connectivity matrix.

```
+--rw connectivity-matrix* [id]
   +--rw id            uint32
   +--rw from-link
      +--rw topo-ref?       leafref
      +--rw node-ref?       leafref
      +--rw link-end-ref?   leafref
   +--rw to-link
      +--rw topo-ref?       leafref
      +--rw node-ref?       leafref
      +--rw link-end-ref?   leafref
   +--rw is-allowed?        Boolean
```

2.1.3. TED Information Sources

The model allows each TE topological element to have multiple TE information sources (OSPF-TE, ISIS-TE, BGP-LS, User-Configured, Other). Each information source is associated with a credibility preference to indicate precedence.

The model captures overlay and underlay relationship for TE nodes/ links. For example - in networks where multiple TE Topologies are built hierarchically, this model allows the user to start from a specific topological element in the top most topology and traverse all the way down to the supporting topological elements in the bottom most topology.

2.2. Overlay/Underlay Relationship

```
+--rw node* [te-node-id]
             :
   +--rw te-node-attributes
             :
      +--rw underlay-topology?  leafref {te-topology-hierarchy}?


+--rw link* [source-te-node-id source-te-link-id dest-te-node-id
             dest-te-link-id]
             :
   +--rw te-link-attributes
      +--rw underlay! {te-topology-hierarchy}?
         +--rw underlay-path
                 :
         +--rw underlay-backup-path
                 :
         +--rw underlay-protection-type?   uint16
         +--rw underlay-trail-src
             :
         +--rw underlay-trail-des
             :
```

2.3. Scheduling Parameters

The model allows time scheduling parameters to be specified for each topological element. These parameters allow the provider to present different topological views to the client at different time slots.

```
+--rw schedules* [schedule-id]
|  +--rw schedule-id          uint32
|  +--rw start?               yang:date-and-time
|  +--rw schedule-duration?   string
|  +--rw repeat-interval?     string
```

2.4. Abstract TE Topologies

The model allows the provider to present the network in abstract terms on per client basis and facilitates the notion of "TE Topology as a service". These topologies are typically decoupled from the actual network topology and are supposed to be fully comprehensible by the clients and contain sufficient information for the client path computers to select service paths according to the client policies. The model also allows the client to request changes to the abstract TE Topology that is presented to it and thus manipulate it.

2.5. Open Items

- o Relationship with "generic network topology" model: The generic network topology building blocks are discussed in [YANG-NET-TOPO]. This version of the document does not use any of those building blocks. The authors would like to explore the possibility of doing that in future revisions of this document.

- o Incremental notifications: The model proposed in this version does not cover incremental notifications. The authors intend to add this in future revisions of the document.

3. Tree Structure

```
module: ietf-te-topology
   +--rw te-topologies
   |  +--rw topology* [te-topology-id]
   |  |  +--rw te-topology-id    te-topology-id
   |  |  +--rw topology-types
   |  |  |  +--rw te-topology!
   |  |  +--rw node* [te-node-id]
   |  |  |  +--rw te-node-id            te-node-id
   |  |  |  +--rw te-node-template?     leafref
   |  |  |  +--rw te-node-attributes
   |  |  |     +--rw schedules* [schedule-id]
   |  |  |     |  +--rw schedule-id          uint32
   |  |  |     |  +--rw start?               yang:date-and-time
   |  |  |     |  +--rw schedule-duration?   string
   |  |  |     |  +--rw repeat-interval?     string
   |  |  |     +--rw name?                  inet:domain-name
   |  |  |     +--rw signaling-address*     inet:ip-address
   |  |  |     +--rw flag*                  flag-type
   |  |  |     +--rw is-abstract?           boolean
   |  |  |     +--rw underlay-topology?     leafref
 {te-topology-hierarchy}?
   |  |  |     +--rw te-link* [te-link-id]
   |  |  |     |  +--rw te-link-id         te-link-id
   |  |  |     |  +--rw (stack-level)?
   |  |  |     |     +--:(bundle)
   |  |  |     |     |  +--rw bundled-links
   |  |  |     |     |     +--rw bundled-link* [sequence]
   |  |  |     |     |        +--rw sequence       uint32
   |  |  |     |     |        +--rw te-link-ref?   leafref
   |  |  |     |     +--:(component)
```

```
   |  |  |     |        +--rw component-links
   |  |  |     |           +--rw component-link* [sequence]
   |  |  |     |              +--rw sequence              uint32
   |  |  |     |              +--rw component-link-ref?   leafref
   |  |  |     +--rw connectivity-matrix* [id]
   |  |  |     |  +--rw id            uint32
   |  |  |     |  +--rw from-link
   |  |  |     |  |  +--rw topo-ref?       leafref
   |  |  |     |  |  +--rw node-ref?       leafref
   |  |  |     |  |  +--rw link-end-ref?   leafref
   |  |  |     |  +--rw to-link
   |  |  |     |  |  +--rw topo-ref?       leafref
   |  |  |     |  |  +--rw node-ref?       leafref
   |  |  |     |  |  +--rw link-end-ref?   leafref
   |  |  |     |  +--rw is-allowed?   boolean
   |  |  |     +--rw ted
   |  |  |        +--rw te-router-id-ipv4?    inet:ipv4-address
   |  |  |        +--rw te-router-id-ipv6?    inet:ipv6-address
   |  |  |        +--rw ipv4-local-address* [ipv4-prefix]
   |  |  |        |  +--rw ipv4-prefix    inet:ipv4-prefix
   |  |  |        +--rw ipv6-local-address* [ipv6-prefix]
   |  |  |        |  +--rw ipv6-prefix      inet:ipv6-prefix
   |  |  |        |  +--rw prefix-option?   uint8
   |  |  |        +--rw pcc-capabilities?     pcc-capabilities
   |  |  +--rw link* [source-te-node-id source-te-link-id
                 dest-te-node-id dest-te-link-id]
   |  |     +--rw source-te-node-id     leafref
   |  |     +--rw source-te-link-id     leafref
   |  |     +--rw dest-te-node-id       leafref
   |  |     +--rw dest-te-link-id       leafref
   |  |     +--rw te-link-template?     leafref
   |  |     +--rw te-link-attributes
   |  |        +--rw schedules* [schedule-id]
   |  |        |  +--rw schedule-id          uint32
   |  |        |  +--rw start?               yang:date-and-time
   |  |        |  +--rw schedule-duration?   string
   |  |        |  +--rw repeat-interval?     string
   |  |        +--rw name?          string
   |  |        +--rw flag*          flag-type
   |  |        +--rw is-abstract?   boolean
   |  |        +--rw underlay! {te-topology-hierarchy}?
   |  |        |  +--rw underlay-path
   |  |        |  |  +--rw topology-id?    leafref
   |  |        |  |  +--rw path-element* [path-element-id]
   |  |        |  |     +--rw path-element-id    uint32
   |  |        |  |     +--rw loose?             boolean
   |  |        |  |     +--rw (element-type)?
   |  |        |  |        +--:(numbered-link)
```

```
   |  |        |  |        |  +--rw link-ip-address?  inet:ip-address
   |  |        |  |        +--:(unnumbered-link)
   |  |        |  |        |  +--rw link-node-id?      uint32
   |  |        |  |        |  +--rw te-link-id?        uint32
   |  |        |  |        +--:(node)
   |  |        |  |        |  +--rw te-node-id?        uint32
   |  |        |  |        +--:(label)
   |  |        |  |           +--rw label?             uint32
   |  |        |  +--rw underlay-backup-path
   |  |        |  |  +--rw topology-id?    leafref
   |  |        |  |  +--rw path-element* [path-element-id]
   |  |        |  |     +--rw path-element-id    uint32
   |  |        |  |     +--rw loose?             boolean
   |  |        |  |     +--rw (element-type)?
   |  |        |  |        +--:(numbered-link)
   |  |        |  |        |  +--rw link-ip-address?  inet:ip-address
   |  |        |  |        +--:(unnumbered-link)
   |  |        |  |        |  +--rw link-node-id?      uint32
   |  |        |  |        |  +--rw te-link-id?        uint32
   |  |        |  |        +--:(node)
   |  |        |  |        |  +--rw te-node-id?        uint32
   |  |        |  |        +--:(label)
   |  |        |  |           +--rw label?             uint32
   |  |        |  +--rw underlay-protection-type?   uint16
   |  |        |  +--rw underlay-trail-src
   |  |        |  |  +--rw topo-ref?       leafref
   |  |        |  |  +--rw node-ref?       leafref
   |  |        |  |  +--rw link-end-ref?   leafref
   |  |        |  +--rw underlay-trail-des
   |  |        |     +--rw topo-ref?       leafref
   |  |        |     +--rw node-ref?       leafref
   |  |        |     +--rw link-end-ref?   leafref
   |  |        +--rw ted
   |  |           +--rw admin-status?                    enumeration
   |  |           +--rw oper-status?                     enumeration
   |  |           +--rw area-id?                         binary
   |  |           +--rw performance-metric-throttle
{te-performance-metric}?
   |  |           |  +--rw unidirectional-delay-offset?        uint32
   |  |           |  +--rw measure-interval?                   uint32
   |  |           |  +--rw advertisement-interval?             uint32
   |  |           |  +--rw suppression-interval?               uint32
   |  |           |  +--rw threshold-out
   |  |           |  |  +--rw unidirectional-delay?            uint32
   |  |           |  |  +--rw unidirectional-min-delay?        uint32
   |  |           |  |  +--rw unidirectional-max-delay?        uint32
   |  |           |  |  +--rw unidirectional-delay-variation?  uint32
   |  |           |  |  +--rw unidirectional-packet-loss?   decimal64
```

```
   |  |           |  |  +--rw unidirectional-residual-bandwidth?
decimal64
   |  |           |  |  +--rw unidirectional-available-bandwidth?
decimal64
   |  |           |  |  +--rw unidirectional-utilized-bandwidth?
decimal64
   |  |           |  +--rw threshold-in
   |  |           |  |  +--rw unidirectional-delay?
uint32
   |  |           |  |  +--rw unidirectional-min-delay?
uint32
   |  |           |  |  +--rw unidirectional-max-delay?
uint32
   |  |           |  |  +--rw unidirectional-delay-variation?
uint32
   |  |           |  |  +--rw unidirectional-packet-loss?
decimal64
   |  |           |  |  +--rw unidirectional-residual-bandwidth?
decimal64
   |  |           |  |  +--rw unidirectional-available-bandwidth?
decimal64
   |  |           |  |  +--rw unidirectional-utilized-bandwidth?
decimal64
   |  |           |  +--rw threshold-accelerated-advertisement
   |  |           |     +--rw unidirectional-delay?            uint32
   |  |           |     +--rw unidirectional-min-delay?        uint32
   |  |           |     +--rw unidirectional-max-delay?        uint32
   |  |           |     +--rw unidirectional-delay-variation?  uint32
   |  |           |     +--rw unidirectional-packet-loss?   decimal64
   |  |           |     +--rw unidirectional-residual-bandwidth?
decimal64
   |  |           |     +--rw unidirectional-available-bandwidth?
decimal64
   |  |           |     +--rw unidirectional-utilized-bandwidth?
decimal64
   |  |           +--rw information-source?              enumeration
   |  |           +--rw credibility-preference?               uint16
   |  |           +--rw link-index?                           uint64
   |  |           +--rw administrative-group* [sequence]
   |  |           |  +--rw sequence      uint32
   |  |           |  +--rw ag-element?   uint32
   |  |           +--rw max-link-bandwidth?                decimal64
   |  |           +--rw max-resv-link-bandwidth?           decimal64
   |  |           +--rw unreserved-bandwidth* [priority]
   |  |           |  +--rw priority     uint8
   |  |           |  +--rw bandwidth?   decimal64
   |  |           +--rw te-default-metric?                 uint32
   |  |           +--rw performance-metric {te-performance-metric}?
```

```
   |  |           |  +--rw measurement
   |  |           |  |  +--rw unidirectional-delay?            uint32
   |  |           |  |  +--rw unidirectional-min-delay?        uint32
   |  |           |  |  +--rw unidirectional-max-delay?        uint32
   |  |           |  |  +--rw unidirectional-delay-variation?  uint32
   |  |           |  |  +--rw unidirectional-packet-loss?   decimal64
   |  |           |  |  +--rw unidirectional-residual-bandwidth?
decimal64
   |  |           |  |  +--rw unidirectional-available-bandwidth?
decimal64
   |  |           |  |  +--rw unidirectional-utilized-bandwidth?
decimal64
   |  |           |  +--rw normality
   |  |           |     +--rw unidirectional-delay?
performance-metric-normality
   |  |           |     +--rw unidirectional-min-delay?
performance-metric-normality
   |  |           |     +--rw unidirectional-max-delay?
performance-metric-normality
   |  |           |     +--rw unidirectional-delay-variation?
performance-metric-normality
   |  |           |     +--rw unidirectional-packet-loss?
performance-metric-normality
   |  |           |     +--rw unidirectional-residual-bandwidth?
performance-metric-normality
   |  |           |     +--rw unidirectional-available-bandwidth?
performance-metric-normality
   |  |           |     +--rw unidirectional-utilized-bandwidth?
performance-metric-normality
   |  |           +--rw link-protection-type?
enumeration
   |  |           +--rw interface-switching-capabilities*
[switching-capability]
   |  |           |  +--rw switching-capability
ted:switching-capabilities
   |  |           |  +--rw encoding?
ted:encoding-type
   |  |           |  +--rw max-lsp-bandwidth* [priority]
   |  |           |  |  +--rw priority     uint8
   |  |           |  |  +--rw bandwidth?   decimal64
   |  |           |  +--rw packet-switch-capable
   |  |           |  |  +--rw minimum-lsp-bandwidth?   decimal64
   |  |           |  |  +--rw interface-mtu?           uint16
   |  |           |  +--rw time-division-multiplex-capable
   |  |           |     +--rw minimum-lsp-bandwidth?   decimal64
   |  |           |     +--rw indication?              enumeration
   |  |           +--rw srlg
   |  |           |  +--rw srlg-values* [srlg-value]
```

```
   |  |            |     +--rw srlg-value    uint32
   |  |            +--rw alt-information-sources* [information-source]
   |  |               +--rw information-source             enumeration
   |  |               +--rw credibility-preference?             uint16
   |  |               +--rw link-index?                         uint64
   |  |               +--rw administrative-group* [sequence]
   |  |               |  +--rw sequence      uint32
   |  |               |  +--rw ag-element?   uint32
   |  |               +--rw max-link-bandwidth?              decimal64
   |  |               +--rw max-resv-link-bandwidth?         decimal64
   |  |               +--rw unreserved-bandwidth* [priority]
   |  |               |  +--rw priority     uint8
   |  |               |  +--rw bandwidth?   decimal64
   |  |               +--rw te-default-metric?                  uint32
   |  |               +--rw performance-metric
 {te-performance-metric}?
   |  |               |  +--rw measurement
   |  |               |  |  +--rw unidirectional-delay?         uint32
   |  |               |  |  +--rw unidirectional-min-delay?     uint32
   |  |               |  |  +--rw unidirectional-max-delay?     uint32
   |  |               |  |  +--rw unidirectional-delay-variation?
uint32
   |  |               |  |  +--rw unidirectional-packet-loss?
decimal64
   |  |               |  |  +--rw unidirectional-residual-bandwidth?
decimal64
   |  |               |  |  +--rw unidirectional-available-bandwidth?
decimal64
   |  |               |  |  +--rw unidirectional-utilized-bandwidth?
decimal64
   |  |               |  +--rw normality
   |  |               |     +--rw unidirectional-delay?
performance-metric-normality
   |  |               |     +--rw unidirectional-min-delay?
performance-metric-normality
   |  |               |     +--rw unidirectional-max-delay?
performance-metric-normality
   |  |               |     +--rw unidirectional-delay-variation?
performance-metric-normality
   |  |               |     +--rw unidirectional-packet-loss?
performance-metric-normality
   |  |               |     +--rw unidirectional-residual-bandwidth?
performance-metric-normality
   |  |               |     +--rw unidirectional-available-bandwidth?
performance-metric-normality
   |  |               |     +--rw unidirectional-utilized-bandwidth?
performance-metric-normality
   |  |               +--rw link-protection-type?
```

```
enumeration
   |  |              +--rw interface-switching-capabilities*
[switching-capability]
   |  |              |  +--rw switching-capability
ted:switching-capabilities
   |  |              |  +--rw encoding?
ted:encoding-type
   |  |              |  +--rw max-lsp-bandwidth* [priority]
   |  |              |  |  +--rw priority     uint8
   |  |              |  |  +--rw bandwidth?   decimal64
   |  |              |  +--rw packet-switch-capable
   |  |              |  |  +--rw minimum-lsp-bandwidth?   decimal64
   |  |              |  |  +--rw interface-mtu?           uint16
   |  |              |  +--rw time-division-multiplex-capable
   |  |              |     +--rw minimum-lsp-bandwidth?   decimal64
   |  |              |     +--rw indication?              enumeration
   |  |              +--rw srlg
   |  |                 +--rw srlg-values* [srlg-value]
   |  |                    +--rw srlg-value    uint32
   |  +--rw node-template* [name]
   |  |  +--rw name                 te-template-name
   |  |  +--rw te-node-attributes
   |  |     +--rw schedules* [schedule-id]
   |  |     |  +--rw schedule-id          uint32
   |  |     |  +--rw start?               yang:date-and-time
   |  |     |  +--rw schedule-duration?   string
   |  |     |  +--rw repeat-interval?     string
   |  |     +--rw name?                  inet:domain-name
   |  |     +--rw signaling-address*     inet:ip-address
   |  |     +--rw flag*                  flag-type
   |  |     +--rw is-abstract?           boolean
   |  |     +--rw underlay-topology?     leafref
{te-topology-hierarchy}?
   |  |     +--rw te-link* [te-link-id]
   |  |     |  +--rw te-link-id         te-link-id
   |  |     |  +--rw (stack-level)?
   |  |     |     +--:(bundle)
   |  |     |     |  +--rw bundled-links
   |  |     |     |     +--rw bundled-link* [sequence]
   |  |     |     |        +--rw sequence       uint32
   |  |     |     |        +--rw te-link-ref?   leafref
   |  |     |     +--:(component)
   |  |     |        +--rw component-links
   |  |     |           +--rw component-link* [sequence]
   |  |     |              +--rw sequence              uint32
   |  |     |              +--rw component-link-ref?   leafref
   |  |     +--rw connectivity-matrix* [id]
   |  |     |  +--rw id            uint32
```

```
   |  |     |  +--rw from-link
   |  |     |  |  +--rw topo-ref?       leafref
   |  |     |  |  +--rw node-ref?       leafref
   |  |     |  |  +--rw link-end-ref?   leafref
   |  |     |  +--rw to-link
   |  |     |  |  +--rw topo-ref?       leafref
   |  |     |  |  +--rw node-ref?       leafref
   |  |     |  |  +--rw link-end-ref?   leafref
   |  |     |  +--rw is-allowed?   boolean
   |  |     +--rw ted
   |  |        +--rw te-router-id-ipv4?    inet:ipv4-address
   |  |        +--rw te-router-id-ipv6?    inet:ipv6-address
   |  |        +--rw ipv4-local-address* [ipv4-prefix]
   |  |        |  +--rw ipv4-prefix    inet:ipv4-prefix
   |  |        +--rw ipv6-local-address* [ipv6-prefix]
   |  |        |  +--rw ipv6-prefix      inet:ipv6-prefix
   |  |        |  +--rw prefix-option?   uint8
   |  |        +--rw pcc-capabilities?     pcc-capabilities
   |  +--rw link-template* [name]
   |     +--rw name                  te-template-name
   |     +--rw te-link-attributes
   |        +--rw schedules* [schedule-id]
   |        |  +--rw schedule-id          uint32
   |        |  +--rw start?               yang:date-and-time
   |        |  +--rw schedule-duration?   string
   |        |  +--rw repeat-interval?     string
   |        +--rw name?          string
   |        +--rw flag*          flag-type
   |        +--rw is-abstract?   boolean
   |        +--rw underlay! {te-topology-hierarchy}?
   |        |  +--rw underlay-path
   |        |  |  +--rw topology-id?    leafref
   |        |  |  +--rw path-element* [path-element-id]
   |        |  |     +--rw path-element-id    uint32
   |        |  |     +--rw loose?             boolean
   |        |  |     +--rw (element-type)?
   |        |  |        +--:(numbered-link)
   |        |  |        |  +--rw link-ip-address?   inet:ip-address
   |        |  |        +--:(unnumbered-link)
   |        |  |        |  +--rw link-node-id?      uint32
   |        |  |        |  +--rw te-link-id?        uint32
   |        |  |        +--:(node)
   |        |  |        |  +--rw te-node-id?        uint32
   |        |  |        +--:(label)
   |        |  |           +--rw label?             uint32
   |        |  +--rw underlay-backup-path
   |        |  |  +--rw topology-id?    leafref
   |        |  |  +--rw path-element* [path-element-id]
```

```
   |         |  |     +--rw path-element-id    uint32
   |         |  |     +--rw loose?             boolean
   |         |  |     +--rw (element-type)?
   |         |  |        +--:(numbered-link)
   |         |  |        |  +--rw link-ip-address?   inet:ip-address
   |         |  |        +--:(unnumbered-link)
   |         |  |        |  +--rw link-node-id?      uint32
   |         |  |        |  +--rw te-link-id?        uint32
   |         |  |        +--:(node)
   |         |  |        |  +--rw te-node-id?        uint32
   |         |  |        +--:(label)
   |         |  |           +--rw label?             uint32
   |         |  +--rw underlay-protection-type?   uint16
   |         |  +--rw underlay-trail-src
   |         |  |  +--rw topo-ref?       leafref
   |         |  |  +--rw node-ref?       leafref
   |         |  |  +--rw link-end-ref?   leafref
   |         |  +--rw underlay-trail-des
   |         |     +--rw topo-ref?       leafref
   |         |     +--rw node-ref?       leafref
   |         |     +--rw link-end-ref?   leafref
   |         +--rw ted
   |            +--rw admin-status?                         enumeration
   |            +--rw oper-status?                          enumeration
   |            +--rw area-id?                              binary
   |            +--rw performance-metric-throttle
{te-performance-metric}?
   |            |  +--rw unidirectional-delay-offset?           uint32
   |            |  +--rw measure-interval?                      uint32
   |            |  +--rw advertisement-interval?                uint32
   |            |  +--rw suppression-interval?                  uint32
   |            |  +--rw threshold-out
   |            |  |  +--rw unidirectional-delay?               uint32
   |            |  |  +--rw unidirectional-min-delay?           uint32
   |            |  |  +--rw unidirectional-max-delay?           uint32
   |            |  |  +--rw unidirectional-delay-variation?     uint32
   |            |  |  +--rw unidirectional-packet-loss?      decimal64
   |            |  |  +--rw unidirectional-residual-bandwidth?
decimal64
   |            |  |  +--rw unidirectional-available-bandwidth?
decimal64
   |            |  |  +--rw unidirectional-utilized-bandwidth?
decimal64
   |            |  +--rw threshold-in
   |            |  |  +--rw unidirectional-delay?               uint32
   |            |  |  +--rw unidirectional-min-delay?           uint32
   |            |  |  +--rw unidirectional-max-delay?           uint32
   |            |  |  +--rw unidirectional-delay-variation?     uint32
```

```
   |           |  |  +--rw unidirectional-packet-loss?      decimal64
   |           |  |  +--rw unidirectional-residual-bandwidth?
decimal64
   |           |  |  +--rw unidirectional-available-bandwidth?
decimal64
   |           |  |  +--rw unidirectional-utilized-bandwidth?
decimal64
   |           |  +--rw threshold-accelerated-advertisement
   |           |     +--rw unidirectional-delay?               uint32
   |           |     +--rw unidirectional-min-delay?           uint32
   |           |     +--rw unidirectional-max-delay?           uint32
   |           |     +--rw unidirectional-delay-variation?     uint32
   |           |     +--rw unidirectional-packet-loss?      decimal64
   |           |     +--rw unidirectional-residual-bandwidth?
decimal64
   |           |     +--rw unidirectional-available-bandwidth?
decimal64
   |           |     +--rw unidirectional-utilized-bandwidth?
decimal64
   |           +--rw information-source?                 enumeration
   |           +--rw credibility-preference?             uint16
   |           +--rw link-index?                         uint64
   |           +--rw administrative-group* [sequence]
   |           |  +--rw sequence      uint32
   |           |  +--rw ag-element?   uint32
   |           +--rw max-link-bandwidth?                 decimal64
   |           +--rw max-resv-link-bandwidth?            decimal64
   |           +--rw unreserved-bandwidth* [priority]
   |           |  +--rw priority     uint8
   |           |  +--rw bandwidth?   decimal64
   |           +--rw te-default-metric?                  uint32
   |           +--rw performance-metric {te-performance-metric}?
   |           |  +--rw measurement
   |           |  |  +--rw unidirectional-delay?               uint32
   |           |  |  +--rw unidirectional-min-delay?           uint32
   |           |  |  +--rw unidirectional-max-delay?           uint32
   |           |  |  +--rw unidirectional-delay-variation?     uint32
   |           |  |  +--rw unidirectional-packet-loss?      decimal64
   |           |  |  +--rw unidirectional-residual-bandwidth?
decimal64
   |           |  |  +--rw unidirectional-available-bandwidth?
decimal64
   |           |  |  +--rw unidirectional-utilized-bandwidth?
decimal64
   |           |  +--rw normality
   |           |     +--rw unidirectional-delay?
performance-metric-normality
   |           |     +--rw unidirectional-min-delay?
```

```
performance-metric-normality
   |           |     +--rw unidirectional-max-delay?
performance-metric-normality
   |           |     +--rw unidirectional-delay-variation?
performance-metric-normality
   |           |     +--rw unidirectional-packet-loss?
performance-metric-normality
   |           |     +--rw unidirectional-residual-bandwidth?
performance-metric-normality
   |           |     +--rw unidirectional-available-bandwidth?
performance-metric-normality
   |           |     +--rw unidirectional-utilized-bandwidth?
performance-metric-normality
   |           +--rw link-protection-type?              enumeration
   |           +--rw interface-switching-capabilities*
[switching-capability]
   |           |  +--rw switching-capability
ted:switching-capabilities
   |           |  +--rw encoding?                   ted:encoding-type
   |           |  +--rw max-lsp-bandwidth* [priority]
   |           |  |  +--rw priority     uint8
   |           |  |  +--rw bandwidth?   decimal64
   |           |  +--rw packet-switch-capable
   |           |  |  +--rw minimum-lsp-bandwidth?   decimal64
   |           |  |  +--rw interface-mtu?           uint16
   |           |  +--rw time-division-multiplex-capable
   |           |     +--rw minimum-lsp-bandwidth?   decimal64
   |           |     +--rw indication?              enumeration
   |           +--rw srlg
   |           |  +--rw srlg-values* [srlg-value]
   |           |     +--rw srlg-value    uint32
   |           +--rw alt-information-sources* [information-source]
   |              +--rw information-source              enumeration
   |              +--rw credibility-preference?           uint16
   |              +--rw link-index?                       uint64
   |              +--rw administrative-group* [sequence]
   |              |  +--rw sequence      uint32
   |              |  +--rw ag-element?   uint32
   |              +--rw max-link-bandwidth?               decimal64
   |              +--rw max-resv-link-bandwidth?          decimal64
   |              +--rw unreserved-bandwidth* [priority]
   |              |  +--rw priority     uint8
   |              |  +--rw bandwidth?   decimal64
   |              +--rw te-default-metric?                   uint32
   |              +--rw performance-metric {te-performance-metric}?
   |              |  +--rw measurement
   |              |  |  +--rw unidirectional-delay?           uint32
   |              |  |  +--rw unidirectional-min-delay?       uint32
```

```
   |              |  |  +--rw unidirectional-max-delay?        uint32
   |              |  |  +--rw unidirectional-delay-variation?  uint32
   |              |  |  +--rw unidirectional-packet-loss?   decimal64
   |              |  |  +--rw unidirectional-residual-bandwidth?
decimal64
   |              |  |  +--rw unidirectional-available-bandwidth?
decimal64
   |              |  |  +--rw unidirectional-utilized-bandwidth?
decimal64
   |              |  +--rw normality
   |              |     +--rw unidirectional-delay?
performance-metric-normality
   |              |     +--rw unidirectional-min-delay?
performance-metric-normality
   |              |     +--rw unidirectional-max-delay?
performance-metric-normality
   |              |     +--rw unidirectional-delay-variation?
performance-metric-normality
   |              |     +--rw unidirectional-packet-loss?
performance-metric-normality
   |              |     +--rw unidirectional-residual-bandwidth?
performance-metric-normality
   |              |     +--rw unidirectional-available-bandwidth?
performance-metric-normality
   |              |     +--rw unidirectional-utilized-bandwidth?
performance-metric-normality
   |              +--rw link-protection-type?             enumeration
   |              +--rw interface-switching-capabilities*
[switching-capability]
   |              |  +--rw switching-capability
ted:switching-capabilities
   |              |  +--rw encoding?                ted:encoding-type
   |              |  +--rw max-lsp-bandwidth* [priority]
   |              |  |  +--rw priority     uint8
   |              |  |  +--rw bandwidth?   decimal64
   |              |  +--rw packet-switch-capable
   |              |  |  +--rw minimum-lsp-bandwidth?   decimal64
   |              |  |  +--rw interface-mtu?           uint16
   |              |  +--rw time-division-multiplex-capable
   |              |     +--rw minimum-lsp-bandwidth?   decimal64
   |              |     +--rw indication?              enumeration
   |              +--rw srlg
   |                 +--rw srlg-values* [srlg-value]
   |                    +--rw srlg-value    uint32
   +--ro te-topologies-state
      +--ro topology* [te-topology-id]
         +--ro te-topology-id     te-topology-id
         +--ro server-provided?   boolean
```

```
         +--ro topology-types
         |  +--ro te-topology!
         +--ro node* [te-node-id]
         |  +--ro te-node-id                  te-node-id
         |  +--ro te-node-template?           leafref
         |  +--ro te-node-attributes
         |  |  +--ro schedules* [schedule-id]
         |  |  |  +--ro schedule-id          uint32
         |  |  |  +--ro start?               yang:date-and-time
         |  |  |  +--ro schedule-duration?   string
         |  |  |  +--ro repeat-interval?     string
         |  |  +--ro name?                  inet:domain-name
         |  |  +--ro signaling-address*     inet:ip-address
         |  |  +--ro flag*                  flag-type
         |  |  +--ro is-abstract?           boolean
         |  |  +--ro underlay-topology?     leafref
{te-topology-hierarchy}?
         |  |  +--ro te-link* [te-link-id]
         |  |  |  +--ro te-link-id         te-link-id
         |  |  |  +--ro (stack-level)?
         |  |  |     +--:(bundle)
         |  |  |     |  +--ro bundled-links
         |  |  |     |     +--ro bundled-link* [sequence]
         |  |  |     |        +--ro sequence       uint32
         |  |  |     |        +--ro te-link-ref?   leafref
         |  |  |     +--:(component)
         |  |  |        +--ro component-links
         |  |  |           +--ro component-link* [sequence]
         |  |  |              +--ro sequence               uint32
         |  |  |              +--ro component-link-ref?   leafref
         |  |  +--ro connectivity-matrix* [id]
         |  |  |  +--ro id            uint32
         |  |  |  +--ro from-link
         |  |  |  |  +--ro topo-ref?       leafref
         |  |  |  |  +--ro node-ref?       leafref
         |  |  |  |  +--ro link-end-ref?   leafref
         |  |  |  +--ro to-link
         |  |  |  |  +--ro topo-ref?       leafref
         |  |  |  |  +--ro node-ref?       leafref
         |  |  |  |  +--ro link-end-ref?   leafref
         |  |  |  +--ro is-allowed?   boolean
         |  |  +--ro ted
         |  |     +--ro te-router-id-ipv4?    inet:ipv4-address
         |  |     +--ro te-router-id-ipv6?    inet:ipv6-address
         |  |     +--ro ipv4-local-address* [ipv4-prefix]
         |  |     |  +--ro ipv4-prefix    inet:ipv4-prefix
         |  |     +--ro ipv6-local-address* [ipv6-prefix]
         |  |     |  +--ro ipv6-prefix      inet:ipv6-prefix
```

```
     |  |     |  +--ro prefix-option?   uint8
     |  |     +--ro pcc-capabilities?     pcc-capabilities
     |  +--ro te-node-state-attributes
     |     +--ro information-source?       enumeration
     |     +--ro credibility-preference?   uint16
     +--ro link* [source-te-node-id source-te-link-id
                  dest-te-node-id dest-te-link-id]
        +--ro source-te-node-id           leafref
        +--ro source-te-link-id           leafref
        +--ro dest-te-node-id             leafref
        +--ro dest-te-link-id             leafref
        +--ro te-link-template?           leafref
        +--ro te-link-attributes
        |  +--ro schedules* [schedule-id]
        |  |  +--ro schedule-id          uint32
        |  |  +--ro start?               yang:date-and-time
        |  |  +--ro schedule-duration?   string
        |  |  +--ro repeat-interval?     string
        |  +--ro name?          string
        |  +--ro flag*          flag-type
        |  +--ro is-abstract?   boolean
        |  +--ro underlay! {te-topology-hierarchy}?
        |  |  +--ro underlay-path
        |  |  |  +--ro topology-id?    leafref
        |  |  |  +--ro path-element* [path-element-id]
        |  |  |     +--ro path-element-id    uint32
        |  |  |     +--ro loose?             boolean
        |  |  |     +--ro (element-type)?
        |  |  |        +--:(numbered-link)
        |  |  |        |  +--ro link-ip-address?  inet:ip-address
        |  |  |        +--:(unnumbered-link)
        |  |  |        |  +--ro link-node-id?      uint32
        |  |  |        |  +--ro te-link-id?        uint32
        |  |  |        +--:(node)
        |  |  |        |  +--ro te-node-id?        uint32
        |  |  |        +--:(label)
        |  |  |           +--ro label?            uint32
        |  |  +--ro underlay-backup-path
        |  |  |  +--ro topology-id?    leafref
        |  |  |  +--ro path-element* [path-element-id]
        |  |  |     +--ro path-element-id    uint32
        |  |  |     +--ro loose?             boolean
        |  |  |     +--ro (element-type)?
        |  |  |        +--:(numbered-link)
        |  |  |        |  +--ro link-ip-address?  inet:ip-address
        |  |  |        +--:(unnumbered-link)
        |  |  |        |  +--ro link-node-id?      uint32
        |  |  |        |  +--ro te-link-id?        uint32
```

```
            |  |  |        +--:(node)
            |  |  |        |  +--ro te-node-id?        uint32
            |  |  |        +--:(label)
            |  |  |           +--ro label?             uint32
            |  |  +--ro underlay-protection-type?   uint16
            |  |  +--ro underlay-trail-src
            |  |  |  +--ro topo-ref?       leafref
            |  |  |  +--ro node-ref?       leafref
            |  |  |  +--ro link-end-ref?   leafref
            |  |  +--ro underlay-trail-des
            |  |  |  +--ro topo-ref?       leafref
            |  |  |  +--ro node-ref?       leafref
            |  |  |  +--ro link-end-ref?   leafref
            |  |  +--ro dynamic?                    boolean
            |  |  +--ro committed?                  boolean
            |  +--ro ted
            |     +--ro admin-status?                     enumeration
            |     +--ro oper-status?                      enumeration
            |     +--ro area-id?                          binary
            |     +--ro performance-metric-throttle {te-performance-metric}?
            |     |  +--ro unidirectional-delay-offset?        uint32
            |     |  +--ro measure-interval?                   uint32
            |     |  +--ro advertisement-interval?             uint32
            |     |  +--ro suppression-interval?               uint32
            |     |  +--ro threshold-out
            |     |  |  +--ro unidirectional-delay?            uint32
            |     |  |  +--ro unidirectional-min-delay?        uint32
            |     |  |  +--ro unidirectional-max-delay?        uint32
            |     |  |  +--ro unidirectional-delay-variation?  uint32
            |     |  |  +--ro unidirectional-packet-loss?
decimal64
            |     |  |  +--ro unidirectional-residual-bandwidth?
decimal64
            |     |  |  +--ro unidirectional-available-bandwidth?
decimal64
            |     |  |  +--ro unidirectional-utilized-bandwidth?
decimal64
            |     |  +--ro threshold-in
            |     |  |  +--ro unidirectional-delay?            uint32
            |     |  |  +--ro unidirectional-min-delay?        uint32
            |     |  |  +--ro unidirectional-max-delay?        uint32
            |     |  |  +--ro unidirectional-delay-variation?  uint32
            |     |  |  +--ro unidirectional-packet-loss?   decimal64
            |     |  |  +--ro unidirectional-residual-bandwidth?
decimal64
            |     |  |  +--ro unidirectional-available-bandwidth?
decimal64
            |     |  |  +--ro unidirectional-utilized-bandwidth?
```

```
decimal64
             |     |  +--ro threshold-accelerated-advertisement
             |     |     +--ro unidirectional-delay?           uint32
             |     |     +--ro unidirectional-min-delay?       uint32
             |     |     +--ro unidirectional-max-delay?       uint32
             |     |     +--ro unidirectional-delay-variation?  uint32
             |     |     +--ro unidirectional-packet-loss?   decimal64
             |     |     +--ro unidirectional-residual-bandwidth?
decimal64
             |     |     +--ro unidirectional-available-bandwidth?
decimal64
             |     |     +--ro unidirectional-utilized-bandwidth?
decimal64
             |     +--ro information-source?              enumeration
             |     +--ro credibility-preference?          uint16
             |     +--ro link-index?                      uint64
             |     +--ro administrative-group* [sequence]
             |     |  +--ro sequence      uint32
             |     |  +--ro ag-element?   uint32
             |     +--ro max-link-bandwidth?                 decimal64
             |     +--ro max-resv-link-bandwidth?            decimal64
             |     +--ro unreserved-bandwidth* [priority]
             |     |  +--ro priority     uint8
             |     |  +--ro bandwidth?   decimal64
             |     +--ro te-default-metric?                  uint32
             |     +--ro performance-metric {te-performance-metric}?
             |     |  +--ro measurement
             |     |  |  +--ro unidirectional-delay?           uint32
             |     |  |  +--ro unidirectional-min-delay?       uint32
             |     |  |  +--ro unidirectional-max-delay?       uint32
             |     |  |  +--ro unidirectional-delay-variation?  uint32
             |     |  |  +--ro unidirectional-packet-loss?   decimal64
             |     |  |  +--ro unidirectional-residual-bandwidth?
decimal64
             |     |  |  +--ro unidirectional-available-bandwidth?
decimal64
             |     |  |  +--ro unidirectional-utilized-bandwidth?
decimal64
             |     |  +--ro normality
             |     |     +--ro unidirectional-delay?
performance-metric-normality
             |     |     +--ro unidirectional-min-delay?
performance-metric-normality
             |     |     +--ro unidirectional-max-delay?
performance-metric-normality
             |     |     +--ro unidirectional-delay-variation?
performance-metric-normality
             |     |     +--ro unidirectional-packet-loss?
```

```
performance-metric-normality
            |     |     +--ro unidirectional-residual-bandwidth?
performance-metric-normality
            |     |     +--ro unidirectional-available-bandwidth?
performance-metric-normality
            |     |     +--ro unidirectional-utilized-bandwidth?
performance-metric-normality
            |     +--ro link-protection-type?            enumeration
            |     +--ro interface-switching-capabilities*
[switching-capability]
            |     |  +--ro switching-capability
ted:switching-capabilities
            |     |  +--ro encoding?
ted:encoding-type
            |     |  +--ro max-lsp-bandwidth* [priority]
            |     |  |  +--ro priority     uint8
            |     |  |  +--ro bandwidth?   decimal64
            |     |  +--ro packet-switch-capable
            |     |  |  +--ro minimum-lsp-bandwidth?   decimal64
            |     |  |  +--ro interface-mtu?           uint16
            |     |  +--ro time-division-multiplex-capable
            |     |     +--ro minimum-lsp-bandwidth?   decimal64
            |     |     +--ro indication?              enumeration
            |     +--ro srlg
            |     |  +--ro srlg-values* [srlg-value]
            |     |     +--ro srlg-value    uint32
            |     +--ro alt-information-sources* [information-source]
            |        +--ro information-source            enumeration
            |        +--ro credibility-preference?            uint16
            |        +--ro link-index?                        uint64
            |        +--ro administrative-group* [sequence]
            |        |  +--ro sequence      uint32
            |        |  +--ro ag-element?   uint32
            |        +--ro max-link-bandwidth?              decimal64
            |        +--ro max-resv-link-bandwidth?         decimal64
            |        +--ro unreserved-bandwidth* [priority]
            |        |  +--ro priority     uint8
            |        |  +--ro bandwidth?   decimal64
            |        +--ro te-default-metric?                 uint32
            |        +--ro performance-metric
{te-performance-metric}?
            |        |  +--ro measurement
            |        |  |  +--ro unidirectional-delay?         uint32
            |        |  |  +--ro unidirectional-min-delay?     uint32
            |        |  |  +--ro unidirectional-max-delay?     uint32
            |        |  |  +--ro unidirectional-delay-variation?
uint32
            |        |  |  +--ro unidirectional-packet-loss?
```

```
decimal64
            |        |  |  +--ro unidirectional-residual-bandwidth?
decimal64
            |        |  |  +--ro unidirectional-available-bandwidth?
decimal64
            |        |  |  +--ro unidirectional-utilized-bandwidth?
decimal64
            |        |  +--ro normality
            |        |     +--ro unidirectional-delay?
performance-metric-normality
            |        |     +--ro unidirectional-min-delay?
performance-metric-normality
            |        |     +--ro unidirectional-max-delay?
performance-metric-normality
            |        |     +--ro unidirectional-delay-variation?
performance-metric-normality
            |        |     +--ro unidirectional-packet-loss?
performance-metric-normality
            |        |     +--ro unidirectional-residual-bandwidth?
performance-metric-normality
            |        |     +--ro unidirectional-available-bandwidth?
performance-metric-normality
            |        |     +--ro unidirectional-utilized-bandwidth?
performance-metric-normality
            |        +--ro link-protection-type?          enumeration
            |        +--ro interface-switching-capabilities*
[switching-capability]
            |        |  +--ro switching-capability
ted:switching-capabilities
            |        |  +--ro encoding?             ted:encoding-type
            |        |  +--ro max-lsp-bandwidth* [priority]
            |        |  |  +--ro priority     uint8
            |        |  |  +--ro bandwidth?   decimal64
            |        |  +--ro packet-switch-capable
            |        |  |  +--ro minimum-lsp-bandwidth?   decimal64
            |        |  |  +--ro interface-mtu?           uint16
            |        |  +--ro time-division-multiplex-capable
            |        |     +--ro minimum-lsp-bandwidth?   decimal64
            |        |     +--ro indication?              enumeration
            |        +--ro srlg
            |           +--ro srlg-values* [srlg-value]
            |              +--ro srlg-value    uint32
            +--ro te-link-state-attributes
               +--ro information-source?       enumeration
               +--ro credibility-preference?   uint16
notifications:
   +---n te-node-event
   |  +--ro event-type?           te-topology-event-type
```

```
   |  +--ro topo-ref?             leafref
   |  +--ro node-ref?             leafref
   |  +--ro te-topology!
   |  +--ro te-node-attributes
   |     +--ro schedules* [schedule-id]
   |     |  +--ro schedule-id          uint32
   |     |  +--ro start?               yang:date-and-time
   |     |  +--ro schedule-duration?   string
   |     |  +--ro repeat-interval?     string
   |     +--ro name?                  inet:domain-name
   |     +--ro signaling-address*     inet:ip-address
   |     +--ro flag*                  flag-type
   |     +--ro is-abstract?           boolean
   |     +--ro underlay-topology?    leafref {te-topology-hierarchy}?
   |     +--ro te-link* [te-link-id]
   |     |  +--ro te-link-id         te-link-id
   |     |  +--ro (stack-level)?
   |     |     +--:(bundle)
   |     |     |  +--ro bundled-links
   |     |     |     +--ro bundled-link* [sequence]
   |     |     |        +--ro sequence       uint32
   |     |     |        +--ro te-link-ref?   leafref
   |     |     +--:(component)
   |     |        +--ro component-links
   |     |           +--ro component-link* [sequence]
   |     |              +--ro sequence             uint32
   |     |              +--ro component-link-ref?   leafref
   |     +--ro connectivity-matrix* [id]
   |     |  +--ro id            uint32
   |     |  +--ro from-link
   |     |  |  +--ro topo-ref?       leafref
   |     |  |  +--ro node-ref?       leafref
   |     |  |  +--ro link-end-ref?   leafref
   |     |  +--ro to-link
   |     |  |  +--ro topo-ref?       leafref
   |     |  |  +--ro node-ref?       leafref
   |     |  |  +--ro link-end-ref?   leafref
   |     |  +--ro is-allowed?   boolean
   |     +--ro ted
   |        +--ro te-router-id-ipv4?    inet:ipv4-address
   |        +--ro te-router-id-ipv6?    inet:ipv6-address
   |        +--ro ipv4-local-address* [ipv4-prefix]
   |        |  +--ro ipv4-prefix    inet:ipv4-prefix
   |        +--ro ipv6-local-address* [ipv6-prefix]
   |        |  +--ro ipv6-prefix      inet:ipv6-prefix
   |        |  +--ro prefix-option?   uint8
   |        +--ro pcc-capabilities?     pcc-capabilities
   +---n te-link-event
```

```
      +--ro event-type?              te-topology-event-type
      +--ro topo-ref?                leafref
      +--ro source-te-node-id-ref?   leafref
      +--ro source-te-link-id-ref?   leafref
      +--ro dest-te-node-id-ref?     leafref
      +--ro dest-te-link-id-ref?     leafref
      +--ro te-topology!
      +--ro te-link-attributes
         +--ro schedules* [schedule-id]
         |  +--ro schedule-id          uint32
         |  +--ro start?               yang:date-and-time
         |  +--ro schedule-duration?   string
         |  +--ro repeat-interval?     string
         +--ro name?          string
         +--ro flag*          flag-type
         +--ro is-abstract?   boolean
         +--ro underlay! {te-topology-hierarchy}?
         |  +--ro underlay-path
         |  |  +--ro topology-id?    leafref
         |  |  +--ro path-element* [path-element-id]
         |  |     +--ro path-element-id    uint32
         |  |     +--ro loose?             boolean
         |  |     +--ro (element-type)?
         |  |        +--:(numbered-link)
         |  |        |  +--ro link-ip-address?   inet:ip-address
         |  |        +--:(unnumbered-link)
         |  |        |  +--ro link-node-id?      uint32
         |  |        |  +--ro te-link-id?        uint32
         |  |        +--:(node)
         |  |        |  +--ro te-node-id?        uint32
         |  |        +--:(label)
         |  |           +--ro label?             uint32
         |  +--ro underlay-backup-path
         |  |  +--ro topology-id?    leafref
         |  |  +--ro path-element* [path-element-id]
         |  |     +--ro path-element-id    uint32
         |  |     +--ro loose?             boolean
         |  |     +--ro (element-type)?
         |  |        +--:(numbered-link)
         |  |        |  +--ro link-ip-address?   inet:ip-address
         |  |        +--:(unnumbered-link)
         |  |        |  +--ro link-node-id?      uint32
         |  |        |  +--ro te-link-id?        uint32
         |  |        +--:(node)
         |  |        |  +--ro te-node-id?        uint32
         |  |        +--:(label)
         |  |           +--ro label?             uint32
         |  +--ro underlay-protection-type?   uint16
```

```
         |  +--ro underlay-trail-src
         |  |  +--ro topo-ref?       leafref
         |  |  +--ro node-ref?       leafref
         |  |  +--ro link-end-ref?   leafref
         |  +--ro underlay-trail-des
         |  |  +--ro topo-ref?       leafref
         |  |  +--ro node-ref?       leafref
         |  |  +--ro link-end-ref?   leafref
         |  +--ro dynamic?                    boolean
         |  +--ro committed?                  boolean
         +--ro ted
            +--ro admin-status?                       enumeration
            +--ro oper-status?                        enumeration
            +--ro area-id?                            binary
            +--ro performance-metric-throttle{te-performance-metric}?
            |  +--ro unidirectional-delay-offset?           uint32
            |  +--ro measure-interval?                      uint32
            |  +--ro advertisement-interval?                uint32
            |  +--ro suppression-interval?                  uint32
            |  +--ro threshold-out
            |  |  +--ro unidirectional-delay?                 uint32
            |  |  +--ro unidirectional-min-delay?             uint32
            |  |  +--ro unidirectional-max-delay?             uint32
            |  |  +--ro unidirectional-delay-variation?       uint32
            |  |  +--ro unidirectional-packet-loss?
decimal64
            |  |  +--ro unidirectional-residual-bandwidth?
decimal64
            |  |  +--ro unidirectional-available-bandwidth?
decimal64
            |  |  +--ro unidirectional-utilized-bandwidth?
decimal64
            |  +--ro threshold-in
            |  |  +--ro unidirectional-delay?                 uint32
            |  |  +--ro unidirectional-min-delay?             uint32
            |  |  +--ro unidirectional-max-delay?             uint32
            |  |  +--ro unidirectional-delay-variation?       uint32
            |  |  +--ro unidirectional-packet-loss?         decimal64
            |  |  +--ro unidirectional-residual-bandwidth?  decimal64
            |  |  +--ro unidirectional-available-bandwidth? decimal64
            |  |  +--ro unidirectional-utilized-bandwidth?  decimal64
            |  +--ro threshold-accelerated-advertisement
            |     +--ro unidirectional-delay?                 uint32
            |     +--ro unidirectional-min-delay?             uint32
            |     +--ro unidirectional-max-delay?             uint32
            |     +--ro unidirectional-delay-variation?       uint32
            |     +--ro unidirectional-packet-loss?         decimal64
            |     +--ro unidirectional-residual-bandwidth?  decimal64
```

```
            |     +--ro unidirectional-available-bandwidth? decimal64
            |     +--ro unidirectional-utilized-bandwidth?  decimal64
            +--ro information-source?                 enumeration
            +--ro credibility-preference?             uint16
            +--ro link-index?                         uint64
            +--ro administrative-group* [sequence]
            |  +--ro sequence      uint32
            |  +--ro ag-element?   uint32
            +--ro max-link-bandwidth?                 decimal64
            +--ro max-resv-link-bandwidth?            decimal64
            +--ro unreserved-bandwidth* [priority]
            |  +--ro priority     uint8
            |  +--ro bandwidth?   decimal64
            +--ro te-default-metric?                  uint32
            +--ro performance-metric {te-performance-metric}?
            |  +--ro measurement
            |  |  +--ro unidirectional-delay?                 uint32
            |  |  +--ro unidirectional-min-delay?             uint32
            |  |  +--ro unidirectional-max-delay?             uint32
            |  |  +--ro unidirectional-delay-variation?       uint32
            |  |  +--ro unidirectional-packet-loss?         decimal64
            |  |  +--ro unidirectional-residual-bandwidth?  decimal64
            |  |  +--ro unidirectional-available-bandwidth? decimal64
            |  |  +--ro unidirectional-utilized-bandwidth?  decimal64
            |  +--ro normality
            |     +--ro unidirectional-delay?
performance-metric-normality
            |     +--ro unidirectional-min-delay?
performance-metric-normality
            |     +--ro unidirectional-max-delay?
performance-metric-normality
            |     +--ro unidirectional-delay-variation?
performance-metric-normality
            |     +--ro unidirectional-packet-loss?
performance-metric-normality
            |     +--ro unidirectional-residual-bandwidth?
performance-metric-normality
            |     +--ro unidirectional-available-bandwidth?
performance-metric-normality
            |     +--ro unidirectional-utilized-bandwidth?
performance-metric-normality
            +--ro link-protection-type?               enumeration
            +--ro interface-switching-capabilities*
[switching-capability]
            |  +--ro switching-capability  ted:switching-capabilities
            |  +--ro encoding?             ted:encoding-type
            |  +--ro max-lsp-bandwidth* [priority]
            |  |  +--ro priority     uint8
```

```
            |  |  +--ro bandwidth?   decimal64
            |  +--ro packet-switch-capable
            |  |  +--ro minimum-lsp-bandwidth?   decimal64
            |  |  +--ro interface-mtu?           uint16
            |  +--ro time-division-multiplex-capable
            |     +--ro minimum-lsp-bandwidth?   decimal64
            |     +--ro indication?              enumeration
            +--ro srlg
            |  +--ro srlg-values* [srlg-value]
            |     +--ro srlg-value    uint32
            +--ro alt-information-sources* [information-source]
               +--ro information-source                 enumeration
               +--ro credibility-preference?            uint16
               +--ro link-index?                        uint64
               +--ro administrative-group* [sequence]
               |  +--ro sequence      uint32
               |  +--ro ag-element?   uint32
               +--ro max-link-bandwidth?                decimal64
               +--ro max-resv-link-bandwidth?           decimal64
               +--ro unreserved-bandwidth* [priority]
               |  +--ro priority     uint8
               |  +--ro bandwidth?   decimal64
               +--ro te-default-metric?                 uint32
               +--ro performance-metric {te-performance-metric}?
               |  +--ro measurement
               |  |  +--ro unidirectional-delay?              uint32
               |  |  +--ro unidirectional-min-delay?          uint32
               |  |  +--ro unidirectional-max-delay?          uint32
               |  |  +--ro unidirectional-delay-variation?    uint32
               |  |  +--ro unidirectional-packet-loss?     decimal64
               |  |  +--ro unidirectional-residual-bandwidth?
decimal64
               |  |  +--ro unidirectional-available-bandwidth?
decimal64
               |  |  +--ro unidirectional-utilized-bandwidth?
decimal64
               |  +--ro normality
               |     +--ro unidirectional-delay?
performance-metric-normality
               |     +--ro unidirectional-min-delay?
performance-metric-normality
               |     +--ro unidirectional-max-delay?
performance-metric-normality
               |     +--ro unidirectional-delay-variation?
performance-metric-normality
               |     +--ro unidirectional-packet-loss?
performance-metric-normality
               |     +--ro unidirectional-residual-bandwidth?
```

```
performance-metric-normality
               |     +--ro unidirectional-available-bandwidth?
performance-metric-normality
               |     +--ro unidirectional-utilized-bandwidth?
performance-metric-normality
               +--ro link-protection-type?              enumeration
               +--ro interface-switching-capabilities*
[switching-capability]
               |  +--ro switching-capability
ted:switching-capabilities
               |  +--ro encoding?                  ted:encoding-type
               |  +--ro max-lsp-bandwidth* [priority]
               |  |  +--ro priority     uint8
               |  |  +--ro bandwidth?   decimal64
               |  +--ro packet-switch-capable
               |  |  +--ro minimum-lsp-bandwidth?   decimal64
               |  |  +--ro interface-mtu?           uint16
               |  +--ro time-division-multiplex-capable
               |     +--ro minimum-lsp-bandwidth?   decimal64
               |     +--ro indication?              enumeration
               +--ro srlg
                  +--ro srlg-values* [srlg-value]
                     +--ro srlg-value    uint32
```

3.1. TE Topology Yang Module

```
<CODE BEGINS> file "ietf-te-topology@2015-03-23.yang"
module ietf-te-topology {
  yang-version 1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-topology";
  // replace with IANA namespace when assigned

  prefix "tet";

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

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

  import ted {
    prefix "ted";
  }

  import ietf-interfaces {
```

```
     prefix "if";
   }

   organization "TBD";
   contact "TBD";
   description "TE topology model";

   revision "2015-03-23" {
     description "Initial revision";
     reference "TBD";
   }

   /*
    * Features
    */

   feature te-topology-hierarchy {
     description
       "This feature indicates that the system allows underlay
        and/or overlay TE topology hierarchy.";
   }

   /*
    * Typedefs
    */

   typedef te-topology-id {
     type string {
       pattern '/?([a-zA-Z0-9\-_.]+)(/[a-zA-Z0-9\-_.]+)*';
     }
     description
       "An identifier for a topology.";
   }

   typedef te-template-name {
     type string {
       pattern '/?([a-zA-Z0-9\-_.]+)(/[a-zA-Z0-9\-_.]+)*';
     }
     description
       "A type for the name of a TE node template or TE link
       template.";
   }

   typedef te-node-id {
     type inet:ip-address;
     description
       "An identifier for a node in a topology.
        The identifier is represented as an IPv4 or IPv6 address.
```

```
         The identifier SHOULD be chosen such that the same node in a
         real network topology will always be identified through the
         same identifier, even if the model is instantiated in
          separate datastores. An implementation MAY choose to capture
         semantics in the identifier, for example to indicate the type
           of node and/or the type of topology that the node is a part
         of.";
    }

    typedef te-link-id {
      type union {
        type uint32;          // Unnumbered
        type inet:ip-address; // IPv4 or IPv6 address
      }
      description
        "An identifier for a TE link on a node.
         The identifier may be opaque.
         The identifier SHOULD be chosen such that the same TP in a
         real network topology will always be identified through the
         same identifier, even if the model is instantiated in
          separate datastores. An implementation MAY choose to capture
         semantics in the identifier, for example to indicate the type
         of TP and/or the type of node and topology that the TP is a
         part of.";
    }


    typedef te-topology-event-type {
      type enumeration {
        enum "add" {
          value 0;
          description
            "A TE node or te-link has
            been added";
        }
        enum "remove" {
          value 1;
          description
            "A TE node or te-link has
            been removed";
        }
        enum "update" {
          value 2;
          description
            "A TE node or te-link has
            been updated";
        }
      }
```

```
      description "TE  Event type for notifications";
    } // te-topology-event-type

    /*
     * Identities
     */

    identity flag-identity {
      description "Base type for flags";
    }

    identity undefined-flag {
      base "flag-identity";
      description "Undefined flag";
    }

    typedef flag-type {
      type identityref {
        base "flag-identity";
      }
      description "Type for flags";
    }

    /*
     * Groupings
     */

    grouping topo-ref {
      description
        "Grouping for an absolute reference to a topology instance.";
      leaf topo-ref {
        type leafref {
          path "/tet:te-topologies/tet:topology/tet:te-topology-id";
        }
        description
          "An absolute reference to a topology instance.";
      }
    }

    grouping link-ref {
      description
        "Grouping for an absolute reference to a link instance.";
      uses topo-ref;
      leaf source-te-node-id-ref {
        type leafref {
          path "/tet:te-topologies/tet:topology"
            +"[tet:te-topology-id = current()/../topo-ref]"
            +"/tet:link/tet:source-te-node-id";
```

```
        }
        description
          "An absolute reference to a link instance.";
      }
      leaf source-te-link-id-ref {
        type leafref {
          path "/tet:te-topologies/tet:topology"
            +"[tet:te-topology-id = current()/../topo-ref]"
            +"/tet:link/tet:source-te-link-id";
        }
        description
          "An absolute reference to a link instance.";
      }
      leaf dest-te-node-id-ref {
        type leafref {
          path "/tet:te-topologies/tet:topology"
            +"[tet:te-topology-id = current()/../topo-ref]"
            +"/tet:link/tet:dest-te-node-id";
        }
        description
          "An absolute reference to a link instance.";
      }
      leaf dest-te-link-id-ref {
        type leafref {
          path "/tet:te-topologies/tet:topology"
            +"[tet:te-topology-id = current()/../topo-ref]"
            +"/tet:link/tet:dest-te-link-id";
        }
        description
          "An absolute reference to a link instance.";
      }
    }

    grouping node-ref {
      description
        "Grouping for an absolute reference to a node instance.";
      uses topo-ref;
      leaf node-ref {
        type leafref {
          path "/tet:te-topologies/tet:topology"
            +"[tet:te-topology-id = current()/../topo-ref]"
            +"/tet:node/tet:te-node-id";
        }
        description
          "An absolute reference to a node instance.";
      }
    }
```

```
    grouping link-end-ref {
      description
        "Grouping for an absolute reference to a TE link end, which is
        the local representation of a TE link on a node.";
      uses node-ref;
      leaf link-end-ref {
        type leafref {
          path "/tet:te-topologies/tet:topology"
            +"[tet:te-topology-id = current()/../topo-ref]"
            +"/tet:node[tet:te-node-id = current()/../node-ref]"
            +"/tet:te-node-attributes/tet:te-link/tet:te-link-id";
        }
        description
          "Grouping for an absolute reference to a TE link end.";
      }
    }

    grouping te-topology-type {
      description
        "Identifies the TE topology type.";
      container te-topology {
        presence "indicates TE topology";
        description
          "Its presence identifies the TE topology type.";
      }
    }

    grouping te-path-element {
      description
        "A group of attributes defining an element in a TE path
        such as TE node, TE link, TE aotomic resource or label.";
      leaf loose {
        type boolean;
        description "true if the element is loose.";
      }
      choice element-type {
        description "Attributes for various element types.";
        case numbered-link {
          leaf link-ip-address {
            type inet:ip-address;
            description "IPv4 or IPv6 address.";
          }
        }
        case unnumbered-link {
          leaf link-node-id {
            type uint32;
            description
              "Node ID of the node where the link end point resides.";
```

```
          }
          leaf te-link-id {
            type uint32;
            description "Identifies the link end point.";
          }
        }
        case node {
          leaf te-node-id {
            type uint32;
            description "Identifies the node.";
          }
        }
        case label {
          leaf label {
            type uint32;
            description "Identifies atomic TE resource or label.";
          }
        }
      }
    } // te-path-element

    grouping config-schedule-attributes {
      description
        "A list of schedules defining when a particular
         configuration takes effect.";
      list schedules {
        key "schedule-id";
        description "A list of schedule elements.";

        leaf schedule-id {
          type uint32;
          description "Identifies the schedule element.";
        }
        leaf start {
          type yang:date-and-time;
          description "Start time.";
        }
        leaf schedule-duration {
          type string {
            pattern
              'P(\d+Y)?(\d+M)?(\d+W)?(\d+D)?T(\d+H)?(\d+M)?(\d+S)?';
          }
          description "Schedule duration in ISO 8601 format.";
        }
        leaf repeat-interval {
          type string {
            pattern
              'R\d*/P(\d+Y)?(\d+M)?(\d+W)?(\d+D)?T(\d+H)?(\d+M)?'
```

```
              + '(\d+S)?';
          }
          description "Repeat interval in ISO 8601 format.";
       }
      }
    }

    grouping information-source-attributes {
      description
        "The attributes identifying source that has provided the
        related information, and the source credibility.";
      leaf information-source {
        type enumeration {
          enum "unknown" {
            description "The source is unknown";
          }
          enum "locally-configured" {
            description "Configured TE link";
          }
          enum "ospfv2" {
            description "OSPFv2";
          }
          enum "ospfv3" {
            description "OSPFv3";
          }
          enum "isis" {
            description "ISIS";
          }
          enum "other" {
            description "Other source";
          }
        }
        description
          "Indicates the source of the information.";
      }
      leaf credibility-preference {
        type uint16;
        description
          "The preference value to calculate the traffic
          engineering database credibility value used for
          tie-break selection between different
          information-source values.
          Higher value is more preferable.";
      }
    }

    grouping te-node-attributes {
      description "Node attributes in a TE topology.";
```

```
      container te-node-attributes {
        description "Node attributes in a TE topology.";
        uses config-schedule-attributes;
        leaf name {
          type inet:domain-name;
          description "Node name.";
        }
        leaf-list signaling-address {
          type inet:ip-address;
          description "Node signaling address.";
        }
        leaf-list flag {
          type flag-type;
          description "Node operational flags.";
        }
        leaf is-abstract {
          type boolean;
          description
            "true if the node is abstract, false when the node is
            actual.";
        }
        leaf underlay-topology {
          if-feature te-topology-hierarchy;
          type leafref {
            path "/tet:te-topologies/tet:topology/tet:te-topology-id";
          }
          description
            "When an abstract node encapsulates a topology,
             this reference points to said topology.";
        }
        list te-link {
          key "te-link-id";
          description
            "The local representation of a TE link, which
              interconnect TE nodes.";
          leaf te-link-id {
            type te-link-id;
            description
              "TE link identifier.";
          }
          choice stack-level {
            description
              "The TE link can be partitioned into bundled
                links, or component links.";
            case bundle {
              container bundled-links {
                description
                  "A set of bundled links";
```

```
                list bundled-link {
                  key "sequence";
                  description
                    "Specify a bundled interfface that is
                      further partitioned.";
                  leaf sequence {
                    type uint32;
                    description
                      "Identify the sequence in the bundle.";
                  }
                  leaf te-link-ref {
                    type leafref {
                      path "../../../te-link-id";
                      require-instance "true";
                    }
                    description
                      "Reference to TE link on this node.";
                  }
                }
              }
            }
            case component {
              container component-links {
                description
                  "A set of component links";
                list component-link {
                  key "sequence";
                  description
                    "Specify a component interfface that is
                      sufficient to unambiguously identify the
                      appropriate resources";

                  leaf sequence {
                    type uint32;
                    description
                      "Identify the sequence in the bundle.";
                  }
                  leaf component-link-ref {
                    type leafref {
                      path "/if:interfaces/if:interface/if:name";
                      require-instance "false";
                    }
                    description
                      "Reference to component link on this
                        node.";
                  }
                }
              }
```

```
          }
        }
      }
      list connectivity-matrix {
        key "id";
        description
          "Represents node’s switching limitations, i.e. limitations
           in interconnecting network TE links across the node.";
        leaf id {
          type uint32;
          description "Identifies the connectivity-matrix entry.";
        }
        container from-link {
          uses tet:link-end-ref;
          description
            "Reference to source NTP.";
        }
        container to-link {
          uses tet:link-end-ref;
          description
            "Reference to destination NTP.";
        }
        leaf is-allowed {
          type boolean;
          description
            "true  - switching is allowed,
             false - switching is disallowed.";
        }
      }
      container ted {
        description "Includes TE node attributes.";
        uses ted:ted-node-attributes;
      }
    }
  } // te-node-attributes

  grouping te-node-state-attributes {
    description "Node state attributes in a TE topology.";
    container te-node-state-attributes {
      description "Node state attributes in a TE topology.";
      uses information-source-attributes;
    }
  } // te-node-state-attributes

  grouping te-link-underlay-attributes {
    description "Attributes for  te-link underlay.";
    container underlay-path {
      description
```

```
          "The service path on the underlay topology that
           supports this link.";
        leaf topology-id {
          type leafref {
            path "/tet:te-topologies/tet:topology/tet:te-topology-id";
            require-instance false;
          }
          description
            "Identifies the topology where the path belongs.";
        }
        list path-element {
          key "path-element-id";
          description
            "A list of path elements describing the service path";
          leaf path-element-id {
            type uint32;
            description "To identify the element in a path.";
          }
          uses te-path-element;
        }
      } // underlay-path
      container underlay-backup-path {
        description
          "The backup service path on the underlay topology that
           supports this link.";
        leaf topology-id {
          type leafref {
            path "/tet:te-topologies/tet:topology/tet:te-topology-id";
            require-instance false;
          }
          description
            "Identifies the topology where the path belongs.";
        }
        list path-element {
          key "path-element-id";
          description
            "A list of path elements describing the backup service
             path";
          leaf path-element-id {
            type uint32;
            description "To identify the element in a path.";
          }
          uses te-path-element;
        }
      } // underlay-backup-path
      leaf underlay-protection-type {
        type uint16;
        description
```

```
          "Underlay protection type desired for this link";
      }
      container underlay-trail-src {
        uses tet:link-end-ref;
        description
          "Source TE link of the underlay trail.";
      }
      container underlay-trail-des {
        uses tet:link-end-ref;
        description
          "Destination TE link of the underlay trail.";
      }
    } // te-link-underlay-attributes

    grouping te-link-state-underlay-attributes {
      description "State attributes for te-link underlay.";
      leaf dynamic {
        type boolean;
        description
          "true if the underlay is dynamically created.";
      }
      leaf committed {
        type boolean;
        description
          "true if the underlay is committed.";
      }
    } // te-link-state-underlay-attributes

    grouping te-link-attributes {
      description
        "Link attributes in a TE topology.";
      container te-link-attributes {
        description "Link attributes in a TE topology.";
        uses config-schedule-attributes;
        leaf name {
          type string;
          description "Link Name";
        }
        leaf-list flag {
          type flag-type;
          description "Link flags";
        }
        leaf is-abstract {
          type boolean;
          description "true if the link is abstract.";
        }
        container underlay {
          if-feature te-topology-hierarchy;
```

```
          presence
            "Indicates the underlay exists for this link.";
          description "State of the underlay of this link.";

          uses te-link-underlay-attributes;
        } // underlay
        container ted {
          description "Includes TE link attributes.";
          uses ted:ted-link-attributes;
        }
      }
    } // te-link-attributes

    grouping te-link-state-attributes {
      description
        "Link state attributes in a TE topology.";
      container te-link-state-attributes {
        description "Link state attributes in a TE topology.";
        uses information-source-attributes;
      }
    } // te-link-state-attributes

    /*
     * Configuration data nodes
     */

    container te-topologies {
      description
        "This container acts as the top-level data element of
         configuration data.";
      list topology {
        key "te-topology-id";
        description
          "This is the model of an abstract topology. A topology
          contains nodes and links. Each topology MUST be identified
          by a unique te-topology-id for reason that a network could
          contain many topologies.";
        leaf te-topology-id {
          type te-topology-id;
          description
            "It is presumed that a datastore will contain many
            topologies. To distinguish between topologies it is
            vital to have UNIQUE topology identifiers.";
        }
        container topology-types {
          description
            "This container is used to identify the type, or types (as
            a topology can support several types simultaneously), of
```

```
            the topology.
            Topology types are the subject of several integrity
            constraints that an implementing server can validate in
            order to maintain integrity of the datastore.
            Topology types are indicated through separate data nodes;
            the set of topology types is expected to increase over
            time.
            To add support for a new topology, an augmenting module
            needs to augment this container with a new empty optional
            container to indicate the new topology type.
            The use of a container allows to indicate a
            subcategorization of topology types.
            The container SHALL NOT be augmented with any data nodes
            that serve a purpose other than identifying a particular
            topology type.";
          uses te-topology-type; // Defines the TE topology type.
        }
        list node {
          key "te-node-id";
          leaf te-node-id {
            type te-node-id;
            description
              "The identifier of a node in the topology.
              A node is specific to a topology to which it belongs.";
          }
          description
            "The list of network nodes defined for the topology.";
          leaf te-node-template {
            type leafref {
              path "/te-topologies/node-template/name";
            }
            description
              "The reference to a TE node template.";
          }
          uses te-node-attributes;
        }
        list link {
          key "source-te-node-id source-te-link-id "
            + "dest-te-node-id dest-te-link-id";
          leaf source-te-node-id {
            type leafref {
              path "../../node/te-node-id";
            }
            mandatory true;
            description
              "Source node identifier, must be in same topology.";
          }
          leaf source-te-link-id {
```

```
            type leafref {
              path "../../node[te-node-id = "
                + "current()/../source-te-node-id]/"
                + "te-node-attributes/te-link/te-link-id";
            }
            mandatory true;
            description
              "Source TE link identifier, must be in same topology.";
          }
          leaf dest-te-node-id {
            type leafref {
              path "../../node/te-node-id";
            }
            mandatory true;
            description
              "Destination node identifier, must be in the same
              topology.";
          }
          leaf dest-te-link-id {
            type leafref {
              path "../../node[te-node-id = "
                + "current()/../dest-te-node-id]/"
                + "te-node-attributes/te-link/te-link-id";
            }
            mandatory true;
            description
              "Destination TE link identifier, must be in same
              topology.";
          }
          description
            "TE link is a logical construct that represents a way
            to group/map information about certain physical
            resources (and their properties) that interconnect TE
            nodes.
            A Network Link connects a by Local (Source) node and
            a Remote (Destination) Network Nodes via a set of the
            nodes’ TE links.
            As it is possible to have several links between the
            same source and destination nodes, and as a link
            could potentially be re-homed, to ensure that we
            would always know to distinguish between
            links, every link is identified by a dedicated link
            identifier.
            Note that a link models a point-to-point link, not a
            multipoint link.";
          leaf te-link-template {
            type leafref {
              path "/te-topologies/link-template/name";
```

```
            }
            description
              "The reference to a TE link template.";
          }
          uses te-link-attributes;
        } // link
      } // topology

      list node-template {
        key "name";
        leaf name {
          type te-template-name;
          description
            "The name to identify a TE node template.";
        }
        description
          "The list of TE node templates used to define sharable
          and reusable TE node attributes.";
        uses te-node-attributes;
      } // node

      list link-template {
        key "name";
        leaf name {
          type te-template-name;
          description
            "The name to identify a TE link template.";
        }
        description
          "The list of TE link templates used to define sharable
          and reusable TE link attributes.";
        uses te-link-attributes;
      } // link
    } // te-topologies

    /*
     * Operational state data nodes
     */

    container te-topologies-state {
      config "false";
      description
        "This container acts as the top-level state data element of
         operational data.";
      list topology {
        key "te-topology-id";
        description
          "This is the model of an abstract topology. A topology
```

```
          contains nodes and links. Each topology MUST be identified
          by a unique te-topology-id for reason that a network could
          contain many topologies.";
        leaf te-topology-id {
          type te-topology-id;
          description
            "It is presumed that a datastore will contain many
            topologies. To distinguish between topologies it is
            vital to have UNIQUE topology identifiers.";
        }
        leaf server-provided {
          type boolean;
          config false;
          description
            "Indicates whether the topology is configurable by
               clients, or whether it is provided by the server. This
              leaf is populated by the server implementing the model.
            It is set to false for topologies that are created by a
            client; it is set to true otherwise.  If it is set to
              true, any attempt to edit the topology MUST be rejected.";
        }
        container topology-types {
          description
            "This container is used to identify the type, or types (as
            a topology can support several types simultaneously), of
            the topology.
            Topology types are the subject of several integrity
            constraints that an implementing server can validate in
            order to maintain integrity of the datastore.
            Topology types are indicated through separate data nodes;
            the set of topology types is expected to increase over
            time.
            To add support for a new topology, an augmenting module
            needs to augment this container with a new empty optional
            container to indicate the new topology type.
            The use of a container allows to indicate a
            subcategorization of topology types.
            The container SHALL NOT be augmented with any data nodes
            that serve a purpose other than identifying a particular
            topology type.";
          uses te-topology-type; // Defines the TE topology type.
        }
        list node {
          key "te-node-id";
          leaf te-node-id {
            type te-node-id;
            description
              "The identifier of a node in the topology.
```

```
            A node is specific to a topology to which it belongs.";
        }
        description
          "The list of network nodes defined for the topology.";
        leaf te-node-template {
          type leafref {
            path "/te-topologies/node-template/name";
          }
          description
            "The reference to a TE node template.";
        }
        uses te-node-attributes;
        uses te-node-state-attributes;
      }
      list link {
        key "source-te-node-id source-te-link-id "
          + "dest-te-node-id dest-te-link-id";
        leaf source-te-node-id {
          type leafref {
            path "../../node/te-node-id";
          }
          mandatory true;
          description
            "Source node identifier, must be in same topology.";
        }
        leaf source-te-link-id {
          type leafref {
            path "../../node[te-node-id = "
              + "current()/../source-te-node-id]/"
              + "te-node-attributes/te-link/te-link-id";
          }
          mandatory true;
          description
            "Source TE link identifier, must be in same topology.";
        }
        leaf dest-te-node-id {
          type leafref {
            path "../../node/te-node-id";
          }
          mandatory true;
          description
            "Destination node identifier, must be in the same
            topology.";
        }
        leaf dest-te-link-id {
          type leafref {
            path "../../node[te-node-id = "
              + "current()/../dest-te-node-id]/"
```

```
                + "te-node-attributes/te-link/te-link-id";
            }
            mandatory true;
            description
              "Destination TE link identifier, must be in same
              topology.";
          }
          description
            "TE link is a logical construct that represents a way
            to group/map information about certain physical
            resources (and their properties) that interconnect TE
            nodes.
            A Network Link connects a by Local (Source) node and
            a Remote (Destination) Network Nodes via a set of the
            nodes' TE links.
            As it is possible to have several links between the
            same source and destination nodes, and as a link
            could potentially be re-homed, to ensure that we
            would always know to distinguish between
            links, every link is identified by a dedicated link
            identifier.
            Note that a link models a point-to-point link, not a
            multipoint link.";
          leaf te-link-template {
            type leafref {
              path "/te-topologies/link-template/name";
            }
            description
              "The reference to a TE link template.";
          }
          uses te-link-attributes;
          uses te-link-state-attributes;
        } // link
      } // topology
    } // te-topologies

    augment "/te-topologies-state/topology/link/te-link-attributes/"
      + "underlay" {
      description "Add state attributes to te-link underlay.";
      uses te-link-state-underlay-attributes;
    }

    /*
     * Notifications
     */

    notification te-node-event {
      description "Notification event for TE node";
```

```
      leaf event-type {
        type te-topology-event-type;
        description "Event type";
      }
      uses node-ref;
      uses te-topology-type;
      uses tet:te-node-attributes;
    }

    notification te-link-event {
      description "Notification event for TE link";
      leaf event-type {
        type te-topology-event-type;
        description "Event type";
      }
      uses link-ref;
      uses te-topology-type;
      uses tet:te-link-attributes;
    }

    augment "/te-link-event/te-link-attributes/underlay" {
      description "Add state attributes to te-link underlay.";
      uses te-link-state-underlay-attributes;
    }
  }
  <CODE ENDS>
```

4. Normative References

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

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Authors' Addresses

Xufeng Liu
Ericsson

Email: xufeng.liu@ericsson.com

Igor Bryskin
ADVA Optical Networking

Email: ibryskin@advaoptical.com

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Tarek Saad
Cisco Systems Inc.

Email: tsaad@cisco.com

Himanshu Shah
Ciena

Email: tsaad@cisco.com

q

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

J. George
Google
L. Fang
Microsoft
E. Osborne
Level 3
R. Shakir
Jive Communications

October 19, 2015

MPLS / TE Model for Service Provider Networks
draft-openconfig-mpls-consolidated-model-02

Abstract

This document defines a framework for a YANG data model for configuring and managing label switched paths, including the signaling protocols, traffic engineering, and operational aspects based on carrier and content provider operational requirements.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes a YANG [RFC6020] data model for MPLS and traffic engineering, covering label switched path (LSP) configuration, as well as signaling protocol configuration. The model is intended to be vendor-neutral, in order to allow operators to manage MPLS in heterogeneous environments with physical or virtual devices (routers, switches, servers, etc.) supplied by multiple vendors. The model is also intended to be readily mapped to existing implementations, to facilitate support from as large a set of routing hardware and software vendors as possible.

1.1. Goals and approach

The focus area of the model in this revision, is to set forth a framework for MPLS, with hooks into which information specific to various signaling-protocols can be added. The framework is built around functionality from a network operator perspective rather than a signaling protocol-centric approach. For example, a traffic-engineered LSP will have configuration relating to its path computation method, regardless of whether it is signaled with RSVP-TE or with segment routing. Thus, rather than creating separate per-signaling protocol models and trying to stitch them under a common umbrella, this framework focuses on functionality, and adds signaling protocol-specific information under it where applicable.

This model does not aim to be feature complete (i.e., cover all possible aspects or features of MPLS). Rather its development is driven by examination of actual production configurations in use across a number of operator network deployments.

Configuration items that are deemed to be widely available in existing major implementations are included in the model. Those configuration items that are only available from a single implementation are omitted from the model with the expectation they will be available in companion modules that augment the current model. This allows clarity in identifying data that is part of the vendor-neutral model.

An important aspect of the model is the representation of operational state data. This draft takes the approach described in [I-D.openconfig-netmod-opstate] and models configuration and operational state together. Thus, rather than building a separate tree of operational state, the operational state and configuration data are located in parallel containers at the leaves of the data model. This approach allows easy reuse of groupings across models, as well as making it easier to correlate configuration and state.

The consolidated MPLS model encompasses the signaling protocols, label-switched paths (configuration and operational state), and generic TE attributes. The model is designed from an operational and functional perspective, rather than focusing on protocol-centric configuration. This allows protocol-independent functions to be logically separated from protocol-specific details.

One question that arises in this approach is how the consolidated model is integrated with routing instances (e.g., VRFs). This model should be considered as part of a higher level network device model which includes definitions for other routing protocols and system services. For example, in [I-D.openconfig-netmod-model-structure], VRFs and other logical instances are defined with MPLS/TE components within VRFs as appropriate. In particular, some parts of the MPLS model would be instantiated within a VRF, while other parts would have common definitions across VRFs.

Where possible, naming in the model follows conventions used in available standards documents, and otherwise tries to be self-explanatory with sufficient descriptions of the intended behavior. Similarly, configuration data value constraints and default values, where used, are based on recommendations in current standards documentation. Since implementations vary widely in this respect, this version of the model specifies only a limited set of defaults and ranges with the expectation of being more prescriptive in future versions based on actual operator use.

Note that this version of the model is a work-in-progress in several respects. Although we present a complete framework for MPLS and traffic engineering from an operational perspective, some signaling protocol configuration will be completed in future revisions. The current revision has focus on traffic engineered LSPs signaled with RSVP.

2. Model overview

The overall MPLS model is defined across several YANG modules and submodules but at a high level is organized into 4 main sections:

- o global -- configuration affecting MPLS behavior which exists independently of the underlying signaling protocol or label switched path configuration.
- o te-global-attributes -- configuration affecting MPLS-TE behavior which exists independently of the underlying signaling protocol or label switched path configuration.
- o signaling protocols -- configuration specific to signaling protocols used to setup and manage label switched paths.
- o label switched paths -- configuration specific to instantiating and managing individual label switched paths.

The top level of the model is shown in the tree view below:

```
+--rw mpls!
      +--rw global
      |     ...
      +--rw te-global-attributes
      |     ...
      +--rw signaling-protocols
      |     ...
      +--rw lsps
            ...
```

2.1. MPLS global

The global section of the framework provides configuration data for MPLS items which exist independently of an individual label switched path or signaling protocol and are applicable to the MPLS protocol itself. Items such as the depth of the label stack supported, or specific label ranges may be included here.

2.2. TE global attributes

The TE global attributes section of the framework provides configuration control for MPLS-TE items which exist independently of an individual label switched path or signaling protocol. These standalone items are applicable to the entire logical routing device, and establish fundamental configuration such as the threshold for interface bandwidth change that triggers update events into the IGP traffic engineering database (TED). Timers are also specified which determine the length of time an LSP must be present before being considered viable for forwarding use (te-lsp-install-delay), and the length of time between LSP teardown and removal of the LSP from the network element’s forwarding information base (te-lsp-cleanup-delay). Also specified are the name to value mappings of MPLS administrative groups (mpls-admin-groups) and shared risk link groups (mpls-te-srlg).

```
    +--rw te-global-attributes
         |  +--rw mpls-te-srlg
         |  |  +--rw srlg* [srlg-name]
         |  |     +--rw srlg-name       leafref
         |  |     +--rw config
         |  |     |  +--rw srlg-name?    string
         |  |     |  +--rw srlg-value?   uint32
         |  |     |  +--rw srlg-cost?    uint32
         |  |     +--ro state
         ...
         |  |     +--rw members-list* [from-address]
         |  |        +--rw from-address    leafref
         |  |        +--rw config
         |  |        |  +--rw from-address?   inet:ip-address
         |  |        |  +--rw to-address?     inet:ip-address
         ...
         |  +--rw igp-flooding-bandwidth
         |  |  +--rw config
         |  |  |  +--rw threshold-type?            enumeration
         |  |  |  +--rw delta-percentage?          oc-types:percentage
         |  |  |  +--rw threshold-specification?   enumeration
         |  |  |  +--rw up-thresholds*             oc-types:percentage
         |  |  |  +--rw down-thresholds*           oc-types:percentage
         |  |  |  +--rw up-down-thresholds*        oc-types:percentage
         |  |  +--ro state
         ...
         |  +--rw mpls-admin-groups
         |  |  +--rw admin-group* [admin-group-name]
         |  |     +--rw admin-group-name    leafref
         |  |     +--rw config
         |  |     |  +--rw admin-group-name?    string
         |  |     |  +--rw admin-group-value?   uint32
         |  |     +--rw state
         ...
         |  +--rw te-lsp-timers
         |     +--rw config
         |     |  +--rw te-lsp-install-delay?      uint16
         |     |  +--rw te-lsp-cleanup-delay?      uint16
         |     |  +--rw te-lsp-reoptimize-timer?   uint16
         |     +--ro state
         ...
```

2.3. TE interface attributes overview

The TE interface attributes section of the framework provides configuration and state related to traffic engineering such as te-metric or shared risk link group configuration.

```
      +--rw te-intf-attributes
      |  +--rw interface* [interface-name]
      |     +--rw interface-name    leafref
      |     +--rw config
      |     |  +--rw interface-name?           ocif:interface-ref
      |     |  +--rw te-metric?                uint32
      |     |  +--rw srlg* [srlg-name]
      |     |     ...
      |     |  +--rw admin-group* [admin-group-name]
      |     |     ...
      |     |  +--rw igp-flooding-bandwidth
      |     |     ...
      |     +--ro state
      ...
```

2.4. Signaling protocol overview

The signaling protocol section of the framework provides configuration elements for configuring three major methods of signaling label switched paths: RSVP-TE, segment routing, and label distribution protocol (LDP). BGP-LU will be included in a future version of this draft by definitions in the BGP model ([I-D.ietf-idr-bgp-model]) and corresponding augmentations to the MPLS model.

```
+--rw signaling-protocols
      |  +--rw rsvp-te
      |     ...
      |  +--rw segment-routing
      |     ...
      |  +--rw ldp
      |     ...
```

Configuration of RSVP-TE is centered around interfaces on the device which participate in the protocol. A key focus is to expose common RSVP-TE configuration parameters which are used to enhance scale and reliability. Items which are applicable globally in the RSVP-TE protocol such as graceful restart, soft preemption and various statistics are grouped into a global section under the protocol. RSVP neighbor and session state are also available in the RSVP section.

```
         +--rw rsvp-te
         |  |  +--rw rsvp-sessions
         |  |  |  +--rw config
         |  |  |  +--ro state
         |  |  |     +--ro rsvp-session* [source-port destination-port
                           source-address destination-address]
                     ...
         |  |  +--rw rsvp-neighbors
         |  |  |  +--rw config
         |  |  |  +--ro state
         |  |  |     +--ro rsvp-neighbor* [neighbor-address]
                     ...
         |  |  +--rw global
         |  |  |  +--rw graceful-restart
                     ...
         |  |  |  +--rw soft-preemption
                     ...
         |  |  |  +--ro statistics
         |  |  |     +--ro counters
                     ....
         |  |  +--rw interface-attributes
         |  |     +--rw interface* [interface-name]
         |  |        +--rw interface-name    leafref
                     ...
         |  |        +--rw rsvp-hellos
                     ...
         |  |        +--rw authentication
                     ...
         |  |        +--rw subscription
                     ...
         |  |        +--rw protection
                     ...
         ...
```

Containers for specifying signaling via segment routing and LDP are also present. Specific subelements will be added for those protocols, as well as for BGP labeled unicast, in the next revision.

2.5. LSP overview

This part of the framework contains LSP information. At the high level, LSPs are split into three categories: traffic-engineering-capable (constrained-path), non-traffic-engineered determined by the IGP (unconstrained-path), and hop-by-hop configured (static).

```
+--rw mpls!
      +--rw lsps
         +--rw constrained-path
         |     ...
         +--rw unconstrained-path
         |     ...
         +--rw static-lsps
               ...
```

The first two categories, constrained-path and unconstrained-path are the ones for which multiple signaling protocols exist, and are organized in protocol-specific and protocol-independent sections. For example, traffic-engineered (constrained path) LSPs may be set up using RSVP-TE or segment routing, and unconstrained LSPs that follow the IGP path may be signaled with LDP or with segment routing. IGP-determined LSPs may also be signaled by RSVP but this usage is not considered in the current version of the model.

A portion of the data model for constrained path traffic-engineered LSPs signaled with RSVP is shown below. It contains configuration for named explicit paths and for tunnels. Tunnel configuration differs for p2p and p2mp LSPs. In either case, some part of the model is signaling-protocol independent. For example for a p2p LSP, attributes such as the path computation method, the constraints for the the path, the bandwidth allocated to it, and even the frequency of reoptimization are signaling-protocol independent, while other data, such as the setup and hold priorities are protocol-specific and are specified in the protocol specific part of the model.

```
+--rw mpls!
      +--rw lsps
         +--rw constrained-path
        |  +--rw explicit-path* [name]
             ...
          |  +--rw tunnel* [name type]
          |     +--rw name      leafref
          |     +--rw type      leafref
          |     +--rw config
          |     |  +--rw name?                       string
          |     |  +--rw type?                       identityref
          |     |  +--rw local-id?                   union
          |     |  +--rw description?                string
          |     |  +--rw admin-status?               identityref
          |     |  +--rw preference?                 uint8
          |     |  +--rw metric?                     te-metric-type
          |     |  +--rw (bandwidth)?
                   ...
          |     |  +--rw protection-style-requested?   identityref
          |     |  +--rw te-lsp-reoptimize-timer?      uint16
          |     |  +--rw (signaling-specific-tunnel-attributes)?
          |     |  |  +--:(RSVP)
          |     |  |     +--rw source?                     inet:ip-address
          |     |  |     +--rw soft-preemption?            boolean
          |     |  +--rw (tunnel-type)?
          |     |     +--:(p2p)
          |     |     |  +--rw destination?                inet:ip-address
          |     |     |  +--rw primary-paths* [name]
          |     |     |  |  +--rw name                     string
          |     |     |  |  +--rw preference?              uint8
          |     |     |  |  +--rw path-computation-method
                                ...
          |     |     |  |  +--rw admin-groups
                                ...
          |     |     |  |  +--rw no-cspf?                 empty
          |     |     |  |  +--rw (sigaling-specific-path-attributes)?
          |     |     |  |     +--:(RSVP)
          |     |     |  |     |  +--rw setup-priority?           uint8
          |     |     |  |     |  +--rw hold-priority?            uint8
          |     |     |  |     |  +--rw retry-timer?              uint16
          |     |     |  |     +--:(SR)
          |     |     |  |        +--rw sid-selection-mode?       enumeration
          |     |     |  |        +--rw sid-protection-required?   boolean
          |     |     |  +--rw secondary-paths* [name]
                         ...
          |     +--ro state
                ...
```

Similarly, the partial model for non-traffic-engineered, or IGP-based, LSPs is shown below:

```
+--rw mpls!
      +--rw lsps
         +--rw unconstrained-path
            +--rw path-setup-protocol
               +--rw ldp!
               |     ...
               +--rw segment-routing!
                     ...
```

3. Example use cases

3.1. Traffic engineered p2p LSP signaled with RSVP

A possible scenario may be the establishment of a mesh of traffic-engineered LSPs where RSVP signaling is desired, and the LSPs use a local constrained path calculation to determine their path. These LSPs would fall into the category of a constrained-path LSP, and the tunnel type is p2p. Attributes such as metric, bandwidth or the style of protection desired are also defined at this (protocol-independent) level in the model. The path is defined to be locally-computed under the path-computation-method container, specifying the use of CSPF (use-cspf). Additional attributes for the path, such as its RSVP priorities are specified at the path level under the protocol-specific stanza.

```
+--rw mpls!
      +--rw lsps
         +--rw constrained-path
            ...
         |  +--rw tunnel* [name type]
         |     +--rw name      leafref
         |     +--rw type      leafref
         |     +--rw config
         |     |  +--rw name?                          string
         |     |  +--rw type?                          identityref
         |     |  +--rw metric?                        te-metric-type
         |     |  +--rw (bandwidth)?
                  ...
         |     |  +--rw protection-style-requested?   identityref
         |     |  +--rw te-lsp-reoptimize-timer?      uint16
                  ...
         |     |  +--rw (tunnel-type)?
         |     |     +--:(p2p)
         |     |     |  +--rw destination?                  inet:ip-address
         |     |     |  +--rw primary-paths* [name]
         |     |     |  |  +--rw name                       string
         |     |     |  |  +--rw preference?                uint8
         |     |     |  |  +--rw path-computation-method
                              ...
         |     |     |  |  +--rw admin-groups
                              ...
         |     |     |  |  +--rw no-cspf?                   empty
         |     |     |  |  +--rw (sigaling-specific-path-attributes)?
         |     |     |  |     +--:(RSVP)
         |     |     |  |     |  +--rw setup-priority?            uint8
         |     |     |  |     |  +--rw hold-priority?             uint8
         |     |     |  |     |  +--rw retry-timer?               uint16
         |     |     |  |     +--:(SR)
         |     |     |  |        +--rw sid-selection-mode?        enumeration
         |     |     |  |        +--rw sid-protection-required?   boolean
         |     |     |  +--rw secondary-paths* [name]
                        ...
         |     +--ro state
```

3.2. Traffic engineered LSP signaled with SR

A possible scenario may be the establishment of disjoint paths in a network where there is no requirement for per-LSP state to be held on midpoint nodes within the network, or RSVP-TE is unsuitable (as described in [I-D.ietf-spring-segment-routing-mpls] and [I-D.shakir-rtgwg-sr-performance-engineered-lsps]). Such LSPs fall in the constrained-path category. Similar to any other traffic engineered LSPs, the path computation method must be specified. Path attributes, such as the as lsp- placement-constraints (expressed as administrative groups) or metric must be defined. Finally, the path must be specified in a signaling- protocol specific manner appropriate for SR. The same configuration elements from the tree above apply in this case, except that path setup is done by the head-end by building a label stack, rather than signaled.

3.3. IGP-congruent LDP-signaled LSP

A possible scenario may be the establishment of a full mesh of LSPs. When traffic engineering is not an objective, no constraints are placed on the end-to-end path, and the best- effort path can be setup using LDP signaling simply for label distribution. The LSPs follow IGP-computed paths, and fall in the unconstrained-path category in the model. Protocol-specific configuration pertaining to the signaling protocol used, such as the FEC definition and metrics assigned are in the path- setup-protocol portion of the model.

The relevant part of the model for this case is shown below:

```
+--rw mpls!
    +--rw lsps
       +--rw unconstrained-path
          +--rw path-setup-protocol
             +--rw ldp!
                +--rw tunnel
                   +--rw tunnel-type?   mplst:tunnel-type
                   +--rw ldp-type?      enumeration
                   +--rw p2p-lsp
                   |  +--rw fec-address*   inet:ip-prefix
                   +--rw p2mp-lsp
                   +--rw mp2mp-lsp
```

A common operational issue encountered when using LDP is traffic blackholing under the following scenario: when an IGP failure occurs, LDP is not aware of it as these are two protocols running independently, resulting in traffic blackholing at the IGP failure point even though LDP is up and running. LDP-IGP synchronization [RFC5443] can be used to cost out the IGP failing point/segment to avoid the blackholing issue. The LDP-IGP synchronization function will be incorporated in a future version of this document.

Note that targeted LDP sessions are not discussed in this use case, and will be incorporated as a separate use case in a future version of this document.

4. Security Considerations

MPLS configuration has a significant impact on network operations, and as such any related protocol or model carries potential security risks.

YANG data models are generally designed to be used with the NETCONF protocol over an SSH transport. This provides an authenticated and secure channel over which to transfer BGP configuration and operational data. Note that use of alternate transport or data encoding (e.g., JSON over HTTPS) would require similar mechanisms for authenticating and securing access to configuration data.

Most of the data elements in the configuration model could be considered sensitive from a security standpoint. Unauthorized access or invalid data could cause major disruption.

5. IANA Considerations

This YANG data model and the component modules currently use a temporary ad-hoc namespace. If and when it is placed on redirected for the standards track, an appropriate namespace URI will be registered in the IETF XML Registry" [RFC3688]. The MPLS YANG modules will be registered in the "YANG Module Names" registry [RFC6020].

6. YANG modules

The modules and submodules comprising the MPLS configuration and operational model are currently organized as depcited below.

```
                                 +-------+
               +----------------->| MPLS  |<--------------+
               |                 +-------+               |
               |                     ^                   |
               |                     |                   |
          +----+-----+      +--------+-------+     +-----+-----+
          | TE LSPs  |      | IGP-based LSPs |     |static LSPs|
          |          |      |                |     |           |
          +----------+      +----------------+     +-----------+
              ^  ^                    ^  ^
              |  +----------------+   |  +--------+
              |                   |   |           |
              |   +------+      +-+---+-+      +--+--+
              +---+ RSVP |      |SEGMENT|      | LDP |
                  +------+      |ROUTING|      +-----+
                                +-------+
```

The base MPLS module includes submodules describing the three different types of support LSPs, i.e., traffic-engineered (constrained-path), IGP congruent (unconstrained-path), and static. The signaling protocol specific parts of the model are described in separate modules for RSVP, segment routing, and LDP. As mentioned earlier, support for BGP labeled unicast is also planned in a future revision.

A module defining various reusable MPLS types is included, and these modules also make use of the standard Internet types, such as IP addresses, as defined in RFC 6991 [RFC6991].

6.1. MPLS base modules

```
                 <CODE BEGINS> file openconfig-mpls.yang
module openconfig-mpls {

  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/mpls";

  prefix "mpls";


  // import some basic types
  import openconfig-mpls-rsvp { prefix rsvp; }
  import openconfig-mpls-sr { prefix sr; }
  import openconfig-mpls-ldp { prefix ldp; }
  import openconfig-types { prefix oc-types; }
  import openconfig-interfaces { prefix ocif; }

  // include submodules
  include openconfig-mpls-te;
  include openconfig-mpls-igp;
  include openconfig-mpls-static;


  // meta
  organization "OpenConfig working group";

  contact
    "OpenConfig working group
    netopenconfig@googlegroups.com";

  description
    "This module provides data definitions for configuration of
```

```
       Multiprotocol Label Switching (MPLS) and associated protocols for
       signaling and traffic engineering.

       RFC 3031: Multiprotocol Label Switching Architecture

       The MPLS / TE data model consists of several modules and
       submodules as shown below.  The top-level MPLS module describes
       the overall framework.  Three types of LSPs are supported:

       i) traffic-engineered (or constrained-path)

       ii) IGP-congruent (LSPs that follow the IGP path)

       iii) static LSPs which are not signaled

       The structure of each of these LSP configurations is defined in
       corresponding submodules.  Companion modules define the relevant
       configuration and operational data specific to key signaling
       protocols used in operational practice.


                                  +-------+
                +---------------->| MPLS  |<--------------+
                |                 +-------+               |
                |                     ^                   |
                |                     |                   |
           +----+-----+      +--------+-------+     +-----+-----+
           | TE LSPs  |      | IGP-based LSPs |     |static LSPs|
           |          |      |                |     |           |
           +----------+      +----------------+     +-----------+
               ^  ^                    ^  ^
               |  +----------------+   |  +--------+
               |                   |   |           |
               |   +-------+      +-+---+-+      +--+--+
               +---+ RSVP |      |SEGMENT|      | LDP |
                   +-------+      |ROUTING|      +-----+
                                 +-------+
       ";

     revision "2015-10-14" {
       description
         "Work in progress";
       reference "TBD";
     }

     // extension statements

     // feature statements
```

```
// identity statements


// grouping statements

grouping mpls-admin-group_config {
  description
    "configuration data for MPLS link admin groups";

    leaf admin-group-name {
      type string;
      description "name for mpls admin-group";
    }

    leaf admin-group-value {
      type uint32;
      description "value for mpls admin-group";
    }

}

grouping mpls-admin-groups-top {

  description "top-level mpls admin-groups config
    and state containers";

  container mpls-admin-groups {
    description
      "Top-level container for admin-groups configuration
      and state";

    list admin-group {
      key admin-group-name;
      description "configuration of value to name mapping
        for mpls affinities/admin-groups";

      leaf admin-group-name {
        type leafref {
          path "../mpls:config/mpls:admin-group-name";
        }
        description
          "name for mpls admin-group";
      }
      container config {
        description "Configurable items for admin-groups";
        uses mpls-admin-group_config;
      }
```

```
          container state {
            description "Operational state for admin-groups";
            uses mpls-admin-group_config;
          }
        }
      }
    }

    grouping mpls-te-igp-flooding-bandwidth_config {
      description
        "Configurable items for igp flooding bandwidth
        threshold configuration.";
      leaf threshold-type {
        type enumeration {
          enum DELTA {
            description "DELTA indicates that the local
              system should flood IGP updates when a
              change in reserved bandwidth >= the specified
              delta occurs on the interface.";
          }
          enum THRESHOLD-CROSSED {
            description "THRESHOLD-CROSSED indicates that
              the local system should trigger an update (and
              hence flood) the reserved bandwidth when the
              reserved bandwidth changes such that it crosses,
              or becomes equal to one of the threshold values.";
          }
        }
        description
          "The type of threshold that should be used to specify the
          values at which bandwidth is flooded. DELTA indicates that
          the local system should flood IGP updates when a change in
          reserved bandwidth >= the specified delta occurs on the
          interface. Where THRESHOLD-CROSSED is specified, the local
          system should trigger an update (and hence flood) the
          reserved bandwidth when the reserved bandwidth changes such
          that it crosses, or becomes equal to one of the threshold
          values";
      }

      leaf delta-percentage {
        when "../threshold-type = 'DELTA'" {
          description
            "The percentage delta can only be specified when the
            threshold type is specified to be a percentage delta of
            the reserved bandwidth";
        }
        type oc-types:percentage;
```

```
          description
            "The percentage of the maximum-reservable-bandwidth
            considered as the delta that results in an IGP update
            being flooded";
        }

        leaf threshold-specification {
          when "../threshold-type = 'THRESHOLD-CROSSED'" {
            description
              "The selection of whether mirrored or separate threshold
              values are to be used requires user specified thresholds to
              be set";
          }
          type enumeration {
            enum MIRRORED-UP-DOWN {
              description
                "MIRRORED-UP-DOWN indicates that a single set of
                threshold values should be used for both increasing
                and decreasing bandwidth when determining whether
                to trigger updated bandwidth values to be flooded
                in the IGP TE extensions.";
            }
            enum SEPARATE-UP-DOWN {
              description
                "SEPARATE-UP-DOWN indicates that a separate
                threshold values should be used for the increasing
                and decreasing bandwidth when determining whether
                to trigger updated bandwidth values to be flooded
                in the IGP TE extensions.";
            }
          }
          description

            "This value specifies whether a single set of threshold
            values should be used for both increasing and decreasing
            bandwidth when determining whether to trigger updated
            bandwidth values to be flooded in the IGP TE extensions.
            MIRRORED-UP-DOWN indicates that a single value (or set of
            values) should be used for both increasing and decreasing
            values, where SEPARATE-UP-DOWN specifies that the increasing
            and decreasing values will be separately specified";
        }

        leaf-list up-thresholds {
          when "../threshold-type = 'THRESHOLD-CROSSED'" +
            "and ../threshold-specification = 'SEPARATE-UP-DOWN'" {
              description
                "A list of up-thresholds can only be specified when the
```

```
               bandwidth update is triggered based on crossing a
               threshold and separate up and down thresholds are
               required";
          }
        type oc-types:percentage;
        description
          "The thresholds (expressed as a percentage of the maximum
          reservable bandwidth) at which bandwidth updates are to be
          triggered when the bandwidth is increasing.";
      }

      leaf-list down-thresholds {
        when "../threshold-type = 'THRESHOLD-CROSSED'" +
          "and ../threshold-specification = 'SEPARATE-UP-DOWN'" {
            description
              "A list of down-thresholds can only be specified when the
              bandwidth update is triggered based on crossing a
              threshold and separate up and down thresholds are
              required";
          }
        type oc-types:percentage;
        description
          "The thresholds (expressed as a percentage of the maximum
          reservable bandwidth) at which bandwidth updates are to be
          triggered when the bandwidth is decreasing.";
      }

      leaf-list up-down-thresholds {
        when "../threshold-type = 'THRESHOLD-CROSSED'" +
          "and ../threshold-specification = 'MIRRORED-UP-DOWN'" {
            description
              "A list of thresholds corresponding to both increasing
              and decreasing bandwidths can be specified only when an
              update is triggered based on crossing a threshold, and
              the same up and down thresholds are required.";
          }
        type oc-types:percentage;
        description
          "The thresholds (expressed as a percentage of the maximum
          reservable bandwidth of the interface) at which bandwidth
          updates are flooded - used both when the bandwidth is
          increasing and decreasing";
      }
    }

    grouping mpls-te-igp-flooding-bandwdith-if {
      description "Interface-level group for traffic engineering
      database flooding options options";
```

```
        container igp-flooding-bandwidth {
         description "Interface bandwidth change percentages
         that trigger update events into the IGP traffic
         engineering database (TED)";
         uses  mpls-te-igp-flooding-bandwidth_config;
        }
     }


     grouping  mpls-te-igp-flooding-bandwidth {
       description "Top level group for traffic engineering
         database flooding options";
       container igp-flooding-bandwidth {
         description "Interface bandwidth change percentages
           that trigger update events into the IGP traffic
           engineering database (TED)";
         container config {
           description "Configuration parameters for TED
             update threshold ";
           uses  mpls-te-igp-flooding-bandwidth_config;
         }
         container state {
           config false;
           description "State parameters for TED update threshold ";
           uses  mpls-te-igp-flooding-bandwidth_config;
         }
       }
     }


     grouping te_lsp_delay_config {
       description "Group for the timers goerning the delay
       in installation and cleanup of TE LSPs";

       leaf te-lsp-install-delay {
         type uint16 {
           range 0..3600;
         }
         units seconds;
         description "delay the use of newly installed te lsp for a
         specified amount of time.";
       }

       leaf te-lsp-cleanup-delay {
         type uint16;
         units seconds;
         description "delay the removal of old te lsp for a specified
         amount of time";
```

```
      }
    }

    grouping te-interface-attributes-top {
      description
        "Top level grouping for attributes
        for TE interfaces.";

      list interface {
        key interface-name;
        description "List of TE interfaces";

        leaf interface-name {
          type leafref {
            path "../config/interface-name";
            require-instance true;
          }
          description "The interface name";
        }

        container config {
          description
            "Configuration parameters related to TE interfaces:";
          uses te-interface-attributes-config;
        }

        container state {
          config false;
          description "State parameters related to TE interfaces";
          uses te-interface-attributes-config;
        }
      }
    }

    grouping te-interface-attributes-config {
      description "global level definitions for interfaces
        on which TE is run";

      leaf interface-name {
        type ocif:interface-ref;
        description "reference to interface name";
      }

      leaf te-metric {
        type uint32;
        description "TE specific metric for the link";
      }
```

```
      list srlg {
        key srlg-name;
        description "list of shared risk link groups on the
          interface";
        leaf srlg-name {
          type string;
          description "The SRLG group identifier";
        }
      }

      list admin-group {
        key admin-group-name;
        description "list of admin groups on the
          interface";
        leaf admin-group-name {
          type string;
          description "The admin group identifier";
        }
      }

      uses  mpls-te-igp-flooding-bandwdith-if;

    }

    grouping mpls-te-lsp-timers {
      description
        "Grouping for traffic engineering timers";
      container te-lsp-timers {
        description
        "definition for delays associated with setup
        and cleanup of TE LSPs";

        container config {
          description
          "Configuration parameters related
          to timers for TE LSPs";

          uses te_lsp_delay_config;
          uses te-tunnel-reoptimize_config;
        }
        container state {
          config false;
          description "State related to timers for TE LSPs";

          uses te_lsp_delay_config;
          uses te-tunnel-reoptimize_config;
        }
      }
```

```
     }

   container mpls {
      presence "top-level container for MPLS config and operational
      state";

      description "Anchor point for mpls configuration and operational
      data";

      container global {
        // entropy label support, label ranges will be added here.
       description "general mpls configuration applicable to any
       type of LSP and signaling protocol - label ranges,
       entropy label supportmay be added here";
      }

      container te-global-attributes {
        description "traffic-engineering global attributes";
        uses mpls-te-srlg-top;
        uses mpls-te-igp-flooding-bandwidth;
        uses mpls-admin-groups-top;
        uses mpls-te-lsp-timers;
      }

      container te-intf-attributes {
        description "traffic engineering attributes specific
        for interfaces";
        uses te-interface-attributes-top;
      }

      container signaling-protocols {
        description "top-level signaling protocol configuration";

        uses rsvp:rsvp-global;
        uses sr:sr-global;
        uses ldp:ldp-global;
      }

      container lsps {
        description "LSP definitions and configuration";

        container constrained-path {
          description "traffic-engineered LSPs supporting different
          path computation and signaling methods";
          uses explicit-paths-top;
          uses te-tunnels-top;

        }
```

```
          container unconstrained-path {
            description "LSPs that use the IGP-determined path, i.e., non
            traffic-engineered, or non constrained-path";

            uses igp-lsp-common;
            uses igp-lsp-setup;

          }

          container static-lsps {
            description "statically configured LSPs, without dynamic
              signaling";

            uses static-lsp-main;
          }
        }
      }

      // augment statements

      // rpc statements

      // notification statements
    }
                     <CODE ENDS>



                     <CODE BEGINS> file openconfig-mpls-types.yang
    module openconfig-mpls-types {

      yang-version "1";

      // namespace
      namespace "http://openconfig.net/yang/mpls-types";

      prefix "mplst";

      // meta
      organization "OpenConfig working group";

      contact
        "OpenConfig working group
        netopenconfig@googlegroups.com";

      description
        "General types for MPLS / TE data model";
```

```
revision "2015-10-04" {
  description
    "Work in progress";
  reference "TBD";
}

// extension statements

// feature statements

// identity statements

// using identities rather than enum types to simplify adding new
// signaling protocols as they are introduced and supported
identity path-setup-protocol {
  description "base identity for supported MPLS signaling
  protocols";
}

identity path-setup-rsvp {
  base path-setup-protocol;
  description "RSVP-TE signaling protocol";
}

identity path-setup-sr {
  base path-setup-protocol;
  description "Segment routing";
}

identity path-setup-ldp {
  base path-setup-protocol;
  description "lDP - RFC 5036";
}


identity protection-type {
  description "base identity for protection type";
}

identity unprotected {
  base protection-type;
  description "no protection is desired";
}

identity link-protection-requested {
  base protection-type;
  description "link protection is desired";
}
```

```
   identity link-node-protection-requested {
     base protection-type;
     description "node and link protection are both desired";
   }

   identity lsp-role {
     description
       "Base identity for describing the role of
        label switched path at the current node";
   }

   identity INGRESS {
     base "lsp-role";
     description
       "Label switched path is an ingress (headend)
        LSP";
   }

   identity EGRESS {
     base "lsp-role";
     description
       "Label switched path is an egress (tailend)
        LSP";
   }

   identity TRANSIT {
     base "lsp-role";
     description
       "Label switched path is a transit LSP";
   }


   identity tunnel-type {
     description
       "Base identity from which specific tunnel types are
       derived.";
   }

   identity P2P {
     base tunnel-type;
     description
       "TE point-to-point tunnel type.";
   }

   identity P2MP {
     base tunnel-type;
     description
       "TE point-to-multipoint tunnel type.";
```

```
     }


     identity lsp-oper-status {
       description
         "Base identity for LSP operational status";
     }

     identity DOWN {
       base "lsp-oper-status";
       description
         "LSP is operationally down or out of service";
     }

     identity UP {
       base "lsp-oper-status";
       description
         "LSP is operationally active and available
          for traffic.";
     }

     identity tunnel-admin-status {
       description
         "Base identity for tunnel administrative status";
     }

     identity ADMIN_DOWN {
       base "tunnel-admin-status";
       description
         "LSP is administratively down";
     }

     identity ADMIN_UP {
       base "tunnel-admin-status";
       description
         "LSP is administratively up";
     }

     // typedef statements
     typedef mpls-label {
       type union {
         type uint32 {
           range 16..1048575;
         }
         type enumeration {
           enum IPV4_EXPLICIT_NULL {
             value 0;
             description "valid at the bottom of the label stack,
```

```
              indicates that stack must be popped and packet forwarded
              based on IPv4 header";
            }
            enum ROUTER_ALERT {
              value 1;
              description "allowed anywhere in the label stack except
              the bottom, local router delivers packet to the local CPU
              when this label is at the top of the stack";
            }
            enum IPV6_EXPLICIT_NULL {
              value 2;
              description "valid at the bottom of the label stack,
              indicates that stack must be popped and packet forwarded
              based on IPv6 header";
            }
            enum IMPLICIT_NULL {
              value 3;
              description "assigned by local LSR but not carried in
              packets";
            }
            enum ENTROPY_LABEL_INDICATOR {
              value 7;
              description "Entropy label indicator, to allow an LSR
              to distinguish between entropy label and applicaiton
              labels RFC 6790";
            }
          }
        }
        description "type for MPLS label value encoding";
        reference "RFC 3032 - MPLS Label Stack Encoding";
      }

      typedef tunnel-type {
        type enumeration {
          enum P2P {
            description "point-to-point label-switched-path";
          }
          enum P2MP {
            description "point-to-multipoint label-switched-path";
          }
          enum MP2MP {
            description "multipoint-to-multipoint label-switched-path";
          }
        }
        description "defines the tunnel type for the LSP";
        reference
          "RFC 6388 - Label Distribution Protocol Extensions for
          Point-to-Multipoint and Multipoint-to-Multipoint Label Switched
```

```
         Paths
         RFC 4875 - Extensions to  Resource Reservation Protocol
         - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE
         Label Switched Paths (LSPs)";
     }

     typedef bandwidth-kbps {
       type uint64;
       units kbps;
     }

     typedef bandwidth-mbps {
       type uint64;
       units mbps;
     }

     typedef bandwidth-gbps {
       type uint64;
       units gbps;
     }

     // grouping statements

     // data definition statements

     // augment statements

     // rpc statements

     // notification statements

   }
                    <CODE ENDS>
```

6.2. MPLS LSP submodules

```
                    <CODE BEGINS> file openconfig-mpls-te.yang
   submodule openconfig-mpls-te {

     yang-version "1";

     belongs-to "openconfig-mpls" {
       prefix "mpls";
     }
```

```
// import some basic types
import ietf-inet-types { prefix inet; }
import openconfig-mpls-rsvp { prefix rsvp; }
import openconfig-mpls-sr { prefix sr; }
import openconfig-mpls-types {prefix mplst; }
import openconfig-types { prefix oc-types; }
import ietf-yang-types { prefix yang; }


// meta
organization "OpenConfig working group";

contact
  "OpenConfig working group
  netopenconfig@googlegroups.com";

description
  "Configuration related to constrained-path LSPs and traffic
  engineering.  These definitions are not specific to a particular
  signaling protocol or mechanism (see related submodules for
  signaling protocol-specific configuration).";

revision "2015-10-04" {
  description
    "Work in progress";
  reference "TBD";
}

// extension statements

// feature statements

// identity statements

// using identities for path comp method, though enums may also
// be appropriate if we decided these are the primary computation
// mechanisms in future.
identity path-computation-method {
  description
   "base identity for supported path computation
    mechanisms";
}

identity locally-computed {
  base path-computation-method;
  description
    "indicates a constrained-path LSP in which the
    path is computed by the local LER";
```

```
    }

    identity externally-queried {
      base path-computation-method;
      description
       "constrained-path LSP in which the path is
        obtained by querying an external source, such as a PCE server";
    }

    identity explicitly-defined {
      base path-computation-method;
      description
       "constrained-path LSP in which the path is
        explicitly specified as a collection of strict or/and loose
        hops";
    }

    // typedef statements

    typedef mpls-hop-type {
      type enumeration {
        enum LOOSE {
          description "loose hop in an explicit path";
        }
        enum STRICT {
          description "strict hop in an explicit path";
        }
      }
      description
       "enumerated type for specifying loose or strict
        paths";
    }

    typedef te-metric-type {
      type union {
        type enumeration {
          enum IGP {
            description
             "set the LSP metric to track the underlying
              IGP metric";
          }
        }
        type uint32;
      }
      description
       "union type for setting the LSP TE metric to a
        static value, or to track the IGP metric";
    }
```

```
     typedef cspf-tie-breaking {
       type enumeration {
         enum RANDOM {
           description
            "CSPF calculation selects a random path among
             multiple equal-cost paths to the destination";
         }
         enum LEAST_FILL {
           description
            "CSPF calculation selects the path with greatest
             available bandwidth";
         }
         enum MOST_FILL {
           description "CSPF calculation selects the path with the least
             available bandwidth";
         }
       }
       default RANDOM;
       description
        "type to indicate the CSPF selection policy when
         multiple equal cost paths are available";
     }


     // grouping statements

     grouping te-tunnel-reoptimize_config {
       description "Definition for reoptimize timer configuration";
       leaf te-lsp-reoptimize-timer {
         type uint16;
         units seconds;
         description
          "frequency of reoptimization of
           a traffic engineered LSP";
       }
     }

     grouping path-placement-constraints {
       description
         "Top level grouping for path placement constraints";

         container admin-groups {
           description
             "Include/Exclude constraints for link affinities";
           uses te-lsp-exclude-admin-group_config;
           uses te-lsp-include-any-admin-group_config;
           uses te-lsp-include-all-admin-group_config;
         }
```

```
    }

    grouping te-tunnel-bandwidth_config {
      description "Bandwidth configuration for TE LSPs";
      choice bandwidth {
        default explicit;
        description
         "select how bandwidth for the LSP will be
          specified and managed";
        case explicit {
          leaf set-bandwidth {
            type uint32;
            description
             "set bandwidth explicitly, e.g., using
              offline calculation";
          }
        }
        case auto {
          uses te-lsp-auto-bandwidth_config;
        }
      }
    }

    grouping te-lsp-auto-bandwidth_config {
      description "Configuration parameters related to autobandwidth";
      container auto-bandwidth {
        description
         "configure auto-bandwidth operation in
          which devices automatically adjust bandwidth to meet
          requirements";

        leaf enabled {
          type boolean;
          default false;
          description
           "enables mpls auto-bandwidth on the
            lsp";
        }

        leaf min-bw {
          type uint32;
          description
           "set the minimum bandwidth in Mbps for an
            auto-bandwidth LSP";
        }

        leaf max-bw {
          type uint32;
```

```
           description
            "set the maximum bandwidth in Mbps for an
             auto-bandwidth LSP";
         }

         leaf adjust-interval {
           type uint32;
           description
            "time in seconds between adjustments to
             LSP bandwidth";
         }

         leaf adjust-threshold {
           type oc-types:percentage;
           description
            "percentage difference between the LSP’s
             specified bandwidth and its current bandwidth
             allocation -- if the difference is greater than the
             specified percentage, auto-bandwidth adjustment is
             triggered";
         }

         container overflow {
           description
            "configuration of MPLS overflow bandwidth
             adjustement for the LSP";
           uses te-lsp-overflow_config;
         }

         container underflow {
           description
            "configuration of MPLS underflow bandwidth
             adjustement for the LSP";
           uses te-lsp-underflow_config;
         }
       }
     }

     grouping  te-lsp-overflow_config {
       description
        "configuration for mpls lsp bandwidth
         overflow adjustment";

       leaf enabled {
         type boolean;
         default false;
         description
          "enables mpls lsp bandwidth overflow
```

```
             adjustment on the lsp";
         }

         leaf overflow-threshold {
           type oc-types:percentage;
           description
            "bandwidth percentage change to trigger
             an overflow event";

         }

         leaf trigger-event-count {
           type uint16;
           description
            "number of consecutive overflow sample
             events needed to trigger an overflow adjustment";
         }
       }

       grouping te-lsp-underflow_config {
         description
           "configuration for mpls lsp bandwidth
           underflow adjustment";

         leaf enabled {
           type boolean;
           default false;
           description
            "enables bandwidth underflow
             adjustment on the lsp";
         }

         leaf underflow-threshold {
           type oc-types:percentage;
           description
            "bandwidth percentage change to trigger
             and underflow event";
         }

         leaf trigger-event-count {
           type uint16;
           description
            "number of consecutive underflow sample
             events needed to trigger an underflow adjustment";
         }
       }

       grouping te-tunnel-metric_config {
```

```
      description "Configuration parameters related to LSP metric";
      leaf metric {
        type te-metric-type;
        description "LSP metric, either explicit or IGP";
      }
    }

    grouping te-lsp-exclude-admin-group_config {
      description
       "Configuration parameters related to admin-groups
        to exclude in path calculation";
      list exclude-groups {

        key exclude-admin-group-name;

        description
          "list of admin-groups to exclude in path calculation";

        leaf exclude-admin-group-name {
          type leafref {
            path "/mpls/te-global-attributes/mpls-admin-groups/" +
              "admin-group/admin-group-name";
          }
          description
            "name of the admin group -- references a defined admin
            group";
        }
      }
    }

    grouping te-lsp-include-all-admin-group_config {
      description
       "Configuration parameters related to admin-groups
        which all must be included in the path calculation";
      list include-all-groups {

        key all-admin-group-name;
        description
          "list of admin-groups of which all must be included";

        leaf all-admin-group-name {
          type leafref {
            path "/mpls/te-global-attributes/mpls-admin-groups/" +
              "admin-group/admin-group-name";
          }
          description
            "name of the admin group -- references a defined
            admin group";
```

```
        }
      }
    }

    grouping te-lsp-include-any-admin-group_config {
      description
       "Configuration parameters related to admin-groups
        of which one must be included in the path calculation";
      list include-any-groups {

        key any-admin-group-name;
        description
          "list of admin-groups of which one must be included";

        leaf any-admin-group-name {
          type leafref {
            path "/mpls/te-global-attributes/mpls-admin-groups/" +
              "admin-group/admin-group-name";
          }
          description
            "name of the admin group -- references a defined
            admin group";
        }
      }
    }

    grouping te-tunnel-protection_config {
      description
       "Configuration parameters related to LSP
        protection";
      leaf protection-style-requested {
        type identityref {
          base mplst:protection-type;
        }
        default mplst:unprotected;
        description
          "style of mpls frr protection desired: can be
          link, link-node or unprotected.";
      }
    }

    grouping te-lsp-comp-explicit {
      description
       "definitions for LSPs in which hops are explicitly
        specified";

      container explicit-path {
        description "LSP with explicit path specification";
```

```
          leaf path-name {
            type leafref {
              path "/mpls/lsps/constrained-path/"
                + "explicit-path/config/named-explicit-paths/name";
              require-instance true;
            }
            description "reference to a defined path";
          }
        }
      }

      grouping te-lsp-comp-queried {
        description "definitons for LSPs computed by querying a remote
          service, e.g., PCE server";

        container queried-path {
          description "LSP with path queried from an external server";

          leaf path-computation-server {
            type inet:ip-address;
            description
             "Address of the external path computation
              server";
          }
        }
      }

      grouping te-lsp-comp-local {
        description "definitons for locally-computed LSPs";

        container locally-computed {
          description "LSP with path computed by local ingress LSR";

          leaf use-cspf {
            type boolean;
            description "Flag to enable CSPF for locally computed LSPs";
          }
          leaf cspf-tiebreaker {
            type cspf-tie-breaking;
            description
              "Determine the tie-breaking method to choose between
              equally desirable paths during CSFP computation";
          }
        }
      }


      grouping explicit-route-subobject {
```

```
        description
          "The explicit route subobject grouping";
        choice type {
          description
            "The explicit route subobject type";
          case ipv4-address {
            description
              "IPv4 address explicit route subobject";
            leaf address {
              type inet:ip-address;
              description "router hop for the LSP path";
            }

            leaf hop-type {
              type mpls-hop-type;
              description "strict or loose hop";
            }
          }

          case label {
            leaf value {
              type uint32;
              description "the label value";
            }
            description
              "The Label ERO subobject";
          }
        }
      }


      // Explicit paths config somewhat following the IETF model
      grouping named-explicit-path_config {
        description
          "Global explicit path configuration
          grouping";
        list named-explicit-paths {
          key "name";
          description
            "A list of explicit paths";
          leaf name {
            type string;
            description
              "A string name that uniquely identifies
              an explicit path";
          }
          list explicit-route-objects {
            key "index";
```

```
          description
            "List of explicit route objects";
          leaf index {
            type uint8 {
              range "0..255";
            }
            description
              "Index of this explicit route object,
               to express the order of hops in path";
          }
          uses explicit-route-subobject;
        }
      }
    }

    grouping explicit-paths-top {
      description
        "common information for MPLS explicit path definition";
        list explicit-path {
          key name;
          description "Explicit path definition";

          leaf name {
            type leafref {
              path "/mpls/lsps/constrained-path/"
                + "explicit-path/config/named-explicit-paths/name";
              require-instance true;
            }
            description "definition for naming an explicit  path";

          }
          container config {
            description "configuration for an explicit path";
            uses named-explicit-path_config;
          }
          container state {
            config false;
            description "operational state for LSP path name";
            uses named-explicit-path_config;
          }
        }
    }

    grouping mpls-te-srlg_config {
      description
        "Configuration of various attributes associated
        with the SRLG";
```

```
        leaf srlg-name {
          type string;
          description "SRLG group identifier";
        }

        leaf srlg-value {
          type uint32;
          description "group ID for the SRLG";
        }

        leaf srlg-cost {
          type uint32;
          description
            "The cost of the SRLG to the computation
            algorithm";
        }
      }

      grouping mpls-te-srlg-members_config {
        description "Configuration of the membership of the SRLG";

        leaf from-address {
          type inet:ip-address;
          description "IP address of the a-side of the SRLG link";
        }

        leaf to-address {
          type inet:ip-address;
          description "IP address of the z-side of the SRLG link";
        }
      }

      grouping mpls-te-srlg-top {
        description
          "Top level grouping for MPLS shared
          risk link groups.";
        container mpls-te-srlg {
          description
            "Shared risk link groups attributes";
          list srlg {
            key srlg-name;
            description "List of shared risk link groups";

            leaf srlg-name {
              type leafref {
                path "../config/srlg-name";
                require-instance true;
              }
```

```
            description "The SRLG group identifier";
          }

          container config {
            description "Configuration parameters related to the SRLG";
            uses mpls-te-srlg_config;
          }

          container state {
            config false;
            description "State parameters related to the SRLG";
            uses mpls-te-srlg_config;
          }

          list members-list {
            key from-address;
            description
              "List of SRLG members, which are expressed
              as IP address endpoints of links contained in the SRLG";

            leaf from-address {
              type leafref {
                path "../config/from-address";
                require-instance true;
              }
              description "The from address of the link in the SRLG";
            }

            container config {
              description
                "Configuration parameters relating to the
                SRLG members";
              uses mpls-te-srlg-members_config;
            }

            container state {
              config false;
              description
                "State parameters relating to the SRLG
                members";
              uses mpls-te-srlg-members_config;
            }
          }
        }
      }
    }

    grouping tunnel-path_config {
```

```
      description
        "Tunnel path properties grouping";
      container path-computation-method {
        description
          "select and configure the way the LSP path is
          computed";

        leaf path-computation {
          type identityref {
            base path-computation-method;
          }
          description "path computation method to use with the LSP";
        }

        uses te-lsp-comp-explicit;
        uses te-lsp-comp-queried;
        uses te-lsp-comp-local;
      }

      uses path-placement-constraints;

      leaf no-cspf {
        type empty;
        description
          "Indicates no CSPF is to be attempted on this
          path.";
      }

      choice sigaling-specific-path-attributes {
        description "Signaling-protocol specific path attributes.";
        case RSVP {
          uses rsvp:rsvp-p2p-path-attributes_config;
        }
        case SR {
          uses sr:sr-path-attributes_config;
        }
      }
    }

    grouping te-tunnel_config {
      description
        "Configuration parameters relevant to a single
         traffic engineered tunnel.";

      leaf name {
        type string;
        description "The tunnel name";
      }
```

```
        leaf type {
          type identityref {
            base mplst:tunnel-type;
          }
          description "Tunnel type, p2p or p2mp";
        }

        leaf local-id {
          type union {
            type uint32;
            type string;
          }
          description
            "locally signficant optional identifier for the
             tunnel; may be a numerical or string value";
        }

        leaf description {
          type string;
          description "optional text description for the tunnel";
        }

        leaf admin-status {
          type identityref {
            base mplst:tunnel-admin-status;
          }
          default mplst:ADMIN_UP;
          description "TE tunnel administrative state.";
        }

        leaf preference {
          type uint8 {
            range "1..255";
          }
          description "Specifies a preference for this tunnel.
            A lower number signifies a better preference";
        }

        uses te-tunnel-metric_config;
        uses te-tunnel-bandwidth_config;
        uses te-tunnel-protection_config;
        uses te-tunnel-reoptimize_config;

        choice signaling-specific-tunnel-attributes {
          description "Signaling-protocol specific path attributes.";
          case RSVP {
            uses rsvp:rsvp-p2p-tunnel-attributes_config;
          }
```

```
      }

      choice tunnel-type {
        description
          "Describes tunnel by type type";
        case p2p {
          leaf destination {
            type inet:ip-address;
            description
              "P2P tunnel destination address";
          }
          /* P2P list of path(s) */
          list primary-paths {
            key "name";
            leaf name {
              type string;
              description "Path name";
            }
            description
              "List of primary paths for this
              tunnel.";
            leaf preference {
              type uint8 {
                range "1..255";
              }
              description
                "Specifies a preference for
                this path. The lower the
                number higher the
                preference";
            }
            uses tunnel-path_config;
          }

          list secondary-paths {
            key "name";
            description
              "List of secondary paths for this
                tunnel.";
            leaf name {
              type string;
              description "Path name";
            }
            leaf preference {
              type uint8 {
                range "1..255";
              }
              description
```

```
                 "Specifies a preference for
                   this path. The lower the
                   number higher the
                   preference";
             }
             uses tunnel-path_config;
           }
         }
         case p2mp {
           // TODO - complete
         }
       }
     }

     grouping te-tunnel_state {
       description
         "Counters and statistical data relevent to a single
          tunnel.";


       leaf oper-status {
         type identityref {
           base mplst:lsp-oper-status;
         }
         description
           "The operational status of the TE tunnel";
       }

       leaf role {
         type identityref {
           base mplst:lsp-role;
         }
         description
           "The lsp role at the current node, whether it is headend,
           transit or tailend.";
       }

       container counters {
         description
           "State data for MPLS label switched paths. This state
           data is specific to a single label switched path.";

         leaf bytes {
           type yang:counter64;
           description
             "Number of bytes that have been forwarded over the
              label switched path.";
         }
```

```
        leaf packets {
          type yang:counter64;
          description
            "Number of pacets that have been forwarded over the
             label switched path.";
        }

        leaf path-changes {
          type yang:counter64;
          description
            "Number of path changes for the label switched path";
        }

        leaf state-changes {
          type yang:counter64;
          description
            "Number of state changes for the label switched path";
        }

        leaf online-time {
          type yang:date-and-time;
          description
            "Indication of the time the label switched path
             transitioned to an Oper Up or in-service state";
        }

        leaf current-path-time {
          type yang:date-and-time;
          description
            "Indicates the time the LSP switched onto its
             current path. This is reset upon a LSP path
             change.";
        }

        leaf next-reoptimization-time {
          type yang:date-and-time;
          description
            "Indicates the next scheduled time the LSP
             will be reoptimized.";
        }
      }
    }

    grouping te-tunnels-top {
      description
        "Top level grouping for TE tunnels";

      list tunnel {
```

```
         key "name type";
         description "List of TE tunnels";

         leaf name {
           type leafref {
             path "../config/name";
             require-instance true;
           }
           description "The tunnel name";
         }

         leaf type {
           type leafref {
             path "../config/type";
             require-instance true;
           }
           description "The tunnel type, p2p or p2mp.";
         }

         container config {
           description
             "Configuration parameters related to TE tunnels:";
           uses te-tunnel_config;
         }

         container state {
           config false;
           description "State parameters related to TE interfaces";
           uses te-tunnel_config;
           uses te-tunnel_state;
         }
       }
     }

   // data definition statements

   // augment statements

   // rpc statements

   // notification statements
   }
                    <CODE ENDS>
```

6.3. MPLS signaling protocol modules

```
                    <CODE BEGINS> file openconfig-mpls-rsvp.yang
module openconfig-mpls-rsvp {

  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/rsvp";

  prefix "rsvp";

  // import some basic types
  import ietf-inet-types { prefix inet; }
  import openconfig-mpls-types { prefix mplst; }
  import ietf-yang-types { prefix yang; }
  import openconfig-types { prefix oc-types; }


  // meta
  organization "OpenConfig working group";

  contact
    "OpenConfig working group
     netopenconfig@googlegroups.com";

  description
    "Configuration for RSVP-TE signaling, including global protocol
     parameters and LSP-specific configuration for constrained-path
     LSPs";

  revision "2015-09-18" {
    description
      "Initial revision";
    reference "TBD";
  }

  // extension statements

  // feature statements

  // identity statements

  // typedef statements

  // grouping statements
```

```
     grouping mpls-rsvp-soft-preemption_config {
       description "Configuration for MPLS soft preemption";
       leaf enable {
         type boolean;
         default false;
         description "Enables soft preemption on a node.";
       }

       leaf soft-preemption-timeout {
         type uint16 {
           range 0..max;
         }
         // The RFC actually recommends 30 seconds as default.
         default 0;
         description
           "Timeout value for soft preemption to revert
            to hard preemption";
         reference "RFC5712 MPLS-TE soft preemption";
       }
     }

     grouping mpls-rsvp-soft-preemption {
       description "Top level group for MPLS soft preemption";
       container soft-preemption {
         description
           "Protocol options relating to RSVP
            soft preemption";
         container config {
           description
             "Configuration parameters relating to RSVP
              soft preemption support";
           uses mpls-rsvp-soft-preemption_config;
         }
         container state {
           config false;
           description
             "State parameters relating to RSVP
              soft preemption support";
           uses mpls-rsvp-soft-preemption_config;
         }
       }
     }

     grouping mpls-rsvp-hellos_config {
       description "RSVP protocol options configuration.";

       leaf hello-interval {
         type uint16 {
```

```
          range 1000..60000;
        }
        units milliseconds;
        default 9000;
        description
          "set the interval in ms between RSVP hello
           messages";
        reference
          "RFC 3209: RSVP-TE: Extensions to RSVP for
           LSP Tunnels.
           RFC 5495: Description of the Resource
           Reservation Protocol - Traffic-Engineered
           (RSVP-TE) Graceful Restart Procedures";
      }

      leaf refresh-reduction {
        type boolean;
        default true;
        description
          "enables all RSVP refresh reduction message
           bundling, RSVP message ID, reliable message delivery
           and summary refresh";
        reference
          "RFC 2961 RSVP Refresh Overhead Reduction
           Extensions";
      }
    }

    grouping mpls-rsvp-hellos {
      description "Top level grouping for RSVP hellos parameters";
      // TODO: confirm that the described semantics are supported
      // on various implementations. Finer grain configuration
      // will be vendor-specific

      container rsvp-hellos {
        description "Top level container for RSVP hello parameters";
        container config {
          description
            "Configuration parameters relating to RSVP
             hellos";
          uses mpls-rsvp-hellos_config;
        }
        container state {
          config false;
          description "State information associated with RSVP hellos";
          uses mpls-rsvp-hellos_config;
        }
      }
```

```
     }

     grouping mpls-rsvp-subscription_config {
       description "RSVP subscription configuration";
       leaf subscription {
         type oc-types:percentage;
         description
           "percentage of the interface bandwidth that
            RSVP can reserve";
       }
     }
     grouping mpls-rsvp-subscription {
       description "Top level group for RSVP subscription options";
       container subscription {
         description
           "Bandwidth percentage reservable by RSVP
            on an interface";
         container config {
           description
             "Configuration parameters relating to RSVP
              subscription options";
           uses mpls-rsvp-subscription_config;
         }
         container state {
           config false;
           description
           "State parameters relating to RSVP
              subscription options";
           uses mpls-rsvp-subscription_config;
         }
       }
     }

     grouping mpls-rsvp-graceful-restart_config {
       description
         "Configuration parameters relating to RSVP Graceful-Restart";

       leaf enable {
         type boolean;
         default false;
         description "Enables graceful restart on the node.";
       }

       leaf restart-time {
         type uint32;
         description
           "Graceful restart time (seconds).";
         reference
```

```
          "RFC 5495: Description of the Resource
           Reservation Protocol - Traffic-Engineered
           (RSVP-TE) Graceful Restart Procedures";
      }
      leaf recovery-time {
        type uint32;
        description
          "RSVP state recovery time";
      }
    }

    grouping mpls-rsvp-graceful-restart {
      description
        "Top level group for RSVP graceful-restart
         parameters";
      container graceful-restart {
        description "TODO";
        container config {
          description
            "Configuration parameters relating to
             graceful-restart";
          uses mpls-rsvp-graceful-restart_config;
        }
        container state {
          config false;
          description
            "State information associated with
             RSVP graceful-restart";
          uses mpls-rsvp-graceful-restart_config;
        }
      }
    }

    grouping mpls-rsvp-authentication_config {
      description "RSVP authentication parameters container.";
      leaf enable {
        type boolean;
        default false;
        description "Enables RSVP authentication on the node.";
      }
      leaf authentication-key {
        type string {
          // Juniper supports 1..16 while
          // Cisco has a much bigger range, up to 60.
          length "1..32";
        }
        description
          "authenticate RSVP signaling
```

```
            messages";
         reference
           "RFC 2747: RSVP Cryptographic Authentication";
       }
     }

     grouping mpls-rsvp-authentication {
       description
         "Top level group for RSVP authentication,
          as per RFC2747";
       container authentication {
         description "TODO";
         container config {
           description
             "Configuration parameters relating
              to authentication";
           uses mpls-rsvp-authentication_config;
         }
         container state {
           config false;
           description
             "State information associated
              with authentication";
           uses mpls-rsvp-authentication_config;
         }
       }
     }

     grouping mpls-rsvp-protection_config {
       description "RSVP facility (link/node) protection configuration";

       leaf link-protection-style-requested {
         type identityref {
           base mplst:protection-type;
         }
         default mplst:link-node-protection-requested;
         description
           "style of mpls frr protection desired:
           link, link-node, or unprotected";
       }

       leaf bypass-optimize-interval {
         type uint16;
         units seconds;
         description
           "interval between periodic optimization
           of the bypass LSPs";
         // note: this is interface specific on juniper
```

```
        // on iox, this is global. need to resolve.
      }
      // to be completed, things like enabling link protection,
      // optimization times, etc.
    }

    grouping mpls-rsvp-link-protection {
      description "Top level group for RSVP protection";
      container protection {
        description "link-protection (NHOP) related configuration";
        container config {
          description "Configuration for link-protection";
          uses mpls-rsvp-protection_config;
        }
        container state {
          config false;
          description "State for link-protection";
          uses mpls-rsvp-protection_config;
        }
      }
    }

    grouping mpls-rsvp-error-statistics {
      description "RSVP-TE packet statistics";
      container error {
        description "RSVP-TE error statistics";
        leaf authentication-failure {
          type yang:counter32;
          description
            "Authentication failure count";
        }

        leaf path-error {
          type yang:counter32;
          description
            "Path error to client count";
        }

        leaf resv-error {
          type yang:counter32;
          description
            "Resv error to client count";
        }

        leaf path-timeout {
          type yang:counter32;
          description
            "Path timeout count";
```

```
        }

        leaf resv-timeout {
          type yang:counter32;
          description
            "Resv timeout count";
        }

        leaf rate-limit {
          type yang:counter32;
          description
            "Count of packets that were rate limited";
        }

        // TODO - complete the other error statistics
      }
    }

    grouping mpls-rsvp-protocol-statistics {
      description "RSVP protocol statistics";
      container protocol {
        description "RSVP-TE protocol statistics";
        leaf hello-sent {
          type yang:counter32;
          description
            "Hello sent count";
        }

        leaf hello-rcvd {
          type yang:counter32;
          description
            "Hello received count";
        }

        leaf path-sent {
          type yang:counter32;
          description
            "Path sent count";
        }

        leaf path-rcvd {
          type yang:counter32;
          description
            "Path received count";
        }

        // TODO - To be completed the other packet statistics
      }
```

```
     }

     grouping mpls-rsvp-statistics {
       description "Top level grouping for RSVP protocol state";
       uses mpls-rsvp-protocol-state;
     }

     grouping rsvp-global {
       description "Global RSVP protocol configuration";
       container rsvp-te {
         description "RSVP-TE global signaling protocol configuration";

         container rsvp-sessions {
           description "Configuration and state of RSVP sessions";

           container config {
             description
               "Configuration of RSVP sessions on the device";
           }

           container state {
             config false;
             description
               "State information relating to RSVP sessions
               on the device";
             uses mpls-rsvp-session-state;
           }
         }

         container rsvp-neighbors {
           description
             "Configuration and state for RSVP neighbors connecting
             to the device";

           container config {
             description "Configuration of RSVP neighbor information";
           }

           container state {
             config false;
             description
               "State information relating to RSVP neighbors";
             uses mpls-rsvp-neighbor-state;
           }
         }

         container global {
           description "Platform wide RSVP configuration and state";
```

```
          uses mpls-rsvp-graceful-restart;
          uses mpls-rsvp-soft-preemption;

          container statistics {
            config false;
            description "Platform wide RSVP state, including counters";
            // TODO - reconcile global and per-interface
            // protocol-related statistics

            container counters {
              config false;
              description
                "Platform wide RSVP statistics and counters";
              uses mpls-rsvp-global-protocol-state;
              uses mpls-rsvp-statistics;
            }
          }
        }

        container interface-attributes {
          // interfaces, bw percentages, hello timers, etc goes here";

          list interface {
            key interface-name;
            description "list of per-interface RSVP configurations";

            // TODO: update to interface ref -- move to separate
            // augmentation.
            leaf interface-name {
              type leafref {
                path "../config/interface-name";
                require-instance true;
              }
              description "references a configured IP interface";
            }


            container config {
              description
                "Configuration of per-interface RSVP parameters";

              leaf interface-name {
                type string;
                description "Name of configured IP interface";
              }
            }

            container state {
```

```
              config false;
              description
                "Per-interface RSVP protocol and state information";
              uses mpls-rsvp-interfaces-state;

              container counters {
                config false;
                description
                  "Interface specific RSVP statistics and counters";
                uses mpls-rsvp-protocol-state;
              }
            }

            uses mpls-rsvp-hellos;
            uses mpls-rsvp-authentication;
            uses mpls-rsvp-subscription;
            uses mpls-rsvp-link-protection;
          }
        }
      }
    }

    grouping rsvp-p2p-tunnel-attributes_config {
      description "properties of RSPP point-to-point paths";

     leaf source {
       type inet:ip-address;
       description
         "tunnel source address";
     }

     leaf soft-preemption {
        type boolean;
        default false;
        description "enables RSVP soft-preemption on this LSP";
      }
    }


    grouping rsvp-p2p-path-attributes_config {
      description "properties of RSPP point-to-point paths";
      leaf setup-priority {
        type uint8 {
          range 0..7;
        }
        default 7;
        description
          "preemption priority during LSP setup, lower is
```

```
            higher priority; default 7 indicates that LSP will not
            preempt established LSPs during setup";
         reference "RFC 3209 - RSVP-TE: Extensions to RSVP for
           LSP Tunnels";
       }

       leaf hold-priority {
         type uint8 {
           range 0..7;
         }
         default 0;
         description
           "preemption priority once the LSP is established,
            lower is higher priority; default 0 indicates other LSPs
            will not preempt the LSPs once established";
         reference "RFC 3209 - RSVP-TE: Extensions to RSVP for
           LSP Tunnels";
       }

       leaf retry-timer {
         type uint16 {
           range 1..600;
         }
         units seconds;
         description
           "sets the time between attempts to establish the
            LSP";
       }
     }

     grouping mpls-rsvp-neighbor-state {
       description "State information for RSVP neighbors";

       list rsvp-neighbor {
         key "neighbor-address";
         description
           "List of RSVP neighbors connecting to the device,
           keyed by neighbor address";

         leaf neighbor-address {
           type inet:ip-address;
           description "Address of RSVP neighbor";
         }

         leaf detected-interface {
           type string;
           description "Interface where RSVP neighbor was detected";
         }
```

```
          leaf neighbor-status {
            type enumeration {
              enum UP {
                description
                  "RSVP hello messages are detected from the neighbor";
              }
              enum DOWN {
                description
                  "RSVP neighbor not detected as up, due to a
                   communication failure or IGP notification
                   the neighbor is unavailable";
              }
            }
            description "Enumuration of possible RSVP neighbor states";
          }

          leaf neighbor-refresh-reduction {
            type boolean;
            description
              "Suppport of neighbor for RSVP refresh reduction";
            reference
              "RFC 2961 RSVP Refresh Overhead Reduction
               Extensions";
          }
        }
      }

      grouping mpls-rsvp-session-state {
        description "State information for RSVP TE sessions";
        list rsvp-session {
          key "source-port destination-port
           source-address destination-address";
          description "List of RSVP sessions";

          leaf source-address {
            type inet:ip-address;
            description "Origin address of RSVP session";
          }

          leaf destination-address {
            type inet:ip-address;
            description "Destination address of RSVP session";
          }

          leaf source-port {
            type uint16;
            description "RSVP source port";
            reference "RFC 2205";
```

```
         }

         leaf destination-port {
           type uint16;
           description "RSVP source port";
           reference "RFC 2205";

         }

         leaf session-state {
           type enumeration {
             enum UP {
               description "RSVP session is up";
             }
             enum DOWN {
               description "RSVP session is down";
             }
           }
           description "Enumeration of RSVP session states";
         }

         leaf session-type {
           type enumeration {
             enum SOURCE {
               description "RSVP session originates on this device";
             }
             enum TRANSIT {
               description "RSVP session transits this device only";
             }
             enum DESTINATION {
               description "RSVP session terminates on this device";
             }
           }
           description "Enumeration of possible RSVP session types";
         }

         leaf tunnel-id {
           type uint16;
           description "Unique identifier of RSVP session";
         }

         leaf label-in {
           type mplst:mpls-label;
           description
             "Incoming MPLS label associated with this RSVP session";
         }

         leaf label-out {
```

```
          type mplst:mpls-label;
          description
            "Outgoing MPLS label associated with this RSVP session";
        }

        leaf-list associated-lsps {
          type leafref {
            path "/mpls/lsps/constrained-path/tunnel/" +
            "config/name";
          }
          description
            "List of label switched paths associated with this RSVP
             session";
        }
      }
    } //rsvp-session-state

    grouping mpls-rsvp-interfaces-state {
      description "RSVP state information relevant to an interface";

      list bandwidth {
        key priority;
        description
          "Available and reserved bandwidth by priority on
           the interface.";

        leaf priority {
          type uint8 {
            range 0..7;
          }
          description
            "RSVP priority level for LSPs traversing the interface";
        }

        leaf available-bandwidth {
          type mplst:bandwidth-mbps;
          description "Bandwidth currently available";
        }

        leaf reserved-bandwidth {
          type mplst:bandwidth-mbps;
          description "Bandwidth currently reserved";
        }
      }

      leaf highwater-mark {
        type mplst:bandwidth-mbps;
        description "Maximum bandwidth ever reserved";
```

```
      }

      leaf active-reservation-count {
        type yang:gauge64;
        description "Number of active RSVP reservations";
      }
    }

    grouping mpls-rsvp-global-protocol-state {
      description "RSVP protocol statistics which may not apply
        on an interface, but are significant globally.";

      leaf path-timeouts {
        type yang:counter64;
        description "TODO";
      }

      leaf reservation-timeouts {
        type yang:counter64;
        description "TODO";
      }

      leaf rate-limited-messages {
        type yang:counter64;
        description "RSVP messages dropped due to rate limiting";
      }
    }

    grouping mpls-rsvp-protocol-state {
      description "RSVP protocol statistics and message counters";
      leaf in-path-messages {
        type yang:counter64;
        description "Number of received RSVP Path messages";
      }

      leaf in-path-error-messages {
        type yang:counter64;
        description "Number of received RSVP Path Error messages";
      }

      leaf in-path-tear-messages {
        type yang:counter64;
        description "Number of received RSVP Path Tear messages";
      }

      leaf in-reservation-messages {
        type yang:counter64;
        description "Number of received RSVP Resv messages";
```

```
        }

        leaf in-reservation-error-messages {
          type yang:counter64;
          description "Number of received RSVP Resv Error messages";
        }

        leaf in-reservation-tear-messages {
          type yang:counter64;
          description "Number of received RSVP Resv Tear messages";
        }

        leaf in-rsvp-hello-messages {
          type yang:counter64;
          description "Number of received RSVP hello messages";
        }

        leaf in-rsvp-srefresh-messages {
          type yang:counter64;
          description "Number of received RSVP summary refresh messages";
        }

        leaf in-rsvp-ack-messages {
          type yang:counter64;
          description
            "Number of received RSVP refresh reduction ack
             messages";
        }

        leaf out-path-messages {
          type yang:counter64;
          description "Number of sent RSVP PATH messages";
        }

        leaf out-path-error-messages {
          type yang:counter64;
          description "Number of sent RSVP Path Error messages";
        }

        leaf out-path-tear-messages {
          type yang:counter64;
          description "Number of sent RSVP Path Tear messages";
        }

        leaf out-reservation-messages {
          type yang:counter64;
          description "Number of sent RSVP Resv messages";
        }
```

```
       leaf out-reservation-error-messages {
         type yang:counter64;
         description "Number of sent RSVP Resv Error messages";
       }

       leaf out-reservation-tear-messages {
         type yang:counter64;
         description "Number of sent RSVP Resv Tear messages";
       }

       leaf out-rsvp-hello-messages {
         type yang:counter64;
         description "Number of sent RSVP hello messages";
       }

       leaf out-rsvp-srefresh-messages {
         type yang:counter64;
         description "Number of sent RSVP summary refresh messages";
       }

       leaf out-rsvp-ack-messages {
         type yang:counter64;
         description
           "Number of sent RSVP refresh reduction ack messages";
       }
     }


     // data definition statements

     // augment statements

     // rpc statements

     // notification statements

   }
                    <CODE ENDS>



                    <CODE BEGINS> file openconfig-mpls-sr.yang
   module openconfig-mpls-sr {

     yang-version "1";

     // namespace
```

```
namespace "http://openconfig.net/yang/sr";

prefix "sr";


// import some basic types
import ietf-inet-types { prefix inet; }
import openconfig-mpls-types { prefix mplst; }

// meta
organization "OpenConfig working group";

contact
  "OpenConfig working group
  netopenconfig@googlegroups.com";

description
  "Configuration for MPLS with segment routing-based LSPs,
  including global parameters, and LSP-specific configuration for
  both constrained-path and IGP-congruent LSPs";

revision "2015-10-14" {
  description
    "Work in progress";
  reference "TBD";
}

// extension statements

// feature statements

// identity statements

// typedef statements

grouping srgb_config {

  // Matches the "global" configuration options in
  // draft-litkowski-spring-yang...
  // TODO: request to Stephane for this to be a separate
  // grouping such that it can be included.

  leaf lower-bound {
    type uint32;
    description
      "Lower value in the block.";
  }
  leaf upper-bound {
```

```
          type uint32;
          description
            "Upper value in the block.";
        }
        description
          "List of global blocks to be advertised.";
      }

      grouping srgb_state {
        description
          "State parameters relating to the SRGB";

        leaf size {
          type uint32;
          description
            "Number of indexes in the SRGB block";
        }
        leaf free {
          type uint32;
          description
            "Number of SRGB indexes that have not yet been allocated";
        }
        leaf used {
          type uint32;
          description
            "Number of SRGB indexes that are currently allocated";
        }

        // TODO: where do we put LFIB entries?

      }

      grouping adjacency-sid_config {
        description
          "Configuration related to an Adjacency Segment Identifier
          (SID)";

        // tuned from draft-litkowski-spring-yang
        // TODO: need to send a patch to Stephane

        leaf-list advertise {
          type enumeration {
            enum "PROTECTED" {
              description
                "Advertise an Adjacency-SID for this interface, which is
                eligible to be protected using a local protection
                mechanism on the local LSR. The local protection
                mechanism selected is dependent upon the configuration
```

```
                 of RSVP-TE FRR or LFA elsewhere on the system";
             }
             enum UNPROTECTED {
               description
                 "Advertise an Adajcency-SID for this interface, which is
                 explicitly excluded from being protected by any local
                 protection mechanism";
             }
           }
           description
             "Specifies the type of adjacency SID which should be
             advertised for the specified entity.";
         }

         leaf-list groups {
           type uint32;
           description
             "Specifies the groups to which this interface belongs.
             Setting a value in this list results in an additional AdjSID
             being advertised, with the S-bit set to 1. The AdjSID is
             assumed to be protected";
         }
       }

       grouping interface_config {
         description
           "Configuration parameters relating to a Segment Routing
           enabled interface";

         leaf interface {
           type string;
           // TODO: this should be changed to a leafref.
           description
             "Reference to the interface for which segment routing
             configuration is to be applied.";
         }
       }

       // grouping statements

       grouping sr-global {
         description "global segment routing signaling configuration";

         container segment-routing {
           description "SR global signaling config";

           list srgb {
             key "lower-bound upper-bound";
```

```
          uses srgb_config;
          container config {
            description
              "Configuration parameters relating to the Segment Routing
              Global Block (SRGB)";
            uses srgb_config;
          }
          container state {
            config false;
            description
              "State parameters relating to the Segment Routing Global
              Block (SRGB)";
            uses srgb_config;
            uses srgb_state;
          }
          description
            "List of Segment Routing Global Block (SRGB) entries. These
            label blocks are reserved to be allocated as domain-wide
            entries.";
        }

        list interfaces {
          key "interface";
          uses interface_config;
          container config {
            description
              "Interface configuration parameters for Segment Routing
              relating to the specified interface";
            uses interface_config;
          }
          container state {
            config false;
            description
              "State parameters for Segment Routing features relating
              to the specified interface";
            uses interface_config;
          }
          container adjacency-sid {
            description
              "Configuration for Adjacency SIDs that are related to
              the specified interface";
            container config {
              description
                "Configuration parameters for the Adjacency-SIDs
                that are related to this interface";
              uses adjacency-sid_config;
            }
            container state {
```

```
              config false;
              description
                "State parameters for the Adjacency-SIDs that are
                related to this interface";
              uses adjacency-sid_config;
            }
          }
          description
            "List of interfaces with associated segment routing
            configuration";
        }
      }
    }

    grouping sr-path-attributes_config {
      description
        "Configuration parameters relating to SR-TE LSPs";

      leaf sid-selection-mode {
        type enumeration {
          enum "ADJ-SID-ONLY" {
            description
              "The SR-TE tunnel should only use adjacency SIDs
              to build the SID stack to be pushed for the LSP";
          }
          enum "MIXED-MODE" {
            description
              "The SR-TE tunnel can use a mix of adjacency
              and prefix SIDs to build the SID stack to be pushed
              to the LSP";
          }
        }
        default "MIXED-MODE";
        description
          "The restrictions placed on the SIDs to be selected by the
          calculation method for the SR-TE LSP";
      }

      leaf sid-protection-required {
        type boolean;
        default "false";
        description
          "When this value is set to true, only SIDs that are
          protected are to be selected by the calculating method
          for the SR-TE LSP.";
      }
    }
```

```
     grouping sr_fec-address_config {
       description
         "Configuration parameters relating to a FEC that is to be
         advertised by Segment Routing";

       leaf fec-address {
         type inet:ip-prefix;
         description
           "FEC that is to be advertised as part of the Prefix-SID";
       }
     }

     grouping sr_fec-prefix-sid_config {
       description
         "Configuration parameters relating to the nature of the
         Prefix-SID that is to be advertised for a particular FEC";

       leaf type {
         type enumeration {
           enum "INDEX" {
             description
               "Set when the value of the prefix SID should be specified
               as an off-set from the SRGB's zero-value. When multiple
               SRGBs are specified, the zero-value is the minimum
               of their lower bounds";
           }
           enum "ABSOLUTE" {
             description
               "Set when the value of a prefix SID is specified as the
               absolute value within an SRGB. It is an error to specify
               an absolute value outside of a specified SRGB";
           }
         }
         default "INDEX";
         description
           "Specifies how the value of the Prefix-SID should be
           interpreted - whether as an offset to the SRGB, or as an
           absolute value";
       }

       leaf node-flag {
         type boolean;
         description
           "Specifies that the Prefix-SID is to be treated as a Node-SID
           by setting the N-flag in the advertised Prefix-SID TLV in the
           IGP";
       }
```

```
    leaf last-hop-behavior {
      type enumeration {
        enum "EXPLICIT-NULL" {
          description
            "Specifies that the explicit null label is to be used
            when the penultimate hop forwards a labelled packet to
            this Prefix-SID";
        }
        enum "UNCHANGED" {
          description
            "Specicies that the Prefix-SID’s label value is to be
            left in place when the penultimate hop forwards to this
            Prefix-SID";
        }
        enum "PHP" {
          description
            "Specicies that the penultimate hop should pop the
            Prefix-SID label before forwarding to the eLER";
        }
      }
      description
        "Configuration relating to the LFIB actions for the
        Prefix-SID to be used by the penultimate-hop";
    }
  }


  grouping igp-tunnel-sr {
    description "defintiions for SR-signaled, IGP-based LSP tunnel
    types";

    container tunnel {
      description "contains configuration stanzas for different LSP
      tunnel types (P2P, P2MP, etc.)";

      leaf tunnel-type {
        type mplst:tunnel-type;
        description "specifies the type of LSP, e.g., P2P or P2MP";
      }

      container p2p-lsp {
        when "tunnel-type = ’P2P’" {
          description "container active when LSP tunnel type is
          point to point";
        }
        description "properties of point-to-point tunnels";

        list fec {
```

```
            key "fec-address";
            uses sr_fec-address_config;

            description
              "List of FECs that are to be originated as SR LSPs";

            container config {
              description
                "Configuration parameters relating to the FEC to be
                advertised by SR";
              uses sr_fec-address_config;
            }
            container state {
              config false;
              description
                "Operational state relating to a FEC advertised by SR";
              uses sr_fec-address_config;
            }
            container prefix-sid {
              description
                "Parameters relating to the Prefix-SID
                used for the originated FEC";

              container config {
                description
                  "Configuration parameters relating to the Prefix-SID
                  used for the originated FEC";
                uses sr_fec-prefix-sid_config;
              }
              container state {
                config false;
                description
                  "Operational state parameters relating to the
                  Prefix-SID used for the originated FEC";
                uses sr_fec-prefix-sid_config;
              }
            }
          }
        }
      }
    }

    grouping igp-lsp-sr-setup {
      description "grouping for SR-IGP path setup for IGP-congruent
      LSPs";

      container segment-routing {
```

```
          presence "Presence of this container sets the LSP to use
          SR signaling";

          description "segment routing signaling extensions for
          IGP-confgruent LSPs";

          uses igp-tunnel-sr;

        }
      }

      // data definition statements

      // augment statements

      // rpc statements

      // notification statements

   }
                     <CODE ENDS>



                     <CODE BEGINS> file openconfig-mpls-ldp.yang
   module openconfig-mpls-ldp {

      yang-version "1";

      // namespace
      namespace "http://openconfig.net/yang/ldp";

      prefix "ldp";

      // import some basic types
      import ietf-inet-types { prefix inet; }
      import openconfig-mpls-types { prefix mplst; }


      // meta
      organization "OpenConfig working group";

      contact
        "OpenConfig working group
        netopenconfig@googlegroups.com";

      description
        "Configuration of Label Distribution Protocol global and LSP-
```

```
       specific parameters for IGP-congruent LSPs";

     revision "2015-07-04" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     // grouping statements

      grouping ldp-global {
       description "global LDP signaling configuration";

       container ldp {
         description "LDP global signaling configuration";

         container timers {
           description "LDP timers";
         }
       }
     }


     grouping igp-tunnel-ldp {
       description "common defintiions for LDP-signaled LSP tunnel
       types";

       container tunnel {
         description "contains configuration stanzas for different LSP
         tunnel types (P2P, P2MP, etc.)";

         leaf tunnel-type {
           type mplst:tunnel-type;
             description "specifies the type of LSP, e.g., P2P or P2MP";
         }

         leaf ldp-type {
           type enumeration {
             enum BASIC {
               description "basic hop-by-hop LSP";
```

```
            }
            enum TARGETED {
              description "tLDP LSP";
            }
          }
          description "specify basic or targeted LDP LSP";
        }

        container p2p-lsp {
          when "tunnel-type = 'P2P'" {
            description "container active when LSP tunnel type is
            point to point";
          }

          description "properties of point-to-point tunnels";

          leaf-list fec-address {
            type inet:ip-prefix;
            description "Address prefix for packets sharing the same
            forwarding equivalence class for the IGP-based LSP";
          }
        }

        container p2mp-lsp {
          when "tunnel-type = 'P2MP'" {
            description "container is active when LSP tunnel type is
            point to multipoint";
          }

          description "properties of point-to-multipoint tunnels";

          // TODO: specify group/source, etc.

        }

        container mp2mp-lsp {
          when "tunnel-type = 'MP2MP'" {
            description "container is active when LSP tunnel type is
            multipoint to multipoint";
          }

          description "properties of multipoint-to-multipoint tunnels";

          // TODO: specify group/source, etc.

        }
      }
    }
```

```
     grouping igp-lsp-ldp-setup {
       description "grouping for LDP setup attributes";

       container ldp {

         presence "Presence of this container sets the LSP to use
         LDP signaling";

         description "LDP signaling setup for IGP-congruent LSPs";

         // include tunnel (p2p, p2mp, ...)

         uses igp-tunnel-ldp;

       }
     }

     // data definition statements

     // augment statements

     // rpc statements

     // notification statements

   }
                  <CODE ENDS>



                  <CODE BEGINS> file openconfig-mpls-igp.yang
   submodule openconfig-mpls-igp {

     yang-version "1";

     belongs-to "openconfig-mpls" {
       prefix "mpls";
     }


     // import some basic types
     import openconfig-mpls-ldp { prefix ldp; }
     import openconfig-mpls-sr { prefix sr; }



     // meta
     organization "OpenConfig working group";
```

```
     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "Configuration generic configuration parameters for IGP-congruent
       LSPs";

     revision "2015-07-04" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     // grouping statements


     grouping igp-lsp-common {
       description "common definitions for IGP-congruent LSPs";

       // container path-attributes {
       //  description "general path attribute settings for IGP-based
       //  LSPs";

       //}

     }


     grouping igp-lsp-setup {
       description "signaling protocol definitions for IGP-based LSPs";

       container path-setup-protocol {
         description "select and configure the signaling method for
             the LSP";

         // uses path-setup-common;
         uses ldp:igp-lsp-ldp-setup;
         uses sr:igp-lsp-sr-setup;
       }
```

```
     }


     // data definition statements

     // augment statements

     // rpc statements

     // notification statements

   }
                   <CODE ENDS>



                   <CODE BEGINS> file openconfig-mpls-static.yang
   submodule openconfig-mpls-static {

     yang-version "1";

     belongs-to "openconfig-mpls" {
       prefix "mpls";
     }

     // import some basic types
     import openconfig-mpls-types {prefix mplst; }
     import ietf-inet-types { prefix inet; }


     // meta
     organization "OpenConfig working group";

     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "Defines static LSP configuration";

     revision "2015-07-04" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements
```

```
   // feature statements

   // identity statements

   // typedef statements

   // grouping statements

   grouping static-lsp-common {
     description "common definitions for static LSPs";

     leaf next-hop {
       type inet:ip-address;
       description "next hop IP address for the LSP";
     }

     leaf incoming-label {
       type mplst:mpls-label;
       description "label value on the incoming packet";
     }

     leaf push-label {
       type mplst:mpls-label;
       description "label value to push at the current hop for the
       LSP";
     }
   }

   grouping static-lsp-main {
     description "grouping for top level list of static LSPs";


     list label-switched-path {
       key name;
       description "list of defined static LSPs";

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

       // TODO: separation into ingress, transit, egress may help
       // to figure out what exactly is configured, but need to
       // consider whether implementations can support the
       // separation
       container ingress {
         description "Static LSPs for which the router is an
           ingress node";
```

```
          uses static-lsp-common;
        }

        container transit {
          description "static LSPs for which the router is a
            transit node";

          uses static-lsp-common;
        }

        container egress {
          description "static LSPs for which the router is a
            egress  node";

          uses static-lsp-common;
        }
      }
    }

    // data definition statements

    // augment statements

    // rpc statements

    // notification statements

  }
                    <CODE ENDS>
```

7. Contributing Authors

The following people contributed significantly to this document and are listed below:

Ina Minei
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US
Email: inaminei@google.com

Anees Shaikh
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: aashaikh@google.com

Phil Bedard
Cox Communications
Atlanta, GA 30319
US
Email: phil.bedard@cox.com

8. Acknowledgements

The authors are grateful for valuable contributions to this document and the associated models from: Ebben Aires, Deepak Bansal, Nabil Bitar, Feihong Chen, Mazen Khaddam.

9. References

[I-D.ietf-idr-bgp-model]
Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Alex, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-ietf-idr-bgp-model-00 (work in progress), July 2015.

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., rjs@rob.sh, r., Tantsura, J., and E. Crabbe, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-02 (work in progress), October 2015.

[I-D.openconfig-netmod-model-structure]
Shaikh, A., Shakir, R., D'Souza, K., and L. Fang, "Operational Structure and Organization of YANG Models", draft-openconfig-netmod-model-structure-00 (work in progress), March 2015.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

[I-D.shaikh-idr-bgp-model]
Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Alex, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-shaikh-idr-bgp-model-02 (work in progress), June 2015.

[I-D.shakir-rtgwg-sr-performance-engineered-lsps]
Shakir, R., Vernals, D., and A. Capello, "Performance Engineered LSPs using the Segment Routing Data-Plane", draft-shakir-rtgwg-sr-performance-engineered-lsps-00 (work in progress), July 2013.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC5443] Jork, M., Atlas, A., and L. Fang, "LDP IGP Synchronization", RFC 5443, DOI 10.17487/RFC5443, March 2009, <http://www.rfc-editor.org/info/rfc5443>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

Authors' Addresses

Joshua George
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: jgeorge@google.com

Luyuan Fang
Microsoft
15590 NE 31st St
Redmond, WA 98052
US

Email: lufang@microsoft.com

Eric Osborne
Level 3

Email: eric.osborne@level3.com

Rob Shakir
Jive Communications, Inc.
1275 West 1600 North, Suite 100
Orem, UT 84057

Email: rjs@rob.sh

TEAS Working Group   Ravi Singh
Internet Draft   Juniper Networks
Intended status: Best Current Practice   Rob Shakir
                                                         British Telecom
                                                     Vishnu Pavan Beeram
                                                        Juniper Networks
                                                              Tarek Saad
                                                           Cisco Systems

Expires: January 2, 2016   July 2, 2015

RSVP Setup Retry - BCP
draft-ravisingh-teas-rsvp-setup-retry-01

Status of this Memo

Copyright Notice

Abstract

This document discusses the best current practices associated with the implementation of RSVP setup-retry timer.

Conventions used in this document

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

Table of Contents

1. Introduction

In an RSVP-TE network with a very large number of LSPs, link/node failure(s) may produce a noticeable increase in RSVP-TE control traffic. As a result, RSVP-TE messages might get delayed by virtue of being stuck in a queue that is overwhelmed with messages to be sent or they might get lost forever. For example, a Path message intended to be sent by a transit router might be stuck in the output queue to be sent to the next-hop. Alternately, it might have got dropped on the receive side due to queue overflows. The same could happen for a Resv message in the reverse direction. Also, in the absence of reliable delivery of Path-Error messages [RFC2961], an error that gets generated at transit/egress for an LSP that is in the process of being setup may never make it to the ingress.

Lost/delayed RSVP-TE messages cause the following problems for an ingress router:

- In the absence of an error indication, how is an ingress to know that an LSP for which signaling was (re-)initiated and a Resv has not yet been received, is ever going to come up?
- In the absence of any indication, what action should the ingress take to support low-latency LSP-setup?

The above problems essentially boil-down to: how long should the ingress continue to wait before giving up on its attempt to bring up the LSP, and take some alternative course of action (e.g., try to bring up the LSP on an alternate path)?. To mitigate this problem, some implementations use a setup-retry timer mechanism. This document discusses the issues associated with a particular implementation of this timer and makes some specific recommendations to get around these issues.

2. Setup-Retry Timer

The setup-retry timer is usually a configurable timer which (in the absence of an error indication) goes off when an LSP with a given LSPID has not received the corresponding Resv in response to its Path during a pre-configured duration after its first Path had been sent.

Use of the setup-retry timer is based on the presumption that if signaling for a given LSP has not been completed within an "expected" duration, it is not going to be completed at all. The intent in the use of this timer is to expeditiously take some alternative course of action when an LSP has not yet completed its signaling within an "expected" duration of time.

3. Possible ill-effects due to implementation choices

As mentioned in the previous section, the intent in the use of this timer is to take some alternative course of action when an LSP has not yet completed its signaling within an "expected" duration of time. One such course of action is for the ingress router to initiate tear-down for the previously in-the-process-of-being-signaled path via a PathTear; run CSPF; and use the outcome of this CSPF to signal the brand-new path for this tunnel with a different LSP-ID, typically, bumped up by 1. This section describes the problems caused by such course of action.

As mentioned in Section 1, in a network with a very large number of RSVP-TE LSPs, link/node failure(s) may produce a noticeable increase in the volume of RSVP-TE control traffic, which in turn might cause a router to either drop RSVP-TE messages or alternately cause them to be sent excessively late.

As a result, the following problems can occur:
- LSP setup latency might be excessively high.
- Error messages that indicate failure in LSP setup might not make it to the ingress router.

A mix of the above problems can cause the setup-retry timer for a given LSP (at the ingress router) to fire repeatedly over a period of time. The situation being such the ingress gets stuck in a cycle as illustrated below for some/many LSPs:

```
   --------------------------------------------------------------------
   Ingress Timeline        | [Ingress]---[]---[]...[Transit]...[]---[]-
   ------------------------|
   1. Trigger LSP setup    | Path
             :             |   TNL-ID=X
             :             |   LSP-ID=Y
             :             | -------->
      <No Resv (X, Y)>     |          ------------> Path (X, Y)
             :             |                        -------> --------->
             :             |                  :
             :             |                  :
   2. Setup-Retry Timer    |                  :
      fires; Recompute     |                  :
      path;                |                  :
   3. Trigger Teardown     | PathTear
                           |   TNL-ID=X
                           |   LSP-ID=Y
                           | -------->
                           |          ------------> PathTear (X, Y)
                           |                        -------> --------->
   4. Trigger setup for new| Path
      instance of the LSP  |   TNL-ID=X
      (same ERO)           |   LSP-ID=Y+1
             :             | -------->
             :             |          ------------> Path (X, Y+1)
             :             |                        -------> --------->
             :             |                                 Resv
      <No Resv (X, Y+1)>   |                                   TNL-ID=X
             :             |                                   LSP-ID=Y
             :             |                                 <---------
             :             |                                 ResvError
             :             |                                   No Path
```

```
          :              |                                 --------->
   5. Repeat loop through |                  :
      2-4                 |                  :
   --------------------------------------------------------------------
```

In the above illustration, notice how the transit router never gets to completely process the "current" LSP-ID (see [RShakir] for more). The implementation recommendations made in this document will help avoid this snowball effect.

4. Causes of the above ill-effects

The implementation issues listed in section 3 end up causing an increase in the control plane load on a network whose control plane is already under stress. The foregoing is caused by unnecessarily doing the following even when there is no change in the computed path:

- Sending PathTears causes excessive and unjustifiable work on those downstream routers on the "previous ERO path" that had managed to bring the LSP UP. In other words, the slowness of a given transit router should not be the cause to penalize all other transit routers downstream of it, as doing so just increases the overall network stress.

- Sending Path for LSPID=Y+1 causes unnecessary work for all routers on the ERO path including those that were already running slow and were the real cause of the Resv for LSPDID=Y not having been received timely by the ingress.

5. Solution to the implementation issues

To eliminate causes of the ill-effects listed in the previous section and thus to eliminate the ill-effects, this document makes the following recommendations.

When the setup-retry timer fires:

If there is no change in the computed path (no error indication for that LSP has been received via a PathErr or a TE update indicating a failure),
- Do not send PathTear for LSPID=Y
- Just let the Path State get refreshed for LSPID=Y.

The recommended default behavior is to keep retrying until the path changes or the user intervenes. Implementations MAY choose to provide the user with an option to override this default behavior and specify a policy to determine when to stop retrying.

Implementations SHOULD use the recommendations listed in this section to avoid getting stuck in a LSP signaling hysteresis.

6. Security Considerations

This document does not introduce any new security concerns.

7. IANA Considerations

None.

8. Normative References

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

[RShakir] Rob Shakir, "The next spring forward", http://rob.sh/files/the-next-spring-forward_rjs120314.pdf March 2014.

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

9. Acknowledgments

The authors would like to thank Yakov Rekhter and Raveendra Torvi for their inputs.

10. Authors' Addresses

Ravi Singh
Juniper Networks
Email: ravis@juniper.net

Rob Shakir
British Telecom
Email: rob.shakir@bt.com

Tarek Saad
Cisco Systems
Email: tsaad@cisco.com

Vishnu Pavan Beeram

Juniper Networks
Email: vbeeram@juniper.net

Contributors

Markus Jork
Juniper Networks
Email: mjork@juniper.net

Aman Kapoor
Juniper Networks
Email: amanka@juniper.net

TEAS Working Group T. Saad, Ed.
Internet-Draft R. Gandhi
Intended status: Standards Track Cisco Systems Inc
Expires: January 7, 2016 X. Liu
Ericsson
V. Beeram
Juniper Networks
H. Shah
Ciena
X. Chen
Huawei Technologies
R. Jones
Brocade
B. Wen
Comcast
July 06, 2015

A YANG Data Model for Traffic Engineering Tunnels and Interfaces
draft-saad-teas-yang-te-02

Abstract

This document defines a YANG data model for the configuration and management of Traffic Engineering (TE) interfaces and tunnels. The model defines generic data that is reusable across multiple data and control plane protocols.

The data model covers the configuration, operational state, remote procedural calls, and event notifications data for TE data.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interface, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage TE interfaces and P2P or P2MP TE tunnels. This data model restricts to TE generic data that is control and data plane agnostic. It is expected that other protocol and data plane specific modules (e.g. RSVP-TE [RFC3209]) will augment this TE model.

1.1. Terminology

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

1.2. Tree Diagram

A simplified graphical representation of the data model is presented in each section of the model. The following notations are used for the YANG model data tree representation.

<status> <flags> <name> <opts> <type>

<status> is one of:
+ for current
x for deprecated
o for obsolete

<flags> is one of:
rw for read-write configuration data
ro for read-only non-configuration data
-x for execution rpcs
-n for notifications

<name> is the name of the node

If the node is augmented into the tree from another module, its name is printed as <prefix>:<name>

<opts> is one of:
? for an optional leaf or node
! for a presence container
* for a leaf-list or list
Brackets [<keys>] for a list’s keys
Curly braces {<condition>} for optional feature that make node conditional
Colon : for marking case nodes
Ellipses ("...") subtree contents not shown

Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

<type> is the name of the type for leafs and leaf-lists.

1.3. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are prefixed using the standard prefix associated with the corresponding YANG imported modules, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and corresponding YANG modules

1.4. Open Issues and Next Steps

This document describes the YANG data model for the TE generic and helper modules. It also describes the high-level relationship between these modules and to other external protocol modules. The current revision of the draft focuses on configuration and state data aspects of the model. It is expected that the future revisions will cover RPC, and notification aspects.

Also, the models that define technology specific extensions to the generic TE model (e.g. OTN [RFC4328] TE extensions), are expected to be addressed in separate documents.

1.4.1. State Data Organization

Pure state data (for example, ephemeral or protocol derived state objects) can be modeled using one of the options below:

- o Contained inside the read-write container, under the "state" sub-container, as shown in Figure 3
- o Contained inside a separate read-only container, for example a tunnels-state container

The first option allows for the reusing of the containers that hold configuration data (in the "config" sub-container), and by placing state data under the read-only "state" sub-container of the parent container. However, when adopting this approach for ephemeral or purely derived states (e.g. auto tunnels), and since in this case the state hangs off the read-write parent container, it will be possible to delete the parent container and subsequently the ephemeral read-only state contained within (see Figure 3).

The second option entails defining a new read-only parent container in the model (e.g. neighbors-state) that holds the data.

This revision of the draft adopts the first option. Further discussions on this topic are expected to close on the best choice to adopt.

2. Data Model Overview

Although the basis of TE elements remain similar across different vendor implementations, however, the details of a TE model will usually vary across different vendor implementations. Also, implementations may vary in their support of the complete set of TE features. The TE YANG module defined in this document is an attempts to define a vendor agnostic model that will prescribe to IETF standard terminology when different representation of data is possible.

The model is composed of common building blocks that are independent of specific data or control plane instantiations. It covers data representation for the configuration, state, remote procedural calls (RPCs), and event notifications.

Throughout the model, the approach described in [I-D.openconfig-netmod-opstate] is adopted to represent data pertaining to configuration intended state, applied state and derived state data elements. Each container in the model hold a "config" and "state" sub-container. The "config" sub-container is used to represent the intended configurable parameters, and the state sub-container is used to represent both the applied configurable parameters and any derived state, such as counters or statistical information.

The decision to use this approach was made to better align with the MPLS consolidated model in [I-D.openconfig-mpls-consolidated-model], and maximize reusability of groupings defined in this document and allow for possible convergence between the two models.

2.1. Design Objectives

The goal of this document is to define a TE data model that can represent different TE vendor implementations, while adhering to standard terminology and behavior when resolving differences in implementations.

The following considerations with respect data organization are taken into account when defining the model:

- o reusable data elements are grouped into separate TE types module(s) that can be readily imported by other modules whenever needed

- o reusable TE data types that are data plane independent are grouped in the TE generic types module "ietf-te-types.yang"

- o reusable TE data elements that are data plane specific (e.g. packet PSC or switching technologies as defined in [RFC3473]) are expected to be grouped in a technology- specific types module, e.g. "ietf-te-psc-types.yang". It is expected that technology specific types will augment TE generic types as shown in Figure 1

```
             +---------+
             | ietf-te |       ^: import
             +---------+       o: augment
           import ^
                  |
                  |
            +---------------+
            | ietf-te-types |
            +---------------+
               o          o
               |           \
               |            \
       +-------------------+  +-------------------+
       | ietf-te-psc-types |  | ietf-te-otn-types |
       +-------------------+  +-------------------+
                               (shown for illustration
                                not in this document)
```

Figure 1: Relationship between generic and technology specific TE types modules

- o TE generic module includes data elements that are control plane independent. Data elements specific to a control plane protocol (e.g. RSVP-TE [RFC3209]) are expected to be in a separate module that augments the TE generic module. It is also expected that data relevant to a specific instantiations of data plane technology will exist in a separate YANG module that augments the TE generic model, see Figure 2.

```
     TE generic     +---------+        ^: import
     module         | ietf-te |        o: augment
                    +---------+
                       |   o
                       |   |
                       v   |
                    +--------------+
     RSVP-TE module | ietf-rsvp-te |o . . .
                    +--------------+         \
                       ^   |                  \
                       |   o                +-------------------+
                    +-----------+           | ietf-rsvp-otn-te  |
     RSVP module    | ietf-rsvp |           +-------------------+
                    +-----------+              RSVP-TE with OTN
                                               extensions
                                              (shown for illustration
                                               not in this document)
```

Figure 2: Relationship of TE module with other control plane protocol modules

o In general, little information in the model is designated as "mandatory", to allow freedom to vendors to adapt the data model to their specific product implementation.

2.2. Optional Features

Optional features are features beyond the generic TE model, and hence, it is up to a vendor to decide whether or not to support of a particular feature on a particular device.

This module declares a number of TE functions as features (such as P2MP-TE, soft-preemption etc.). It is intended that vendors will extend this features list.

2.3. Configuration Inheritance

The defined data model supports configuration inheritance for tunnels, paths, and interfaces. Data elements defined in the main container (e.g. that encompasses the list of tunnels, interfaces, or paths) are assumed to apply equally to all elements of the list, unless overridden explicitly for a certain element (e.g. tunnel, interface or path). Vendors are expected to augment the above container(s) to provide the list of inheritance command for their implementations.

2.4. Vendor Configuration Models

There two main popular types of routing protocol configuration that vendors may support:

- o protocol centric - all the protocol related configuration is contained within the protocol itself. Configuration belonging to multiple instances of the protocol running in different routing-instances (e.g. VRFs) are contained under the default routing instance [I-D.ietf-netmod-routing-cfg]:

- o VRF centric - all the protocol related configuration for a routing- instance is contained within this routing-instance.

On-going discussions within the IETF community have converged on adopting the VRF centric approach. The proposed model in this document adheres to this conclusion.

3. TE Generic Model Organization

This model covers configuration, state, RPC, and notifications data pertaining to TE global parameters, interfaces, and tunnels parameters.

The container "te" is the top level container in this data model. The presence of this container is expected to enable TE function system wide.

The approach described in [I-D.openconfig-netmod-opstate] allows for modeling the intended and respective applied and derived state. The TE state data in this model falls into one of the following categories:

- o State corresponding to applied configuration

- o State corresponding to derived state, counters, stats, etc.

- o State corresponding to ephemeral data (e.g. LSPs, auto-tunnels, etc.)

Data for the first two categories are contained under the respective "state" sub-container of the intended object (e.g. tunnel). The last category falls under a separate - e.g. lsps-state- container that contains the attributes of a purely derived state data (e.g. ephemeral objects) that are not associated with any configuration as shown in Figure 3.

```
module: ietf-te
   +--rw te!
      +--rw globals
            +-- rw config
               <<intended configuration>>
            .
            +-- ro state
               <<applied configuration>>
               <<derived state associated with the tunnel>>
         .
         .
      +--rw interfaces
            +-- rw config
               <<intended configuration>>
            .
            +-- ro state
               <<applied configuration>>
               <<derived state associated with the tunnel>>
         .
         .
      +--rw tunnels
            +-- rw config
               <<intended configuration>>
            .
            +-- ro state
               <<applied configuration>>
               <<derived state associated with the tunnel>>
         .
         .
      +--ro tunnels-state
           <<ephemeral tunnels>>
rpcs:
   +---x globals-rpc
   +---x interfaces-rpc
   +---x tunnels-rpc
notifications:
   +---n globals-notif
   +---n interfaces-notif
   +---n tunnels-notif
```

Figure 3: TE highlevel model view

3.1. Global Configuration and State Data

This branch of the data model covers configurations that control TE features behavior system-wide, and its respective state. Examples of such configuration data are:

o Table of named SRLG mappings

o Table of named (extended) administrative groups mappings

o Table of named explicit paths to be referenced by TE tunnels

o Table of named path-constraints sets

o Auto-bandwidth global parameters

o TE diff-serve TE-class maps

o System-wide capabilities for LSP reoptimization

   * Reoptimization timers (periodic interval, LSP installation and cleanup)

o System-wide capabilities for TE state flooding

   * Periodic flooding interval

o System-wide capabilities that affect the originating, traversing and terminating LSPs. For example:

   * Path selection parameters (e.g. metric) at head-end LSR

   * Path protection parameters at head-end LSR

   * (Soft) preemption parameters

   * Fast reroute parameters

The approach described in [I-D.openconfig-netmod-opstate] is utilised to include the global state data under the global "state" sub-container as shown in Figure 3.

Examples of such states are:

o Global statistics (signaling, admission, preemption, flooding)

o Global counters (number of tunnels/LSPs/interfaces)

```
module: ietf-te
   +--rw te!
      +--rw globals
      |  +--rw config
      |  +--ro state
      |  |  +--ro tunnels-counter?   uint32
```

```
     |  |  +--ro lsps-counter?      uint32
     |  +--rw named-admin-groups
     |  |  +--rw config
     |  |  |  +--rw named-admin-groups* [name]
     |  |  |     +--rw name     string
     |  |  |     +--rw group?   ietf-te-types:admin-groups
     |  |  +--ro state
     |  |     +--ro named-admin-groups* [name]
     |  |        +--ro name     string
     |  |        +--ro group?   ietf-te-types:admin-groups
     |  +--rw named-srlgs
     |  |  +--rw config
     |  |  |  +--rw named-srlgs* [name]
     |  |  |     +--rw name     string
     |  |  |     +--rw group?   ietf-te-types:srlg
     |  |  +--ro state
     |  |     +--ro named-srlgs* [name]
     |  |        +--ro name     string
     |  |        +--ro group?   ietf-te-types:srlg
     |  +--rw named-explicit-paths
     |  |  +--rw config
     |  |  |  +--rw named-explicit-paths* [name]
     |  |  |     +--rw name                      string
     |  |  |     +--rw explicit-route-objects* [index]
     |  |  |        +--rw index                   uint8
     |  |  |        +--rw (type)?
     |  |  |        |  +--:(ipv4-address)
     |  |  |        |  |  +--rw v4-address?      inet:ipv4-address
     |  |  |        |  |  +--rw v4-prefix-length?uint8
     |  |  |        |  |  +--rw v4-loose?        boolean
     |  |  |        |  +--:(ipv6-address)
     |  |  |        |  |  +--rw v6-address?      inet:ipv6-address
     |  |  |        |  |  +--rw v6-prefix-length?uint8
     |  |  |        |  |  +--rw v6-loose?        boolean
     |  |  |        |  +--:(as-number)
     |  |  |        |  |  +--rw as-number?       uint16
     |  |  |        |  +--:(unnumbered-link)
     |  |  |        |  |  +--rw router-id?       inet:ip-address
     |  |  |        |  |  +--rw interface-id?    uint32
     |  |  |        |  +--:(label)
     |  |  |        |     +--rw value?           uint32
     |  |  |        +--rw explicit-route-usage?   identityref
     |  |  +--ro state
     |  |     +--ro named-explicit-paths* [name]
     |  |        +--ro name                      string
     |  |        +--ro explicit-route-objects* [index]
     |  |           +--ro index                   uint8
     |  |           +--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 explicit-route-usage?   identityref
   |  +--rw named-path-constraints
   |     +--rw config
   |     |  +--rw named-path-constraints* [name]
   |     |     +--rw name              string
   |     |     +--rw path-selection
   |     |        +--rw topology?                topology-id
   |     |        +--rw cost-limit?              uint32
   |     |        +--rw hop-limit?               uint8
   |     |        +--rw metric-type?             identityref
   |     |        +--rw tiebreaker-type?         identityref
   |     |        +--rw ignore-overload?         boolean
   |     |        +--rw tunnel-path-affinities
   |     |        |  +--rw (style)?
   |     |        |     +--:(values)
   |     |        |     |  +--rw value?         uint32
   |     |        |     |  +--rw mask?          uint32
   |     |        |     +--:(named)
   |     |        |        +--rw constraints* [usage]
   |     |        |           +--rw usage          identityref
   |     |        |           +--rw constraint
   |     |        |              +--rw affinity-names* [name]
   |     |        |                 +--rw name    string
   |     |        +--rw tunnel-path-srlgs
   |     |           +--rw (style)?
   |     |              +--:(values)
   |     |              |  +--rw usage?         identityref
   |     |              |  +--rw values*        srlg
   |     |              +--:(named)
   |     |                 +--rw constraints* [usage]
   |     |                    +--rw usage          identityref
   |     |                    +--rw constraint
   |     |                       +--rw srlg-names* [name]
```

```
         |     |                          +--rw name    string
         |     +--ro state
         |        +--ro named-path-constraints* [name]
         |           +--ro name              string
         |           +--ro path-selection
         |              +--ro topology?        topology-id
         |              +--ro cost-limit?      uint32
         |              +--ro hop-limit?       uint8
         |              +--ro metric-type?     identityref
         |              +--ro tiebreaker-type? identityref
         |              +--ro ignore-overload? boolean
         |              +--ro tunnel-path-affinities
         |              |  +--ro (style)?
         |              |     +--:(values)
         |              |     |  +--ro value?         uint32
         |              |     |  +--ro mask?          uint32
         |              |     +--:(named)
         |              |        +--ro constraints* [usage]
         |              |           +--ro usage         identityref
         |              |           +--ro constraint
         |              |              +--ro affinity-names* [name]
         |              |                 +--ro name    string
         |              +--ro tunnel-path-srlgs
         |                 +--ro (style)?
         |                    +--:(values)
         |                    |  +--ro usage?         identityref
         |                    |  +--ro values*        srlg
         |                    +--:(named)
         |                       +--ro constraints* [usage]
         |                          +--ro usage         identityref
         |                          +--ro constraint
         |                             +--ro srlg-names* [name]
         |                                +--ro name    string
```

Figure 4: TE globals configuration and state tree

3.2. Interfaces Configuration and State Data

This branch of the data model covers configurations elements that control TE features behavior system-wide. Examples of such configuration data are:

This branch of the data model covers configurations that control TE features behavior system-wide, and its respective state. Examples of such configuration data are:

This branch of the model covers configuration and state data items, the corresponding applied state data, and possible derived state pertaining to TE interfaces. Examples of tunnel configuration date for TE interfaces are:

- o Maximum reservable bandwidth, bandwidth constraints (BC)

- o Flooding parameters

    * Flooding intervals and threshold values

- o Fast reroute backup tunnel properties (such as static, auto-tunnel)

- o interface attributes

    * (Extended) administrative groups

    * SRLG values

    * TE metric value

```
module: ietf-te
   +--rw te!
      +--rw interfaces
      |  +--rw interface* [interface]
      |     +--rw interface                 if:interface-ref
      |     +--rw config
      |     |  +--rw te-metric?   ietf-te-types:te-metric
      |     +--ro state
      |     |  +--ro te-metric? ietf-te-types:te-metric
      |     |  +--ro interface-advertisements_state
      |     |     +--ro flood-interval?           uint32
      |     |     +--ro last-flooded-time?        uint32
      |     |     +--ro next-flooded-time?        uint32
      |     |     +--ro last-flooded-trigger?     enumeration
      |     |     +--ro advertized-level-areas* [level-area]
      |     |        +--ro level-area    uint32
      |     +--rw te-admin-groups
      |     |  +--rw config
      |     |  |  +--rw (admin-group-type)?
      |     |  |     +--:(value-admin-groups)
      |     |  |     |  +--rw (value-admin-group-type)?
      |     |  |     |     +--:(value-admin-groups)
      |     |  |     |     |  +--rw admin-group?
      |     |  |     |     +--:(value-extended-admin-groups)
      |     |  |     |        +--rw extended-admin-group?
      |     |  |     +--:(named-admin-groups)
      |     |  |        +--rw named-admin-groups* [named-admin-group]
      |     |  |           +--rw named-admin-group    leafref
```

```
   |     |  +--ro state
   |     |     +--ro (admin-group-type)?
   |     |        +--:(value-admin-groups)
   |     |        |  +--ro (value-admin-group-type)?
   |     |        |     +--:(value-admin-groups)
   |     |        |     |  +--ro admin-group?
   |     |        |     +--:(value-extended-admin-groups)
   |     |        |        +--ro extended-admin-group?
   |     |        +--:(named-admin-groups)
   |     |           +--ro named-admin-groups* [named-admin-group]
   |     |              +--ro named-admin-group    leafref
   |     +--rw te-srlgs
   |     |  +--rw config
   |     |  |  +--rw (srlg-type)?
   |     |  |     +--:(value-srlgs)
   |     |  |     |  +--rw values* [value]
   |     |  |     |     +--rw value    uint32
   |     |  |     +--:(named-srlgs)
   |     |  |        +--rw named-srlgs* [named-srlg]
   |     |  |           +--rw named-srlg    leafref
   |     |  +--ro state
   |     |     +--ro (srlg-type)?
   |     |        +--:(value-srlgs)
   |     |        |  +--ro values* [value]
   |     |        |     +--ro value    uint32
   |     |        +--:(named-srlgs)
   |     |           +--ro named-srlgs* [named-srlg]
   |     |              +--ro named-srlg    leafref
   |     +--rw te-switching-cap
   |     |  +--rw config
   |     |  |  +--rw switching-capabilities* [switching-capability]
   |     |  |     +--rw switching-capability    identityref
   |     |  |     +--rw encoding?               identityref
   |     |  +--ro state
   |     |     +--ro switching-capabilities* [switching-capability]
   |     |        +--ro switching-capability    identityref
   |     |        +--ro encoding?               identityref
   |     +--rw te-flooding-parameters
   |        +--rw config
   |        |  +--rw thresholds
   |        |     +--rw (type)?
   |        |        +--:(equal-steps)
   |        |        |  +--rw (equal-step-type)?
   |        |        |     +--:(up-down-different-step)
   |        |        |     |  +--rw up-step?      uint8
   |        |        |     |  +--rw down-step?    uint8
   |        |        |     +--:(up-down-same-step)
   |        |        |        +--rw step?         uint8
```

```
        |        |        +--:(unequal-steps)
        |        |           +--rw up-steps* [value]
        |        |           |  +--rw value    uint8
        |        |           +--rw down-steps* [value]
        |        |              +--rw value    uint8
        |        +--ro state
        |           +--ro thresholds
        |              +--ro (type)?
        |                 +--:(equal-steps)
        |                 |  +--ro (equal-step-type)?
        |                 |     +--:(up-down-different-step)
        |                 |     |  +--ro up-step?      uint8
        |                 |     |  +--ro down-step?    uint8
        |                 |     +--:(up-down-same-step)
        |                 |        +--ro step?         uint8
        |                 +--:(unequal-steps)
        |                    +--ro up-steps* [value]
        |                    |  +--ro value    uint8
        |                    +--ro down-steps* [value]
        |                       +--ro value    uint8
```

Figure 5: TE interfaces configuration and state tree

The state corresponding to the TE interfaces applied configuration, protocol derived state, and stats and counters all fall under the interface attributes "state" sub-container as shown in Figure 6 below:

```
module: ietf-te
   +--rw te!
      +--rw interfaces
         .
         +-- rw te-attributes
             +-- rw config
                <<intended configuration>>
             .
             +-- ro state
                <<applied configuration>>
                <<derived state associated with the TE interface>>
```

Figure 6: TE interface state

This covers state data for TE interfaces such as:

- o Bandwidth information: maximum bandwidth, available bandwidth at different priorities and for each class-type (CT)

- o List of admitted LSPs

   * Name, bandwidth value and pool, time, priority

- Statistics: state counters, flooding counters, admission counters (accepted/rejected), preemption counters

- Adjacency information

   * Neighbor address

   * Metric value

3.3. Tunnels Configuration and State Data

This branch of the model covers intended, and corresponding applied configuration for tunnels. As well, it holds possible derived state pertaining to TE tunnels.

The approach described in [I-D.openconfig-netmod-opstate] is utilised for the inclusion of operational and statistical data as shown in Figure 7.

```
module: ietf-te
   +--rw te!
      +--rw tunnels
         .
         +-- rw tunnel-properties
             +-- rw config
                &lt;&lt;intended configuration&gt;&gt;
             .
             +-- ro state
                &lt;&lt;applied configuration&gt;&gt;
                &lt;&lt;derived state associated with the tunnel&gt;&gt;
```

Figure 7: TE interface state tree

Examples of tunnel configuration date for TE tunnels:

- Name and type (e.g. P2P, P2MP) of tunnel

- Admin-state

- Primary and secondary paths

- Routing usage (auto-route announce, forwarding adjacency)

- Policy based routing (PBR) parameters

module: ietf-te

```
   +--rw te!
      +--rw tunnels
      |  +--rw tunnel* [name type]
      |     +--rw name          string
      |     +--rw type          identityref
      |     +--rw identifier?   uint16
      |     +--rw config
      |     |  +--rw description?            string
      |     |  +--rw admin-status?           identityref
      |     |  +--rw (routing-choice)?
      |     |  |  +--:(autoroute)
      |     |  |  |  +--rw autoroute-announce!
      |     |  |  |     +--rw routing-afs*       inet:ip-version
      |     |  |  |     +--rw (metric-type)?
      |     |  |  |        +--:(metric)
      |     |  |  |        |  +--rw metric?            uint32
      |     |  |  |        +--:(relative-metric)
      |     |  |  |        |  +--rw relative-metric?   int32
      |     |  |  |        +--:(absolute-metric)
      |     |  |  |           +--rw absolute-metric?   uint32
      |     |  |  +--:(forwarding-adjacency)
      |     |  |     +--rw forwarding-adjacency!
      |     |  |        +--rw holdtime?      uint32
      |     |  |        +--rw routing-afs*   inet:ip-version
      |     |  +--rw forwarding
      |     |  |  +--rw load-share?   uint32
      |     |  |  +--rw (policy-type)?
      |     |  |     +--:(class)
      |     |  |     |  +--rw class
      |     |  |     |     +--rw class?   uint8
      |     |  |     +--:(group)
      |     |  |        +--rw group
      |     |  |           +--rw classes*   uint8
      |     |  +--rw bidirectional
      |     |  |  +--rw association
      |     |  |     +--rw id?               uint16
      |     |  |     +--rw source?           inet:ip-address
      |     |  |     +--rw global-source?    inet:ip-address
      |     |  |     +--rw type?             identityref
      |     |  |     +--rw provisioing?      identityref
      |     |  +--rw (path-type)?
      |     |     +--:(p2p)
      |     |     |  +--rw destination?            inet:ip-address
      |     |     |  +--rw primary-paths* [preference]
      |     |     |     +--rw preference            uint8
      |     |     |     +--rw tunnel-path-params
      |     |     |     |  +--rw path-named-constraint?   leafref
      |     |     |     |  +--rw path-selection
```

```
      |     |     |     |  |  +--rw topology?         topology-id
      |     |     |     |  |  +--rw cost-limit?       uint32
      |     |     |     |  |  +--rw hop-limit?        uint8
      |     |     |     |  |  +--rw metric-type?      identityref
      |     |     |     |  |  +--rw tiebreaker-type?  identityref
      |     |     |     |  |  +--rw ignore-overload?  boolean
      |     |     |     |  |  +--rw tunnel-path-affinities
      |     |     |     |  |  |  +--rw (style)?
      |     |     |     |  |  |     +--:(values)
      |     |     |     |  |  |     |  +--rw value?         uint32
      |     |     |     |  |  |     |  +--rw mask?          uint32
      |     |     |     |  |  |     +--:(named)
      |     |     |     |  |  |        +--rw constraints* [usage]
      |     |     |     |  |  |          +--rw usage identityref
      |     |     |     |  |  |          +--rw constraint
      |     |     |     |  |  |            +--rw affinity-names* [name]
      |     |     |     |  |  |                +--rw name    string
      |     |     |     |  |  +--rw tunnel-path-srlgs
      |     |     |     |  |     +--rw (style)?
      |     |     |     |  |        +--:(values)
      |     |     |     |  |        |  +--rw usage?         identityref
      |     |     |     |  |        |  +--rw values*        srlg
      |     |     |     |  |        +--:(named)
      |     |     |     |  |           +--rw constraints* [usage]
      |     |     |     |  |              +--rw usage identityref
      |     |     |     |  |              +--rw constraint
      |     |     |     |  |                 +--rw srlg-names* [name]
      |     |     |     |  |                    +--rw name    string
      |     |     |     |  +--rw (type)?
      |     |     |     |  |  +--:(dynamic)
      |     |     |     |  |  |  +--rw dynamic?                 empty
      |     |     |     |  |  +--:(explicit)
      |     |     |     |  |     +--rw explicit-path-name?      leafref
      |     |     |     |  +--rw no-cspf?                 empty
      |     |     |     |  +--rw lockdown?                empty
      |     |     |     +--rw seondary-paths* [preference]
      |     |     |        +--rw preference            uint8
      |     |     |        +--rw tunnel-path-params
      |     |     |           +--rw path-named-constraint?   leafref
      |     |     |           +--rw path-selection
      |     |     |           |  +--rw topology?        topology-id
      |     |     |           |  +--rw cost-limit?      uint32
      |     |     |           |  +--rw hop-limit?       uint8
      |     |     |           |  +--rw metric-type?     identityref
      |     |     |           |  +--rw tiebreaker-type? identityref
      |     |     |           |  +--rw ignore-overload? boolean
      |     |     |           |  +--rw tunnel-path-affinities
      |     |     |           |  |  +--rw (style)?
```

```
      |     |     |           |  |     +--:(values)
      |     |     |           |  |     |  +--rw value?        uint32
      |     |     |           |  |     |  +--rw mask?         uint32
      |     |     |           |  |     +--:(named)
      |     |     |           |  |       +--rw constraints* [usage]
      |     |     |           |  |        +--rw usage  identityref
      |     |     |           |  |        +--rw constraint
      |     |     |           |  |           +--rw affinity-names*[name]
      |     |     |           |  |              +--rw name    string
      |     |     |           |  +--rw tunnel-path-srlgs
      |     |     |           |     +--rw (style)?
      |     |     |           |        +--:(values)
      |     |     |           |        |  +--rw usage?  identityref
      |     |     |           |        |  +--rw values* srlg
      |     |     |           |        +--:(named)
      |     |     |           |           +--rw constraints* [usage]
      |     |     |           |              +--rw usage identityref
      |     |     |           |              +--rw constraint
      |     |     |           |                 +--rw srlg-names* [name]
      |     |     |           |                    +--rw name    string
      |     |     |           +--rw (type)?
      |     |     |           |  +--:(dynamic)
      |     |     |           |  |  +--rw dynamic?                empty
      |     |     |           |  +--:(explicit)
      |     |     |           |     +--rw explicit-path-name? leafref
      |     |     |           +--rw no-cspf?                empty
      |     |     |           +--rw lockdown?               empty
      |     |     +--:(p2mp) {ietf-te-types:p2mp-te}?
      |     |        +--rw p2mp-paths* [destination]
      |     |           +--rw destination      inet:ip-address
      |     |           +--rw primary-paths* [preference]
      |     |              +--rw preference           uint8
      |     |              +--rw tunnel-path-params
      |     |              |  +--rw path-named-constraint?   leafref
      |     |              |  +--rw path-selection
      |     |              |  |  +--rw topology?        topology-id
      |     |              |  |  +--rw cost-limit?      uint32
      |     |              |  |  +--rw hop-limit?       uint8
      |     |              |  |  +--rw metric-type?     identityref
      |     |              |  |  +--rw tiebreaker-type? identityref
      |     |              |  |  +--rw ignore-overload? boolean
      |     |              |  |  +--rw tunnel-path-affinities
      |     |              |  |  |  +--rw (style)?
      |     |              |  |  |     +--:(values)
      |     |              |  |  |     |  +--rw value?        uint32
      |     |              |  |  |     |  +--rw mask?         uint32
      |     |              |  |  |     +--:(named)
      |     |              |  |  |        +--rw constraints* [usage]
```

```
      |     |              |  |  |         +--rw usage identityref
      |     |              |  |  |         +--rw constraint
      |     |              |  |  |          +--rw affinity-names* [name]
      |     |              |  |  |                 +--rw name    string
      |     |              |  |  +--rw tunnel-path-srlgs
      |     |              |  |     +--rw (style)?
      |     |              |  |        +--:(values)
      |     |              |  |        |  +--rw usage? identityref
      |     |              |  |        |  +--rw values*        srlg
      |     |              |  |        +--:(named)
      |     |              |  |           +--rw constraints* [usage]
      |     |              |  |              +--rw usage identityref
      |     |              |  |              +--rw constraint
      |     |              |  |                 +--rw srlg-names* [name]
      |     |              |  |                    +--rw name    string
      |     |              |  +--rw (type)?
      |     |              |  |  +--:(dynamic)
      |     |              |  |  |  +--rw dynamic?                 empty
      |     |              |  |  +--:(explicit)
      |     |              |  |     +--rw explicit-path-name? leafref
      |     |              |  +--rw no-cspf?                 empty
      |     |              |  +--rw lockdown?                empty
      |     |              +--rw seondary-paths* [preference]
      |     |                 +--rw preference           uint8
      |     |                 +--rw tunnel-path-params
      |     |                    +--rw path-named-constraint?   leafref
      |     |                    +--rw path-selection
      |     |                    |  +--rw topology?       topology-id
      |     |                    |  +--rw cost-limit?      uint32
      |     |                    |  +--rw hop-limit?       uint8
      |     |                    |  +--rw metric-type?     identityref
      |     |                    |  +--rw tiebreaker-type? identityref
      |     |                    |  +--rw ignore-overload? boolean
      |     |                    |  +--rw tunnel-path-affinities
      |     |                    |  |  +--rw (style)?
      |     |                    |  |     +--:(values)
      |     |                    |  |     |  +--rw value?         uint32
      |     |                    |  |     |  +--rw mask?          uint32
      |     |                    |  |     +--:(named)
      |     |                    |  |      +--rw constraints* [usage]
      |     |                    |  |       +--rw usage identityref
      |     |                    |  |       +--rw constraint
      |     |                    |  |        +--rw affinity-names*[name]
      |     |                    |  |            +--rw name    string
      |     |                    |  +--rw tunnel-path-srlgs
      |     |                    |     +--rw (style)?
      |     |                    |        +--:(values)
      |     |                    |        |  +--rw usage? identityref
```

```
     |     |                    |        |  +--rw values*        srlg
     |     |                    |        +--:(named)
     |     |                    |           +--rw constraints* [usage]
     |     |                    |              +--rw usage identityref
     |     |                    |              +--rw constraint
     |     |                    |                 +--rw srlg-names*
     |     |                    |                    +--rw name string
     |     |                    +--rw (type)?
     |     |                    |  +--:(dynamic)
     |     |                    |  |  +--rw dynamic? empty
     |     |                    |  +--:(explicit)
     |     |                    |     +--rw explicit-path-name? leafref
     |     |                    +--rw no-cspf?                 empty
     |     |                    +--rw lockdown?                empty
     |     +--ro state
     |        +--ro description?            string
     |        +--ro admin-status?           identityref
     |        +--ro (routing-choice)?
     |        |  +--:(autoroute)
     |        |  |  +--ro autoroute-announce!
     |        |  |     +--ro routing-afs*       inet:ip-version
     |        |  |     +--ro (metric-type)?
     |        |  |        +--:(metric)
     |        |  |        |  +--ro metric?            uint32
     |        |  |        +--:(relative-metric)
     |        |  |        |  +--ro relative-metric?   int32
     |        |  |        +--:(absolute-metric)
     |        |  |           +--ro absolute-metric?   uint32
     |        |  +--:(forwarding-adjacency)
     |        |     +--ro forwarding-adjacency!
     |        |        +--ro holdtime?      uint32
     |        |        +--ro routing-afs*   inet:ip-version
     |        +--ro forwarding
     |        |  +--ro load-share?   uint32
     |        |  +--ro (policy-type)?
     |        |     +--:(class)
     |        |     |  +--ro class
     |        |     |     +--ro class?   uint8
     |        |     +--:(group)
     |        |        +--ro group
     |        |           +--ro classes*   uint8
     |        +--ro bidirectional
     |        |  +--ro association
     |        |     +--ro id?               uint16
     |        |     +--ro source?           inet:ip-address
     |        |     +--ro global-source?    inet:ip-address
     |        |     +--ro type?             identityref
     |        |     +--ro provisioing?      identityref
```

```
      |        +--ro (path-type)?
      |        |  +--:(p2p)
      |        |  |  +--ro destination?            inet:ip-address
      |        |  |  +--ro primary-paths* [preference]
      |        |  |     +--ro preference            uint8
      |        |  |     +--ro tunnel-path-params
      |        |  |     |  +--ro path-named-constraint?   leafref
      |        |  |     |  +--ro path-selection
      |        |  |     |  |  +--ro topology?        topology-id
      |        |  |     |  |  +--ro cost-limit?      uint32
      |        |  |     |  |  +--ro hop-limit?       uint8
      |        |  |     |  |  +--ro metric-type?     identityref
      |        |  |     |  |  +--ro tiebreaker-type? identityref
      |        |  |     |  |  +--ro ignore-overload? boolean
      |        |  |     |  |  +--ro tunnel-path-affinities
      |        |  |     |  |  |  +--ro (style)?
      |        |  |     |  |  |     +--:(values)
      |        |  |     |  |  |     |  +--ro value?         uint32
      |        |  |     |  |  |     |  +--ro mask?          uint32
      |        |  |     |  |  |     +--:(named)
      |        |  |     |  |  |        +--ro constraints* [usage]
      |        |  |     |  |  |           +--ro usage  identityref
      |        |  |     |  |  |           +--ro constraint
      |        |  |     |  |  |              +--ro affinity-names*[name]
      |        |  |     |  |  |                 +--ro name    string
      |        |  |     |  |  +--ro tunnel-path-srlgs
      |        |  |     |  |     +--ro (style)?
      |        |  |     |  |        +--:(values)
      |        |  |     |  |        |  +--ro usage?         identityref
      |        |  |     |  |        |  +--ro values*        srlg
      |        |  |     |  |        +--:(named)
      |        |  |     |  |           +--ro constraints* [usage]
      |        |  |     |  |              +--ro usage identityref
      |        |  |     |  |              +--ro constraint
      |        |  |     |  |                 +--ro srlg-names* [name]
      |        |  |     |  |                    +--ro name    string
      |        |  |     |  +--ro (type)?
      |        |  |     |  |  +--:(dynamic)
      |        |  |     |  |  |  +--ro dynamic?                empty
      |        |  |     |  |  +--:(explicit)
      |        |  |     |  |     +--ro explicit-path-name?      leafref
      |        |  |     |  +--ro no-cspf?                 empty
      |        |  |     |  +--ro lockdown?                empty
      |        |  |     +--ro seondary-paths* [preference]
      |        |  |        +--ro preference            uint8
      |        |  |        +--ro tunnel-path-params
      |        |  |           +--ro path-named-constraint?   leafref
      |        |  |           +--ro path-selection
```

```
      |        |  |           |  +--ro topology?        topology-id
      |        |  |           |  +--ro cost-limit?      uint32
      |        |  |           |  +--ro hop-limit?       uint8
      |        |  |           |  +--ro metric-type?     identityref
      |        |  |           |  +--ro tiebreaker-type? identityref
      |        |  |           |  +--ro ignore-overload? boolean
      |        |  |           |  +--ro tunnel-path-affinities
      |        |  |           |  |  +--ro (style)?
      |        |  |           |  |     +--:(values)
      |        |  |           |  |     |  +--ro value?         uint32
      |        |  |           |  |     |  +--ro mask?          uint32
      |        |  |           |  |     +--:(named)
      |        |  |           |  |        +--ro constraints* [usage]
      |        |  |           |  |           +--ro usage identityref
      |        |  |           |  |           +--ro constraint
      |        |  |           |  |              +--ro affinity-names*
      |        |  |           |  |                 +--ro name    string
      |        |  |           |  +--ro tunnel-path-srlgs
      |        |  |           |     +--ro (style)?
      |        |  |           |        +--:(values)
      |        |  |           |        |  +--ro usage? identityref
      |        |  |           |        |  +--ro values*        srlg
      |        |  |           |        +--:(named)
      |        |  |           |           +--ro constraints* [usage]
      |        |  |           |              +--ro usage identityref
      |        |  |           |              +--ro constraint
      |        |  |           |                 +--ro srlg-names* [name]
      |        |  |           |                    +--ro name    string
      |        |  |           +--ro (type)?
      |        |  |           |  +--:(dynamic)
      |        |  |           |  |  +--ro dynamic?                  empty
      |        |  |           |  +--:(explicit)
      |        |  |           |     +--ro explicit-path-name? leafref
      |        |  |           +--ro no-cspf?                  empty
      |        |  |           +--ro lockdown?                 empty
      |        |  +--:(p2mp) {ietf-te-types:p2mp-te}?
      |        |     +--ro p2mp-paths* [destination]
      |        |        +--ro destination      inet:ip-address
      |        |        +--ro primary-paths* [preference]
      |        |           +--ro preference           uint8
      |        |           +--ro tunnel-path-params
      |        |           |  +--ro path-named-constraint?   leafref
      |        |           |  +--ro path-selection
      |        |           |  |  +--ro topology?        topology-id
      |        |           |  |  +--ro cost-limit?      uint32
      |        |           |  |  +--ro hop-limit?       uint8
      |        |           |  |  +--ro metric-type?     identityref
      |        |           |  |  +--ro tiebreaker-type? identityref
```

```
      |        |           |  |  +--ro ignore-overload? boolean
      |        |           |  |  +--ro tunnel-path-affinities
      |        |           |  |  |  +--ro (style)?
      |        |           |  |  |     +--:(values)
      |        |           |  |  |     |  +--ro value?         uint32
      |        |           |  |  |     |  +--ro mask?          uint32
      |        |           |  |  |     +--:(named)
      |        |           |  |  |        +--ro constraints* [usage]
      |        |           |  |  |           +--ro usage identityref
      |        |           |  |  |           +--ro constraint
      |        |           |  |  |              +--ro affinity-names*
      |        |           |  |  |                 +--ro name    string
      |        |           |  |  +--ro tunnel-path-srlgs
      |        |           |  |     +--ro (style)?
      |        |           |  |        +--:(values)
      |        |           |  |        |  +--ro usage? identityref
      |        |           |  |        |  +--ro values*        srlg
      |        |           |  |        +--:(named)
      |        |           |  |           +--ro constraints* [usage]
      |        |           |  |              +--ro usage identityref
      |        |           |  |              +--ro constraint
      |        |           |  |                 +--ro srlg-names* [name]
      |        |           |  |                    +--ro name    string
      |        |           |  +--ro (type)?
      |        |           |  |  +--:(dynamic)
      |        |           |  |  |  +--ro dynamic?                 empty
      |        |           |  |  +--:(explicit)
      |        |           |  |     +--ro explicit-path-name? leafref
      |        |           |  +--ro no-cspf?                 empty
      |        |           |  +--ro lockdown?                empty
      |        |           +--ro seondary-paths* [preference]
      |        |              +--ro preference           uint8
      |        |              +--ro tunnel-path-params
      |        |                 +--ro path-named-constraint?   leafref
      |        |                 +--ro path-selection
      |        |                 |  +--ro topology?        topology-id
      |        |                 |  +--ro cost-limit?      uint32
      |        |                 |  +--ro hop-limit?       uint8
      |        |                 |  +--ro metric-type?     identityref
      |        |                 |  +--ro tiebreaker-type? identityref
      |        |                 |  +--ro ignore-overload? boolean
      |        |                 |  +--ro tunnel-path-affinities
      |        |                 |  |  +--ro (style)?
      |        |                 |  |     +--:(values)
      |        |                 |  |     |  +--ro value?         uint32
      |        |                 |  |     |  +--ro mask?          uint32
      |        |                 |  |     +--:(named)
      |        |                 |  |        +--ro constraints* [usage]
```

```
      |        |                 |  |           +--ro usage identityref
      |        |                 |  |           +--ro constraint
      |        |                 |  |              +--ro affinity-names*
      |        |                 |  |                 +--ro name string
      |        |                 |  +--ro tunnel-path-srlgs
      |        |                 |     +--ro (style)?
      |        |                 |        +--:(values)
      |        |                 |        |  +--ro usage? identityref
      |        |                 |        |  +--ro values*        srlg
      |        |                 |        +--:(named)
      |        |                 |           +--ro constraints* [usage]
      |        |                 |              +--ro usage identityref
      |        |                 |              +--ro constraint
      |        |                 |                 +--ro srlg-names*
      |        |                 |                    +--ro name string
      |        |                 +--ro (type)?
      |        |                 |  +--:(dynamic)
      |        |                 |  |  +--ro dynamic?            empty
      |        |                 |  +--:(explicit)
      |        |                 |     +--ro explicit-path-name? leafref
      |        |                 +--ro no-cspf?                 empty
      |        |                 +--ro lockdown?                empty
      |        +--ro oper-status?            identityref
      |        +--ro lsp* [source destination tunnel-id lsp-id]
      |           +--ro source                 leafref
      |           +--ro destination            leafref
      |           +--ro tunnel-id              leafref
      |           +--ro lsp-id                 leafref
      |           +--ro extended-tunnel-id?    leafref
      |           +--ro type?                  leafref
      +--ro tunnels-state
      |  +--ro tunnel* [name type]
      |     +--ro name          string
      |     +--ro type          identityref
      |     +--ro identifier?   uint16
```

Figure 8: TE tunnels configuration and state tree

3.4. TE LSPs State Data

TE LSPs are derived state data that is usually instantiated via signaling protocols. TE LSPs exists on routers as ingress (starting point of LSP), transit (mid-point of LSP), or egress (termination point of the LSP). TE LSPs are distinguished by the 5 tuple, and LSP type (P2P or P2MP) as shown in Figure 9.

In the model, the nodes holding LSPs data exist in a read-only list as shown below:

```
         +--ro lsps-state
            +--ro lsp* [source destination tunnel-id lsp-id
                        extended-tunnel-id type]
               +--ro source                      inet:ip-address
               +--ro destination                 inet:ip-address
               +--ro tunnel-id                   uint16
               +--ro lsp-id                      uint16
               +--ro extended-tunnel-id          inet:ip-address
               +--ro type                        identityref
               +--ro oper-status?                identityref
               +--ro origin-type?                enumeration
               +--ro lsp-resource-status?        enumeration
               +--ro lsp-protection-status?      enumeration
               +--ro lsp-operational-status?     empty
               +--ro lsp-timers
               |  +--ro life-time?           uint32
               |  +--ro time-to-install?     uint32
               |  +--ro time-to-die?         uint32
               +--ro downstream-info
               |  +--ro nhop?                   inet:ip-address
               |  +--ro outgoing-interface?     if:interface-ref
               |  +--ro neighbor?               inet:ip-address
               |  +--ro label?                  uint32
               +--ro upstream-info
                  +--ro nhop?                   inet:ip-address
                  +--ro incoming-interface?     if:interface-ref
                  +--ro neighbor?               inet:ip-address
                  +--ro label?                  uint32
```

Figure 9: TE LSPs state tree

3.5. Global RPC Data

This branch of the model covers system-wide RPC execution data to trigger actions and optionally expect responses. Examples of such TE commands are to:

o Clear global TE statistics of various features

3.6. Interface RPC Data

This collection of data in the model defines TE interface RPC execution commands. Examples of these are to:

o Clear TE statistics for all or for individual TE interfaces

o Trigger immediate flooding for one or all TE interfaces

3.7. Tunnel RPC Data

This branch of the model covers TE tunnel RPC execution data to trigger actions and optionally expect responses. Examples of such TE commands are:

- Clear statistics for all or for individual tunnels

3.8. Global Notifications Data

This branch of the model covers system-wide notifications data. The node notifies the registered events to the server using the defined notification messages. Example of such global TE events are:

- Backup tunnel FRR active and not-active state transition events

3.9. Interfaces Notifications Data

This branch of the model covers TE interfaces related notifications data. The TE interface configuration is used for specific events registration. Notifications are sent for registered events to the server. Example events for TE interfaces are:

- Interface creation and deletion

- Interface state transitions

- (Soft) preemption triggers

- Fast reroute activation

3.10. Tunnel Notification Data

This branch of the model covers TE tunnels related notifications data. The TE tunnels configuration is used for specific events registration. Notifications are sent for registered events to the server. Example events for TE tunnels are:

- Tunnel creation and deletion events

- Tunnel state up/down changes

- Tunnel state reoptimization changes

4. TE Generic and Helper YANG Modules

```
<CODE BEGINS>file "ietf-te-types@2015-07-06.yang"
module ietf-te-types {

  namespace "urn:ietf:params:xml:ns:yang:ietf-te-types";

  /* Replace with IANA when assigned */
  prefix "te-types";

  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF TEAS Working Group";

  contact "Fill me";

  description
    "This module contains a collection of generally
    useful TE specific YANG data type defintions.";

  revision 2015-07-06 {
    description "Latest revision of TE basic types";
    reference "RFC3209";
  }

  identity tunnel-type {
    description
      "Base identity from which specific tunnel types are
      derived.";
  }

  identity tunnel-p2p {
    base tunnel-type;
    description
      "TE point-to-point tunnel type.";
  }

  identity tunnel-p2mp {
    base tunnel-type;
    description
      "TE point-to-multipoint tunnel type.";
  }

  identity state-type {
    description
```

```
          "Base identity for TE states";
      }

      identity state-up {
        base state-type;
        description
          "State up";
      }

      identity state-down {
        base state-type;
        description
          "State down";
      }

      identity switching-capabilities {
        description
          "Base identity for interface switching capabilities";
      }

      identity switching-psc1 {
        base switching-capabilities;
        description
          "Packet-Switch Capable-1 (PSC-1)";
      }

      identity switching-evpl {
        base switching-capabilities;
        description
          "Ethernet Virtual Private Line (EVPL)";
      }

      identity switching-l2sc {
        base switching-capabilities;
        description
          "Layer-2 Switch Capable (L2SC)";
      }

      identity switching-tdm {
        base switching-capabilities;
        description
          "Time-Division-Multiplex Capable (TDM)";
      }

      identity switching-otn {
        base switching-capabilities;
        description
          "OTN-TDM capable";
```

```
    }

    identity switching-dcsc {
      base switching-capabilities;
      description
        "Data Channel Switching Capable (DCSC)";
    }

    identity switching-lsc {
      base switching-capabilities;
      description
        "Lambda-Switch Capable (LSC)";
    }

    identity switching-fsc {
      base switching-capabilities;
      description
        "Fiber-Switch Capable (FSC)";
    }

    identity lsp-encoding-types {
      description
        "Base identity for encoding types";
    }

    identity lsp-encoding-packet {
      base lsp-encoding-types;
      description
        "Packet LSP encoding";
    }

    identity lsp-encoding-ethernet {
      base lsp-encoding-types;
      description
        "Ethernet LSP encoding";
    }

    identity lsp-encoding-pdh {
      base lsp-encoding-types;
      description
        "ANSI/ETSI LSP encoding";
    }

    identity lsp-encoding-sdh {
      base lsp-encoding-types;
      description
        "SDH ITU-T G.707 / SONET ANSI T1.105 LSP encoding";
    }
```

```
     identity lsp-encoding-digital-wrapper {
       base lsp-encoding-types;
       description
         "Digital Wrapper LSP encoding";
     }

     identity lsp-encoding-lambda {
       base lsp-encoding-types;
       description
         "Lambda (photonic) LSP encoding";
     }

     identity lsp-encoding-fiber {
       base lsp-encoding-types;
       description
         "Fiber LSP encoding";
     }

     identity lsp-encoding-fiber-channel {
       base lsp-encoding-types;
       description
         "FiberChannel LSP encoding";
     }

     identity lsp-encoding-oduk {
       base lsp-encoding-types;
       description
         "G.709 ODUk (Digital Path)LSP encoding";
     }

     identity lsp-encoding-optical-channel {
       base lsp-encoding-types;
       description
         "Line (e.g., 8B/10B) LSP encoding";
     }

     identity lsp-encoding-line {
       base lsp-encoding-types;
       description
         "Line (e.g., 8B/10B) LSP encoding";
     }

     /* TE basic features */
     feature p2mp-te {
       description
         "Indicates support for P2MP-TE";
     }
```

```
     feature frr-te {
       description
         "Indicates support for TE FastReroute (FRR)";
     }

     feature extended-admin-groups {
       description
         "Indicates support for TE link extended admin
         groups.";
     }

     feature named-path-affinities {
       description
         "Indicates support for named path affinities";
     }

     feature named-extended-admin-groups {
       description
         "Indicates support for named extended admin groups";
     }

     feature named-srlg-groups {
       description
         "Indicates support for named SRLG groups";
     }

     feature named-path-constraints {
       description
         "Indicates support for named path constraints";
     }

     grouping explicit-route-subobject {
       description
         "The explicit route subobject grouping";
       choice type {
         description
           "The explicit route subobject type";
         case ipv4-address {
           description
             "IPv4 address explicit route subobject";
           leaf v4-address {
             type inet:ipv4-address;
             description
               "An IPv4 address.  This address is
               treated as a prefix based on the
               prefix length value below. Bits beyond
               the prefix are ignored on receipt and
               SHOULD be set to zero on transmission.";
```

```
          }
          leaf v4-prefix-length {
            type uint8;
            description
              "Length in bits of the IPv4 prefix";
          }
          leaf v4-loose {
            type boolean;
            description
              "Describes whether the object is loose
              if set, or otherwise strict";
          }
        }
        case ipv6-address {
          description
            "IPv6 address Explicit Route Object";
          leaf v6-address {
            type inet:ipv6-address;
            description
              "An IPv6 address.  This address is
              treated as a prefix based on the
              prefix length value below.  Bits
              beyond the prefix are ignored on
              receipt and SHOULD be set to zero
              on transmission.";
          }
          leaf v6-prefix-length {
            type uint8;
            description
              "Length in bits of the IPv4 prefix";
          }
          leaf v6-loose {
            type boolean;
            description
              "Describes whether the object is loose
              if set, or otherwise strict";
          }
        }
        case as-number {
          leaf as-number {
            type uint16;
            description "AS number";
          }
          description
            "Autonomous System explicit route subobject";
        }
        case unnumbered-link {
          leaf router-id {
```

```
            type inet:ip-address;
            description
              "A router-id address";
          }
          leaf interface-id {
            type uint32;
            description "The interface identifier";
          }
          description
            "Unnumbered link explicit route subobject";
          reference
            "RFC3477: Signalling Unnumbered Links in
            RSVP-TE";
        }
        case label {
          leaf value {
            type uint32;
            description "the label value";
          }
          description
            "The Label ERO subobject";
        }
        /* AS domain sequence..? */
      }
    }

    grouping record-route-subobject {
      description
        "The record route subobject grouping";
      choice type {
        description
          "The record route subobject type";
        case ipv4-address {
          leaf v4-address {
            type inet:ipv4-address;
            description
              "An IPv4 address.  This address is
              treated as a prefix based on the prefix
              length value below. Bits beyond the
              prefix are ignored on receipt and
              SHOULD be set to zero on transmission.";
          }
          leaf v4-prefix-length {
            type uint8;
            description
              "Length in bits of the IPv4 prefix";
          }
          leaf v4-flags {
```

```
            type uint8;
            description
              "IPv4 address sub-object flags";
            reference "RFC3209";
          }
        }
        case ipv6-address {
          leaf v6-address {
            type inet:ipv6-address;
            description
              "An IPv6 address.  This address is
              treated as a prefix based on the
              prefix length value below. Bits
              beyond the prefix are ignored on
              receipt and SHOULD be set to zero
              on transmission.";
          }
          leaf v6-prefix-length {
            type uint8;
            description
              "Length in bits of the IPv4 prefix";
          }
          leaf v6-flags {
            type uint8;
            description
              "IPv6 address sub-object flags";
            reference "RFC3209";
          }
        }
        case label {
          leaf value {
            type uint32;
            description "the label value";
          }
          leaf flags {
            type uint8;
            description
              "Label sub-object flags";
            reference "RFC3209";
          }
          description
            "The Label ERO subobject";
        }
      }
    }

    identity route-usage-type {
      description
```

```
        "Base identity for route usage";
    }

    identity route-include-ero {
      base route-usage-type;
      description
        "Include ERO from route";
    }

    identity route-exclude-ero {
      base route-usage-type;
      description
        "Exclude ERO from route";
    }

    identity route-exclude-srlg {
      base route-usage-type;
      description
        "Exclude SRLG from route";
    }

    identity path-metric-type {
      description
        "Base identity for path metric type";
    }

    identity path-metric-te {
      base path-metric-type;
      description
        "TE path metric";
    }

    identity path-metric-igp {
      base path-metric-type;
      description
        "IGP path metric";
    }

    identity path-tiebreaker-type {
      description
        "Base identity for path tie-breaker type";
    }

    identity path-tiebreaker-minfill {
      base path-tiebreaker-type;
      description
        "Min-Fill LSP path placement";
    }
```

```
     identity path-tiebreaker-maxfill {
       base path-tiebreaker-type;
       description
         "Max-Fill LSP path placement";
     }

     identity path-tiebreaker-randoom {
       base path-tiebreaker-type;
       description
         "Random LSP path placement";
     }

     identity bidir-provisioning-mode {
       description
         "Base identity for bidirectional provisioning
         mode.";
     }

     identity bidir-provisioning-single-sided {
       base bidir-provisioning-mode;
       description
         "Single-sided bidirectional provioning mode";
     }

     identity bidir-provisioning-double-sided {
       base bidir-provisioning-mode;
       description
         "Double-sided bidirectional provioning mode";
     }

     identity bidir-association-type {
       description
         "Base identity for bidirectional association type";
     }

     identity bidir-assoc-corouted {
       base bidir-association-type;
       description
         "Co-routed bidirectional association type";
     }

     identity bidir-assoc-non-corouted {
       base bidir-association-type;
       description
         "Non co-routed bidirectional association type";
     }

     identity resource-affinities-type {
```

```
    description
      "Base identity for resource affinities";
  }

  identity resource-aff-include-all {
    base resource-affinities-type;
    description
      "The set of attribute filters associated with a
      tunnel all of which must be present for a link
      to be acceptable";
  }

  identity resource-aff-include-any {
    base resource-affinities-type;
    description
      "The set of attribute filters associated with a
      tunnel any of which must be present for a link
      to be acceptable";
  }

  identity resource-aff-exclude-any {
    base resource-affinities-type;
    description
      "The set of attribute filters associated with a
      tunnel any of which renders a link unacceptable";
  }

  typedef admin-group {
    type binary {
      length 32;
    }
    description
      "Administrative group/Resource class/Color.";
  }

  typedef extended-admin-group {
    type binary;
    description
      "Extended administrative group/Resource class/Color.";
  }

  typedef admin-groups {
    type union {
      type admin-group;
      type extended-admin-group;
    }
    description "TE administrative group derived type";
  }
```

```
     typedef srlg {
       type uint32;
       description "SRLG type";
     }

     identity path-computation-srlg-type {
       description
         "Base identity for SRLG path computation";
     }

     identity srlg-ignore {
       base path-computation-srlg-type;
       description
         "Ignores SRLGs in path computation";
     }

     identity srlg-strict {
       base path-computation-srlg-type;
       description
         "Include strict SRLG check in path computation";
     }

     identity srlg-preferred {
       base path-computation-srlg-type;
       description
         "Include preferred SRLG check in path computation";
     }

     identity srlg-weighted {
       base path-computation-srlg-type;
       description
         "Include weighted SRLG check in path computation";
     }

     typedef te-metric {
       type uint32;
       description
         "TE link metric";
     }

     typedef topology-id {
       type string {
         pattern '/?([a-zA-Z0-9\-_.]+)(/[a-zA-Z0-9\-_.]+)*';
       }
       description
         "An identifier for a topology.";
     }
```

```
     /**
      * TE tunnel generic groupings
      **/

     /* Tunnel path selection parameters */
     grouping tunnel-path-selection {
       description
         "Tunnel path selection properties grouping";
       container path-selection {
         description
           "Tunnel path selection properties container";
         leaf topology {
           type topology-id;
           description
             "The tunnel path is computed using the specific
             topology identified by this identifier";
         }
         leaf cost-limit {
           type uint32 {
             range "1..4294967295";
           }
           description
             "The tunnel path cost limit.";
         }
         leaf hop-limit {
           type uint8 {
             range "1..255";
           }
           description
             "The tunnel path hop limit.";
         }
         leaf metric-type {
           type identityref {
             base path-metric-type;
           }
           default path-metric-te;
           description
             "The tunnel path metric type.";
         }
         leaf tiebreaker-type {
           type identityref {
             base path-tiebreaker-type;
           }
           default path-tiebreaker-maxfill;
           description
             "The tunnel path computation tie breakers.";
         }
         leaf ignore-overload {
```

```
          type boolean;
          description
            "The tunnel path can traverse overloaded node.";
        }
        uses tunnel-path-affinities;
        uses tunnel-path-srlgs;
      }
    }

    grouping tunnel-path-affinities {
      description
        "Path affinities grouping";
      container tunnel-path-affinities {
        if-feature named-path-affinities;
        description
          "Path affinities container";
        choice style {
          description
            "Path affinities representation style";
          case values {
            leaf value {
              type uint32 {
                range "0..4294967295";
              }
              description
                "Affinity value";
            }
            leaf mask {
              type uint32 {
                range "0..4294967295";
              }
              description
                "Affinity mask";
            }
          }
          case named {
            list constraints {
              key "usage";
              leaf usage {
                type identityref {
                  base resource-affinities-type;
                }
                description "Affinities usage";
              }
              container constraint {
                description
                  "Container for named affinities";
                list affinity-names {
```

```
                   key "name";
                   leaf name {
                     type string;
                     description
                       "Affinity name";
                   }
                   description
                     "List of named affinities";
                 }
               }
               description
                 "List of named affinity constraints";
             }
           }
         }
       }
     }

     grouping tunnel-path-srlgs {
       description
         "Path SRLG properties grouping";
       container tunnel-path-srlgs {
         description
           "Path SRLG properties container";
         choice style {
           description
             "Type of SRLG representation";
           case values {
             leaf usage {
               type identityref {
                 base route-exclude-srlg;
               }
               description "SRLG usage";
             }
             leaf-list values {
               type srlg;
               description "SRLG value";
             }
           }
           case named {
             list constraints {
               key "usage";
               leaf usage {
                 type identityref {
                   base route-exclude-srlg;
                 }
                 description "SRLG usage";
               }
```

```
              container constraint {
                description
                  "Container for named SRLG list";
                list srlg-names {
                  key "name";
                  leaf name {
                    type string;
                    description
                      "The SRLG name";
                  }
                  description
                    "List named SRLGs";
                }
              }
              description
                "List of named SRLG constraints";
            }
          }
        }
      }
    }

    grouping tunnel-bidir-assoc-properties {
      description
        "TE tunnel associated bidirectional properties
        grouping";
      container bidirectional {
        description
          "TE tunnel associated bidirectional attributes.";
        container association {
          description
            "Tunnel bidirectional association properties";
          leaf id {
            type uint16;
            description
              "The TE tunnel association identifier.";
          }
          leaf source {
            type inet:ip-address;
            description
              "The TE tunnel association source.";
          }
          leaf global-source {
            type inet:ip-address;
            description
              "The TE tunnel association global
              source.";
          }
```

```
           leaf type {
             type identityref {
               base bidir-association-type;
             }
             default bidir-assoc-non-corouted;
             description
               "The TE tunnel association type.";
           }
           leaf provisioing {
             type identityref {
               base bidir-provisioning-mode;
             }
             description
               "Describes the provisioning model of the
               associated bidirectional LSP";
             reference
               "draft-ietf-teas-mpls-tp-rsvpte-ext-
               associated-lsp, section-3.2";
           }
         }
       }
     }
     /*** End of TE tunnel groupings ***/

     /**
      * TE interface generic groupings
      **/

     /* TE interface flooding parameters */
     grouping interface-te-flooding-parameters_config {
       description "Interface TE flooding properties.";
       container thresholds {
         description "Flooding threshold values in percentages.";
         choice type {
           description
             "Describes the flooding threshold step method";
           case equal-steps {
             choice equal-step-type {
               description
                 "Describes whether up and down equal step
                 size are same or different";
               case up-down-different-step {
                 leaf up-step {
                   type uint8 {
                     range "0..100";
                   }
                   description
                     "Set single percentage threshold
```

```
                       for increasing resource
                         allocation";
                   }
                   leaf down-step {
                     type uint8 {
                       range "0..100";
                     }
                     description
                       "Set single percentage threshold
                       for decreasing resource
                         allocation";
                   }
                 }
                 case up-down-same-step {
                   leaf step {
                     type uint8 {
                       range "0..100";
                     }
                     description
                       "Set single percentage threshold
                       for increasing and decreasing
                         resource allocation";
                   }
                 }
               }
             }
             case unequal-steps {
               list up-steps {
                 key "value";
                 description
                   "Set nultuple percentage thresholds for
                   increasing resource allocation";
                 leaf value {
                   type uint8 {
                     range "0..100";
                   }
                   description
                     "Percentage value";
                 }
               }
               list down-steps {
                 key "value";
                 description
                   "Set nultuple percentage thresholds for
                   decreasing resource allocation";
                 leaf value {
                   type uint8 {
                     range "0..100";
```

```
                }
                description
                  "Percentage value";
              }
            }
          }
        }
      }
    }

    grouping interface-te-flooding-parameters {
      description "Interface TE flooding properties.";
      container te-flooding-parameters {
        description
          "Parameters for interface flooding";
        container config {
          description
            "Configuration parameters for interface
            flooding";
          uses interface-te-flooding-parameters_config;
        }
        container state {
          config false;
          description
            "State parameters for interface flooding";
          uses interface-te-flooding-parameters_config;
        }
      }
    }
  }
  <CODE ENDS>
```

Figure 10: TE basic types YANG module

```
  <CODE BEGINS> file "ietf-te-psc-types@2015-07-06.yang"
  module ietf-te-psc-types {

    namespace "urn:ietf:params:xml:ns:yang:ietf-te-psc-types";

    /* Replace with IANA when assigned */
    prefix "te-psc-types";

    import ietf-inet-types { prefix inet; }

    organization
      "IETF TEAS Working Group";

    contact "Fill me";
```

```
     description
       "This module contains a collection of generally
       useful TE specific YANG data type defintions.";

     revision 2015-07-06 {
       description "Latest revision of TE MPLS/packet types";
       reference "RFC3209";
     }

     /* Describes egress LSP label allocation */
     typedef egress-label {
       type enumeration {
         enum "IPv4-EXPLICIT-NULL" {
           description
             "Use IPv4 explicit-NULL MPLS label at the
             egress";
         }
         enum "IPv6-EXPLICIT-NULL" {
           description
             "Use IPv6 explicit-NULL MPLS label at the
             egress";
         }
         enum "IMPLICIT-NULL" {
           description
             "Use implicit-NULL MPLS label at the egress";
         }
         enum "NON-NULL"{
           description
           "Use a non NULL MPLS label at the egress";
       }
       }
       description
         "Describes egress label allocation";
     }

     identity backup-type {
       description
         "Base identity for backup protection types";
     }

     identity backup-facility {
       base backup-type;
       description
         "Use facility backup to protect LSPs traversing
         protected TE interface";
       reference
         "RFC49090: RSVP-TE Fast Reroute";
     }
```

```
     identity backup-detour {
       base backup-type;
       description
         "Use detour or 1-for-1 protection";
       reference
         "RFC49090: RSVP-TE Fast Reroute";
     }

     identity backup-protection-type {
       description
         "Base identity for backup protection type";
     }

     identity backup-protection-link {
       base backup-protection-type;
       description
         "backup provides link protection only";
     }

     identity backup-protection-node-link {
       base backup-protection-type;
       description
         "backup offers node (preferred) or link protection";
     }

     identity bc-model-type {
       description
         "Base identity for Diffserv-TE bandwidth constraint
         model type";
     }

     identity bc-model-rdm {
       base bc-model-type;
       description
         "Russian Doll bandwidth constraint model type.";
     }

     identity bc-model-mam {
       base bc-model-type;
       description
         "Maximum Allocation bandwidth constraint
         model type.";
     }

     identity bc-model-mar {
       base bc-model-type;
       description
         "Maximum Allocation with Reservation
```

```
         bandwidth constraint model type.";
     }

     grouping bandwidth-constraint-values {
       description
         "Packet bandwidth contraints values";
       choice value-type {
         description
           "Value representation";
         case percentages {
           container perc-values {
             uses bandwidth-psc-constraints;
             description
               "Percentage values";
           }
         }
         case absolutes {
           container abs-values {
             uses bandwidth-psc-constraints;
             description
               "Absolute values";
           }
         }
       }
     }

     grouping bandwidth-psc-reservable {
       description
         "Packet reservable bandwidth";
       choice bandwidth-value {
         description "Reservable bandwidth configuraiton choice";
         case absolute {
           leaf absolute-value {
             type uint32;
             description "Absolute value of the bandwidth";
           }
         }
         case precentage {
           leaf percent-value {
             type uint32 {
               range "0..4294967295";
             }
             description "Percentage reservable bandwidth";
           }
           description
             "The maximum reservable bandwidth on the
             interface";
         }
```

```
      }
      choice bc-model-type {
        description
          "Reservable bandwidth percentage capacity
          values.";
        case bc-model-rdm {
          container bc-model-rdm {
            description
              "Russian Doll Model Bandwidth Constraints.";
            uses bandwidth-psc-constraints;
          }
        }
        case bc-model-mam {
          container bc-model-mam {
            uses bandwidth-psc-constraints;
            description
              "Maximum Allocation Model Bandwidth
              Constraints.";
          }
        }
        case bc-model-mar {
          container bc-model-mar {
            uses bandwidth-psc-constraints;
            description
              "Maximum Allocation with Reservation Model
              Bandwidth Constraints.";
          }
        }
      }
    }

    typedef bfd-type {
      type enumeration {
        enum classical {
          description "BFD classical session type.";
        }
        enum seamless {
          description "BFD seamless session type.";
        }
      }
      default "classical";
      description
        "Type of BFD session";
    }

    typedef bfd-encap-mode-type {
      type enumeration {
        enum gal {
```

```
          description
            "BFD with GAL mode";
        }
        enum ip {
          description
            "BFD with IP mode";
        }
      }
      default ip;
      description
        "Possible BFD transport modes when running over TE
        LSPs.";
    }

    grouping bandwidth-psc-constraints {
      description "Bandwidth constraints.";
      container bandwidth-psc-constraints {
        description
          "Holds the bandwidth contraints properties";
        leaf maximum-reservable {
          type uint32 {
            range "0..4294967295";
          }
          description
            "The maximum reservable bandwidth on the
            interface";
        }
        leaf-list bc-value {
          type uint32 {
            range "0..4294967295";
          }
          max-elements 8;
          description
            "The bandwidth contraint type";
        }
      }
    }

    grouping tunnel-forwarding-properties {
      description "Properties for using tunnel in forwarding.";
      container forwarding {
        description
          "Tunnel forwarding properties container";
        leaf load-share {
          type uint32 {
            range "1..4294967295";
          }
          description "ECMP tunnel forwarding
```

```
            load-share factor.";
        }
        choice policy-type {
          description
            "Tunnel policy type";
          container class {
            description
              "Tunnel forwarding per class properties";
            leaf class {
              type uint8 {
                range "1..7";
              }
              description
                "The class associated with this tunnel";
            }
          }
          container group {
            description
              "Tunnel frowarding per group properties";
            leaf-list classes {
              type uint8 {
                range "1..7";
              }
              description
                "The forwarding class";
            }
          }
        }
      }
    }

    grouping tunnel-routing-properties {
      description
        "TE tunnel routing properties";
      choice routing-choice {
        description
          "Announces the tunnel to IGP as either
          autoroute or forwarding adjacency.";
        case autoroute {
          container autoroute-announce {
            presence "Enable autoroute announce.";
            description
              "Announce the TE tunnel as autoroute to
              IGP for use as IGP shortcut.";
            leaf-list routing-afs {
              type inet:ip-version;
              description
                "Address families";
```

```
             }
             choice metric-type {
               description
                 "Type of metric to use when announcing
                 the tunnel as shortcut";
               leaf metric {
                 type uint32 {
                   range "1..2147483647";
                 }
                 description
                   "Describes the metric to use when
                   announcing the tunnel as shortcut";
               }
               leaf relative-metric {
                 type int32 {
                   range "-10..10";
                 }
                 description
                   "Relative TE metric to use when
                   announcing the tunnel as shortcut";
               }
               leaf absolute-metric {
                 type uint32 {
                   range "1..2147483647";
                 }
                 description
                   "Absolute TE metric to use when
                   announcing the tunnel as shortcut";
               }
             }
           }
         }
         case forwarding-adjacency {
           container forwarding-adjacency {
             presence "Enable forwarding adjacency
               on the tunnel.";
             description
               "Announce the TE tunnel
               as forwarding adjacency.";
             leaf holdtime {
               type uint32 {
                 range "0..4294967295";
               }
               description
                 "Holdtime in seconds after
                 tunnel becomes UP.";
             }
             leaf-list routing-afs {
```

```
                type inet:ip-version;
                description
                  "Address families";
              }
            }
          }
        }
      }
    }
    <CODE ENDS>
```

Figure 11: TE packet/MPLS specific types YANG module

```
    <CODE BEGINS> file "ietf-te@2015-07-06.yang"
    module ietf-te {

      namespace "urn:ietf:params:xml:ns:yang:ietf-te";

      /* Replace with IANA when assigned */
      prefix "te";

      /* Import TE generic types */
      import ietf-te-types {
        prefix ietf-te-types;
      }

      /* Import TE packet specific types */
      import ietf-te-psc-types {
        prefix ietf-te-psc-types;
      }

      import ietf-interfaces {
        prefix if;
      }

      import ietf-inet-types {
        prefix inet;
      }

      organization
        "IETF TEAS Working Group";

      contact
        "Fill me";

      description
        "YANG data module for TE configuration,
        state, RPC and notifications.";
```

```
     revision 2015-07-06 {
       description "Latest update to TE generic YANG module.";
       reference "TBD";
     }

     /**
      * TE interface generic groupings
      */
     grouping te-admin-groups_config {
       description
         "TE interface affinities grouping";
       choice admin-group-type {
         description
           "TE interface administrative groups
           representation type";
         case value-admin-groups {
           choice value-admin-group-type {
             description "choice of admin-groups";
             case value-admin-groups {
               description
                 "Administrative group/Resource
                 class/Color.";
               leaf admin-group {
                 type ietf-te-types:admin-group;
                 description
                   "TE interface administrative group";
               }
             }
             case value-extended-admin-groups {
               if-feature ietf-te-types:extended-admin-groups;
               description
                 "Extended administrative group/Resource
                 class/Color.";
               leaf extended-admin-group {
                 type ietf-te-types:extended-admin-group;
                 description
                   "TE interface extended administrativei
                   group";
               }
             }
           }
         }
         case named-admin-groups {
           list named-admin-groups {
             if-feature ietf-te-types:extended-admin-groups;
             if-feature ietf-te-types:named-extended-admin-groups;
             key named-admin-group;
             description
```

```
               "A list of named admin-group entries";
             leaf named-admin-group {
               type leafref {
                 path "/te/globals/" +
                   "named-admin-groups/config/"+
                   "named-admin-groups/name";
               }
               description
                 "A named admin-group entry";
             }
           }
         }
       }
     }

     grouping te-admin-groups {
       description "TE admin-group configuration grouping";
       container te-admin-groups {
         description
           "Configuration parameters for interface
           administrative groups";
         container config {
           description
             "Configuration parameters for interface
             administrative groups";
           uses te-admin-groups_config;
         }
         container state {
           config false;
           description
             "Configuration parameters for interface
             administrative groups";
           uses te-admin-groups_config;
         }
       }
     }

     /* TE interface SRLGs */
     grouping te-srlgs_config {
       description "TE interface SRLG grouping";
       choice srlg-type {
         description "Choice of SRLG configuration";
         case value-srlgs {
           list values {
             key "value";
             description "List of SRLG values that
               this link is part of.";
             leaf value {
```

```
              type uint32 {
                range "0..4294967295";
              }
              description
                "Value of the SRLG";
            }
          }
        }
        case named-srlgs {
          list named-srlgs {
            if-feature ietf-te-types:named-srlg-groups;
            key named-srlg;
            description
              "A list of named SRLG entries";
            leaf named-srlg {
              type leafref {
                path "/te:te/te:globals/" +
                  "te:named-srlgs/te:config/te:named-srlgs/te:name";
              }
              description
                "A named SRLG entry";
            }
          }
        }
      }
    }

    grouping te-srlgs {
      description "TE SRLG configuration grouping";
      container te-srlgs {
        description "TBD";
        container config {
          description
            "Configuration parameters for interface SRLG
            groups";
          uses te-srlgs_config;
        }
        container state {
          config false;
          description
            "State parameters for interface SRLG groups";
          uses te-srlgs_config;
        }
      }
    }

    grouping te-metric_config {
      description "Interface TE metric grouping";
```

```
    leaf te-metric {
      type ietf-te-types:te-metric;
      description "Interface TE metric.";
    }
  }

  grouping te-attributes {
    description "TE attributes configuration grouping";
    container config {
      description
        "Configuration parameters for interface TE
         attributes";
      uses te-metric_config;
    }
    container state {
      config false;
      description
        "State parameters for interface TE metric";
      uses te-metric_config;
      uses interface-advertisements_state;
    }
  }

  /* TE interface switching capabilities */
  grouping te-switching-cap_config {
    description
      "TE interface switching capabilities";
    list switching-capabilities {
      key "switching-capability";
      description
        "List of interface capabilities for this interface";
      leaf switching-capability {
        type identityref {
          base ietf-te-types:switching-capabilities;
        }
        description
          "Switching Capability for this interface";
      }
      leaf encoding {
        type identityref {
          base ietf-te-types:lsp-encoding-types;
        }
        description
          "Encoding supported by this interface";
      }
    }
  }
```

```
     grouping te-switching-cap {
       description "TE interface switching capability grouping";
       container te-switching-cap {
         description
           "Interface switching capabilities container";
         container config {
           description
             "Configuration parameters for interface
             switching capabilities";
           uses te-switching-cap_config;
         }
         container state {
           config false;
           description
             "State parameters for interface switching
             capabilities";
           uses te-switching-cap_config;
         }
       }
     }

     grouping interface-advertisements_state {
       description
         "TE interface advertisements state grouping";
       container interface-advertisements_state {
         description
           "TE interface advertisements state container";
         leaf flood-interval {
           type uint32;
           description
             "The periodic flooding interval";
         }
         leaf last-flooded-time {
           type uint32;
           units seconds;
           description
             "Time elapsed since last flooding in seconds";
         }
         leaf next-flooded-time {
           type uint32;
           units seconds;
           description
             "Time remained for next flooding in seconds";
         }
         leaf last-flooded-trigger {
           type enumeration {
             enum link-up {
               description "Link-up flooding trigger";
```

```
            }
            enum link-down {
              description "Link-up flooding trigger";
            }
            enum threshold-up {
              description
                "Bandwidth reservation up threshold";
            }
            enum threshold-down {
              description
                "Bandwidth reservation down threshold";
            }
            enum bandwidth-change {
              description "Banwidth capacity change";
            }
            enum user-initiated {
              description "Initiated by user";
            }
            enum srlg-change {
              description "SRLG property change";
            }
            enum periodic-timer {
              description "Periodic timer expired";
            }
          }
          description "Trigger for the last flood";
        }
        list advertized-level-areas {
          key level-area;
          description
            "List of areas the TE interface is advertised
            in";
          leaf level-area {
            type uint32;
            description
              "The IGP area or level where the TE
              interface state is advertised in";
          }
        }
      }
    }
    /*** End of TE interface groupings ***/

    /**
     * TE tunnel generic groupings
     */

    /* TE tunnel path properties */
```

```
     grouping tunnel-path-params {
       description
         "Tunnel path properties grouping";
       container tunnel-path-params {
         description
           "Defines a TE tunnel path properties";
         leaf path-named-constraint {
           if-feature ietf-te-types:named-path-constraints;
           type leafref {
             path "/te/globals/named-path-constraints/config/"+
               "named-path-constraints/name";
           }
           description
             "Reference to a globally defined named path
             constraint set";
         }
         uses ietf-te-types:tunnel-path-selection;
         choice type {
           description
             "Describes the path type";
           case dynamic {
             leaf dynamic {
               type empty;
               description
                 "A CSPF dynamically computed path";
             }
           }
           case explicit {
             leaf explicit-path-name {
               type leafref {
                 path "/te/globals/named-explicit-paths/config/"+
                   "named-explicit-paths/name";
               }
               description
                 "Reference to a globally defined
                 explicit-path";
             }
           }
         }
         leaf no-cspf {
           type empty;
           description
             "Indicates no CSPF is to be attempted on this
             path.";
         }
         leaf lockdown {
           type empty;
           description
```

```
             "Indicates no reoptimization to be attempted for
             this path.";
         }
       }
     }

     grouping tunnel-properties_config {
       description
         "Configuration parameters relating to TE tunnel";
       leaf description {
         type string;
         description
           "TE tunnel description.";
       }
       leaf admin-status {
         type identityref {
           base ietf-te-types:state-type;
         }
         default ietf-te-types:state-up;
         description "TE tunnel administrative state.";
       }
       uses ietf-te-psc-types:tunnel-routing-properties;
       uses ietf-te-psc-types:tunnel-forwarding-properties;
       uses ietf-te-types:tunnel-bidir-assoc-properties;
       choice path-type {
         description
           "Describes the path type";
         case p2p {
           leaf destination {
             type inet:ip-address;
             description
               "P2P tunnel destination address";
           }
           /* P2P list of path(s) */
           list primary-paths {
             key "preference";
             description
               "List of primary paths for this
               tunnel.";
             leaf preference {
               type uint8 {
                 range "1..255";
               }
               description
                 "Specifies a preference for
                 this path. The lower the
                 number higher the
                 preference";
```

```
            }
            uses tunnel-path-params;
            list seondary-paths {
              key "preference";
              description
                "List of secondary paths for this
                tunnel.";
              leaf preference {
                type uint8 {
                  range "1..255";
                }
                description
                  "Specifies a preference for
                  this path. The lower the
                  number higher the
                  preference";
              }
              uses tunnel-path-params;
            }
          }
        }
        case p2mp {
          if-feature ietf-te-types:p2mp-te;
          list p2mp-paths {
            key "destination";
            description
              "List of destinations and their
              paths.";
            leaf destination {
              type inet:ip-address;
              description
                "P2MP destination leaf address";
            }
            list primary-paths {
              key "preference";
              description
                "List of primary paths";
              leaf preference {
                type uint8 {
                  range "1..255";
                }
                description
                  "Specifies a preference for
                  this path. The lower the
                  number higher the
                  preference";
              }
              uses tunnel-path-params;
```

```
          list seondary-paths {
            key "preference";
            description
              "List of secondary paths";
            leaf preference {
              type uint8 {
                range "1..255";
              }
              description
                "Specifies a preference
                for this path. The lower
                  the number higher
                    the preference";
            }
            uses tunnel-path-params;
          }
        }
      }
    }
  }
}

grouping tunnel-properties {
  description
    "Top level grouping for tunnel properties.";
  container config {
    description
      "Configuration parameters relating to
      tunnel properties";
    uses tunnel-properties_config;
  }
  container state {
    config false;
    description
      "State information associated with tunnel
      properties";
    uses tunnel-properties_config;
    uses tunnel-properties_state;
  }
}

grouping tunnel-properties_state {
  description
    "State parameters relating to TE tunnel";
  leaf oper-status {
    type identityref {
      base ietf-te-types:state-type;
    }
```

```
          description "TE tunnel operational state.";
        }
        list lsp {
          key "source destination tunnel-id lsp-id";
          description "List of LSPs associated with the tunnel.";

          leaf source {
            type leafref {
              path "/te/lsps-state/lsp/source";
            }
            description
              "Tunnel sender address extracted from
              SENDER_TEMPLATE  object";
            reference "RFC3209";
          }
          leaf destination {
            type leafref {
              path "/te/lsps-state/lsp/destination";
            }
            description
              "Tunnel endpoint address extracted from
              SESSION object";
            reference "RFC3209";
          }
          leaf tunnel-id {
            type leafref {
              path "/te/lsps-state/lsp/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/lsps-state/lsp/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/lsps-state/lsp/extended-tunnel-id";
```

```
           }
           description
             "Extended Tunnel ID of the LSP.";
           reference "RFC3209";
         }
         leaf type {
           type leafref {
             path "/te/lsps-state/lsp/type";
           }
           description "LSP type P2P or P2MP";
         }
       }
     }
     /*** End of TE tunnel groupings ***/

     /**
      * LSP related generic groupings
      */

     grouping lsp-properties_state {
       description
         "State parameters relating to LSP";
       leaf oper-status {
         type identityref {
           base ietf-te-types:state-type;
         }
         description "LSP operational state.";
       }

       leaf origin-type {
         type enumeration {
           enum ingress {
             description
               "Origin ingress";
           }
           enum egress {
             description
               "Origin egress";
           }
           enum transit {
             description
               "transit";
           }
         }
         description
           "Origin type of LSP relative to the location
           of the local switch in the path.";
       }
```

```
        leaf lsp-resource-status {
          type enumeration {
            enum primary {
              description
                "A primary LSP is a fully established LSP for
                 which the resource allocation has been committed
                 at the data plane";
            }
            enum secondary {
              description
                "A secondary LSP is an LSP that has been provisioned
                 in the control plane only; e.g. resource allocation
                 has not been committed at the data plane";
            }
          }
          description "LSP resource allocation type";
          reference "rfc4872, section 4.2.1";
        }

        leaf lsp-protection-status {
          type enumeration {
            enum working {
              description
                "A working LSP must be a primary LSP whilst a protecting
                 LSP can be either a primary or a secondary LSP. Also,
                 known as protected LSPs when working LSPs are associated
                 with protecting LSPs.";
            }
            enum protecting {
              description
                "A secondary LSP is an LSP that has been provisioned
                 in the control plane only; e.g. resource allocation
                 has not been committed at the data plane";
            }
          }
          description "LSP role type";
          reference "rfc4872, section 4.2.1";
        }

        leaf lsp-operational-status {
          type empty;
          description
            "This bit is set when a protecting LSP is carrying the normal
             traffic after protection switching";
        }

        container lsp-timers {
          when "../origin-type = 'ingress'" {
```

```
            description "Applicable to ingress LSPs only";
          }
          description "Ingress LSP timers";
          leaf life-time {
            type uint32;
            units seconds;
            description
              "lsp life time";
          }

          leaf time-to-install {
            type uint32;
            units seconds;
            description
              "lsp installation delay time";
          }

          leaf time-to-die {
            type uint32;
            units seconds;
            description
              "lsp expire delay time";
          }
        }

        container downstream-info {
          description
            "downstream information";

          leaf nhop {
            type inet:ip-address;
            description
              "downstream nexthop.";
          }

          leaf outgoing-interface {
            type if:interface-ref;
            description
              "downstream interface.";
          }

          leaf neighbor {
            type inet:ip-address;
            description
              "downstream neighbor.";
          }

          leaf label {
```

```
          type uint32;
          description
            "downstream label.";
        }
      }

      container upstream-info {
        description
          "upstream information";

        leaf nhop { // phop?
          type inet:ip-address;
          description
            "upstream nexthop.";
        }

        leaf incoming-interface {
          type if:interface-ref;
          description
            "upstream interface.";
        }

        leaf neighbor {
          type inet:ip-address;
          description
            "upstream neighbor.";
        }

        leaf label {
          type uint32;
          description
            "upstream label.";
        }
      }
    }
    /*** End of TE LSP groupings ***/

    /**
     * TE global generic groupings
     */

    grouping global-attributes_config {
      description
        "Top level grouping for global config data.";
    }

    grouping global-attributes_state {
      description
```

```
          "Top level grouping for global state data.";
        leaf tunnels-counter {
          type uint32;
          description "Tunnels count";
        }
        leaf lsps-counter {
          type uint32;
          description "Tunnels count";
        }
      }

      grouping global-attributes {
        description "TE Global attributes grouping";
        container config {
          description
            "Global configuration parameters";
          uses global-attributes_config;
        }
        container state {
          config false;
          description
            "Global configuration parameters";
          uses global-attributes_config;
          uses global-attributes_state;
        }
      }

      grouping named-admin-groups_config {
        description
          "Global named administrative groups configuration
          grouping";
        list named-admin-groups {
          if-feature ietf-te-types:extended-admin-groups;
          if-feature ietf-te-types:named-extended-admin-groups;
          key "name";
          description
            "List of named TE admin-groups";
          leaf name {
            type string;
            description
              "A string name that uniquely identifies a TE
              interface named admin-group";
          }
          leaf group {
            type ietf-te-types:admin-groups;
            description
              "An SRLG value";
          }
```

```
    }
  }

  grouping named-admin-groups {
    description
      "Named admin groups grouping";
    container named-admin-groups {
      description
        "Named admin groups container";
      container config {
        description
          "Configuration parameters for named admin
          groups";
        uses named-admin-groups_config;
      }
      container state {
        config false;
        description
          "State parameters for named admin groups";
        uses named-admin-groups_config;
      }
    }
  }

  grouping named-srlgs_config {
    description
      "Global named SRLGs configuration
      grouping";
    list named-srlgs {
      if-feature ietf-te-types:named-srlg-groups;
      key "name";
      description
        "A list of named SRLG groups";
      leaf name {
        type string;
        description
          "A string name that uniquely identifies a TE
          interface named srlg";
      }
      leaf group {
        type ietf-te-types:srlg;
        description "An SRLG value";
      }
    }
  }

  grouping named-srlgs {
    description
```

```
          "Global named SRLGs grouping";
        container named-srlgs {
          description
            "Named SRLGs container";
          container config {
            description
              "Configuration parameters for named SRLG groups";
            uses named-srlgs_config;
          }
          container state {
            config false;
            description
              "State parameters for named SRLG groups";
            uses named-srlgs_config;
          }
        }
      }

      grouping named-explicit-paths_config {
        description
          "Global explicit path configuration
          grouping";
        list named-explicit-paths {
          key "name";
          description
            "A list of explicit paths";
          leaf name {
            type string;
            description
              "A string name that uniquely identifies an
              explicit path";
          }
          list explicit-route-objects {
            key "index";
            description
              "List of explicit route objects";
            leaf index {
              type uint8 {
                range "0..255";
              }
              description
                "Index of this explicit route object";
            }
            uses ietf-te-types:explicit-route-subobject;
            leaf explicit-route-usage {
              type identityref {
                base ietf-te-types:route-usage-type;
              }
```

```
            description
              "An IP hop action.";
          }
        }
      }
    }

    grouping named-explicit-paths {
      description
        "Global named explicit path grouping";
      container named-explicit-paths {
        description
          "Nmaed explicit paths container";
        container config {
          description
            "Configuration parameters for named explicit
            paths";
          uses named-explicit-paths_config;
        }
        container state {
          config false;
          description
            "State parameters for named explicit paths";
          uses named-explicit-paths_config;
        }
      }
    }

    grouping named-path-constraints_config {
      description
        "Global named path constraints configuration
        grouping";
      list named-path-constraints {
        if-feature ietf-te-types:named-path-constraints;
        key "name";
        description
          "A list of named path constraints";
        leaf name {
          type string;
          description
            "A string name that uniquely identifies a
            path constraint set";
        }
        uses ietf-te-types:tunnel-path-selection;
      }
    }

    grouping named-path-constraints {
```

```
      description
        "Global named path constraints grouping";
      container named-path-constraints {
        description
          "Nmaed explicit paths container";
        container config {
          description
            "Configuration parameters for named explicit
            paths";
          uses named-path-constraints_config;
        }
        container state {
          config false;
          description
            "State parameters for named explicit paths";
          uses named-path-constraints_config;
        }
      }
    }
    /*** End of TE global groupings ***/

    /**
     * TE configurations container
     */
    container te {
      presence "Enable TE feature.";
      description
        "TE global container.";

      /* TE Global Configuration Data */
      container globals {
        description
          "Configuration data for Global System-wide
          Traffic Engineering.";
        uses global-attributes;
        uses named-admin-groups;
        uses named-srlgs;
        uses named-explicit-paths;
        uses named-path-constraints;
      }

      /* TE Interface Configuration Data */
      container interfaces {
        description
          "Configuration data model for TE interfaces.";
        list interface {
          key "interface";
          description "TE interfaces.";
```

```
          leaf interface {
            type if:interface-ref;
            description
              "TE interface name.";
          }
          /* TE interface parameters */
          uses te-attributes;
          uses te-admin-groups;
          uses te-srlgs;
          uses te-switching-cap;
          /* TE interface flooding parameters */
          uses ietf-te-types:interface-te-flooding-parameters;
        }
      }

      /* TE Tunnel Configuration Data */
      container tunnels {
        description
          "Configuration, operational, notification and RPC
          data model for TE tunnels.";

        list tunnel {
          key "name type";
          unique "identifier";
          description "TE tunnel.";
          leaf name {
            type string;
            description "TE tunnel name.";
          }
          leaf type {
            type identityref {
              base ietf-te-types:tunnel-type;
            }
            description "TE tunnel type.";
          }
          leaf identifier {
            type uint16;
            description
              "TE tunnel Identifier.";
          }
          uses tunnel-properties;
        }
      }

      /* TE Tunnel Ephemeral State Data (TBD) */
      container tunnels-state {
        config "false";
        description
```

```
            "Derived state corresponding to ephemeral tunnels";

          list tunnel {
            key "name type";
            unique "identifier";
            description "TE tunnel.";
            leaf name {
              type string;
              description "TE tunnel name.";
            }
            leaf type {
              type identityref {
                base ietf-te-types:tunnel-type;
              }
              description "TE tunnel type.";
            }
            leaf identifier {
              type uint16;
              description
                "TE tunnel Identifier.";
            }

          }
        }

        /* TE LSPs State Data */
        container lsps-state {
          config "false";
          description "MPLS-TE LSP operational state data.";

          list lsp {
            key
              "source destination tunnel-id lsp-id "+
              "extended-tunnel-id type";
            description
              "List of LSPs associated with the tunnel.";
            leaf source {
              type inet:ip-address;
              description
                "Tunnel sender address extracted from
                SENDER_TEMPLATE  object";
              reference "RFC3209";
            }
            leaf destination {
              type inet:ip-address;
              description
                "Tunnel endpoint address extracted from
                SESSION object";
```

```
            reference "RFC3209";
          }
          leaf tunnel-id {
            type uint16;
            description
              "Tunnel identifier used in the SESSION
              that remains constant over the life
              of the tunnel.";
            reference "RFC3209";
          }
          leaf lsp-id {
            type uint16;
            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 inet:ip-address;
            description
              "Extended Tunnel ID of the LSP.";
            reference "RFC3209";
          }
          leaf type {
            type identityref {
              base ietf-te-types:tunnel-type;
            }
            description "The LSP type P2P or P2MP";
          }
          uses lsp-properties_state;
        }
      }
    }

    /* TE Global RPCs/execution Data */
    rpc globals-rpc {
      description
        "Execution data for TE global.";
    }

    /* TE interfaces RPCs/execution Data */
    rpc interfaces-rpc {
      description
        "Execution data for TE interfaces.";
    }
```

```
     /* TE Tunnel RPCs/execution Data */
     rpc tunnels-rpc {
       description
         "TE tunnels RPC nodes";
     }


     /* TE Global Notification Data */
     notification globals-notif {
       description
         "Notification messages for Global TE.";
     }

     /* TE Interfaces Notification Data */
     notification interfaces-notif {
       description
         "Notification messages for TE interfaces.";
     }

     /* TE Tunnel Notification Data */
     notification tunnels-notif {
       description
         "Notification messages for TE tunnels.";
     }
   }
   <CODE ENDS>
```

Figure 12: TE generic YANG module

5. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-te XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-te-types XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-te-psc-types 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-te namespace: urn:ietf:params:xml:ns:yang:ietf-te prefix: ietf-te reference: RFC3209

name: ietf-te-types namespace: urn:ietf:params:xml:ns:yang:ietf-te-types prefix: ietf-te-types reference: RFC3209

name: ietf-te-psc-types namespace: urn:ietf:params:xml:ns:yang:ietf-te-psc-types prefix: ietf-te-psc-types reference: RFC3209

6. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF

users to a pre-configured subset of all available NETCONF protocol operations and content.

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

"/te/globals": This module specifies the global TE configurations on a device. Unauthorized access to this container could cause the device to ignore packets it should receive and process.

"/te/tunnels": This list specifies the configured TE tunnels on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

"/te/interfaces": This list specifies the configured TE interfaces on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

7. Acknowledgement

The authors would like to thank Lou Berger for reviewing and providing valuable feedback on this document.

8. References

8.1. 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-19 (work in progress), May 2015.

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

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

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

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

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

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

8.2. Informative References

[I-D.openconfig-mpls-consolidated-model]
George, J., Fang, L., eric.osborne@level3.com, e., and R. Shakir, "MPLS / TE Model for Service Provider Networks", draft-openconfig-mpls-consolidated-model-00 (work in progress), March 2015.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-00 (work in progress), March 2015.

[RFC3473] Berger, L., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, January 2003.

[RFC4328] Papadimitriou, D., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Extensions for G.709 Optical Transport Networks Control", RFC 4328, January 2006.

Authors’ Addresses

Tarek Saad (editor)
Cisco Systems Inc

Email: tsaad@cisco.com

Rakesh Gandhi
Cisco Systems Inc

Email: rgandhi@cisco.com

Xufeng Liu
Ericsson

Email: xufeng.liu@ericsson.com

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Himanshu Shah
Ciena

Email: hshah@ciena.com

Xia Chen
Huawei Technologies

Email: jescia.chenxia@huawei.com

Raqib Jones
Brocade

Email: raqib@Brocade.com

Bin Wen
Comcast

Email: Bin_Wen@cable.comcast.com

MPLS Working Group X. Zhang
Internet-Draft D. Dhody
Intended status: Standards Track Huawei Technologies
Expires: September 4, 2015 March 3, 2015

A generic YANG Data Model for Label Switch Path (LSP).
draft-zhang-mpls-lspdb-yang-00

Abstract

This document defines a generic YANG data model for the Label Switch Paths (LSPs). The data model includes the operational state of LSP (LSP-DB). It is expected that this modules will be augmented by additional YANG modules defining data models for signalling protocol and other functions.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Multiprotocol Label Switching (MPLS) and Generalized MPLS (GMPLS) Label Switched Path (LSP) has various common attributes irrespective of the protocol and signalling mechanism (For example, RSVP-TE, LDP, BGP, PCEP etc).

The objective of this document is to write a high level generic LSP database module to capture these common attributes. Thus this document defines YANG [RFC6020] data model for a generic high-level tree structures of MPLS and GMPLS LSPs.

Further modules augmenting this data model with specific signalling protocol, technology or advanced features will be handled in a separate document A possible relationship is roughly depicted in the figure below.

```
                 +---------------+
                 | Generic LSPDB |
                 |   ietf-lspdb  |
                 +-------^-------+
                         |
         ---------------------------------------------
        |                |                |           |
        |                |                |           |
   +----+----+   +-------+-------+   +----+----+   +--+--+
   | LDP-LSP |   |    BGP-LSP    |   |  TE-LSP |   | SR  |
   |         |   |               |   |         |   |     |
   +---------+   +---------------+   +----^----+   +--^--+
                                          |           |
                                           -----------
                                          |           |
                                          |           |
                                     +----+----+   +--+--+
                                     | RSVP-TE |   | SR- |
                                     |         |   | TE  |
                                     +----^----+   +--^--+
                                          |           |
                                          |           |
                                           -----------
                                          |
                                     +----+----+
                                     |  PCEP   |
                                     |         |
                                     +---------+
```

This document contains a specification of the base LSPDB YANG module, "ietf-lspdb".

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. Tree Diagrams

A graphical representation of the complete data tree is presented in Section 6. 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 Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.

- 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 and "*" denotes a "list" or "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.

4. 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]
if	ietf-interfaces	[RFC7223]

Table 1: Prefixes and corresponding YANG modules

5. 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 MIB Module should be clear.
- o The data model should allow for static configurations of LSP. (Editors Note - Maybe taken for future)

o The data model should include read-only counters in order to gather statistics with errors.

o It should be fairly straightforward to augment the base data model.

6. The Design of LSP-DB Data Model

The module "ietf-lspdb", defines the generic components of a LSP-DB. This module will be augmented with with additional data nodes and features based on signalling protocol and advanced features.

```
module: ietf-lspdb
   +--ro lspdb
      +--ro lsp-num?     uint32
      +--ro lsp-entry* [system-generated-id]
         +--ro system-generated-id    uint64
         +--ro lsp-signaling          lsp-signalingtypes
         +--ro is-primary?            boolean
         +--ro lsr-type?              lsr-types
         +--ro source                 inet:ip-address
         +--ro destination            inet:ip-address
         +--ro creation-time?         yang:date-and-time
         +--ro last-change?           yang:date-and-time
         +--ro operation-status?      status-types
         +--ro incoming
         |  +--ro incoming-interface?   if:interface-state-ref
         |  +--ro incoming-label
         +--ro outgoing
         |  +--ro outgoing-interface?   if:interface-state-ref
         |  +--ro outgoing-label
         +--ro primary-lsp*           lsp-ref
         +--ro backup-lsp*            lsp-ref
         +--ro statistics
```

6.1. The Lsp-entry Lists

The LSP-DB yang module contains the full LSP-DB and thus the status information for all LSPs. The data model presented in this document uses a flat list of LSPs. Each entity in the list is identified by a system generated id (system-generated-id). Furthermore, each entity has a mandatory "lsp-signaling" leaf (the signalling protocol/ mechanism).

6.2. Incoming/Outgoing Containers

These containers contain the interface and label information in incoming and outgoing directions. A reference to the interface name to "ietf-interfaces" module [RFC7223] and an empty label container is maintained. The label should be augmented by the data plane specific YANG modules.

6.3. Primary/Backup Lists

All LSPs (primary and backup) are maintained together in a single flat lsp-entry list. The relationship to other LSPs is maintained via the "primary-lsp" and "backup-lsp" leaf-lists.

A LSP is identified by its system-generated-id, which is unique within the node. This property is captured in "lsp-ref" typedef, which other should use when they need to reference a LSP.

Thus a list of references to primary and backup LSPs are maintained per LSP.

7. The LSP-DB 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-lspdb@2015-03-xx.yang"




module ietf-lspdb {

    namespace "urn:ietf:params:xml:ns:yang:ietf-lspdb";
    prefix lspdb;

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

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

```
       import ietf-interfaces {
           prefix "if";
       }

       organization
           "IETF XXX Working Group";

       contact
           "
           Editor:   Dhruv dhody
                     <dhruv.dhody@huawei.com>
           Editor:   Xian ZHANG
                     <zhang.xian@huawei.com>";
       description
           "This module contains a collection of YANG definitions for
           configuring LSP database. The intent of this module is to
           serve as a base model and it is kept protocol-independent.
           It is expected that it will be augmented depending on the
           targeted protocol.

           Copyright (c) 2015 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 2015-03-05 {
           description
               "Initial revision.";
           reference
               "RFC XXX: A YANG Data Model for Label Switched Path
                           (LSP) Database Management";
       }

       /*
        * Features:none.
        */


       /*
        * Typedefs
        */

       typedef lsp-signalingtypes {
```

```
            type enumeration {
                enum "rsvp" {
                    value 1;
                    description
                    "Resource reSerVation Protocol.";
                }
                enum "ldp" {
                    value 2;
                    description
                    "Label Distribution Protocol";
                }
                enum "bgp" {
                    value 3;
                    description
                    "Border Gateway Protocol.";
                }
                enum "sr" {
                    value 4;
                    description
                    "Segment Routing.";
                }
                enum "static" {
                    value 5;
                    description
                    "Manually Configured.";
                }
            }
            description
                "The signalling type of a LSP.";
        }

        typedef status-types {
            type enumeration {
                enum "up" {
                    value 1;
                    description
                    "UP";
                }
                enum "down" {
                    value 2;
                    description
                    "Not working/failed";
                }
                enum "standby" {
                    value 3;
                    description
                    "Idle state, created by not used";
                }
```

```
            }
            description
                "The status types of a LSP.";
        }

        typedef lsr-types {
            type enumeration {
                enum "ingress" {
                    value 1;
                    description
                    "Ingress node";
                }
                enum "transit" {
                    value 2;
                    description
                    "Transit node";
                }
                enum "egress" {
                    value 3;
                    description
                    "Egress node";
                }
            }
            description
                "the role of the Label Switched Router (LSR) for a
                LSP entry.";
        }

        typedef lsp-ref {
            type leafref {
                path "/lspdb/lsp-entry/system-generated-id";
            }
            description
                "This type is used by data models that need to
                reference other LSP.";
        }

        /*
         * Operational data nodes
         */

        container lspdb {

            config false;

            description
                "This container defines the information of all the
                LSP a node has/stores";
```

```
          leaf lsp-num {
              type uint32;

              description
                  "This stores the total number of LSPs, including
                  working and backup.";
          }

          list lsp-entry {

              key "system-generated-id";

              description
                  "This define each LSP entry.";

              leaf system-generated-id {
                  type uint64;
                  description
                      "This is generated by the local node and it is
                      unique within this node";
              }

              leaf lsp-signaling {
                  type lsp-signalingtypes;
                  mandatory true;
                  description
                      "The signalling protocol/mechanism for the
                                          LSP.";
              }

              leaf is-primary {
                  type boolean;
                  description
                      "Whether a LSP is a primary or second LSP.
                       1-primary, 0-secondary";
              }

              leaf lsr-type {
                  type lsr-types;
                  description
                      "The role of this LSR with regard to the
                       current LSP";
              }

              leaf source {
                  type inet:ip-address;
                  mandatory true;
                  description
```

```
                "The Source node of this LSP";
        }

        leaf destination {
            type inet:ip-address;
            mandatory true;
            description
                "The Destination node of this LSP";
        }

        leaf creation-time {
            type yang:date-and-time;
            description
                "The time the LSP was created.";
        }

        leaf last-change {
            type yang:date-and-time;
            description
                "The time the LSP entered its current state.";
        }

        leaf operation-status {
            type status-types;
            description
                "The operational status of this LSP";
        }



        container incoming {
            description
                "The incoming interface, label
                                    information.";
            leaf incoming-interface {
                type if:interface-state-ref;
                description
                    "The reference to the name of the incoming
                     interface.";
            }
            container incoming-label {
                description
                    "Empty container, Label format to be
                    augmented depending on the  data plane
                    technology";
            }
        }
```

```
            container outgoing {
                description
                    "The outgoing interface, label
                                         information.";
                leaf outgoing-interface {
                    type if:interface-state-ref;
                    description
                        "The reference to the name of the outgoing
                         interface.";
                }

                container outgoing-label {
                    description
                        "Empty container, Label format to be
                        augmented depending on the data plane
                        technology";
                }
            }

            leaf-list primary-lsp {
                type lsp-ref;
                description
                    "A list of references to primary LSPs (if
                    exist) for this LSP.";
                reference
                    "xxx";
            }

            leaf-list backup-lsp {
                type lsp-ref;
                description
                    "A list of references to backup LSPs (if
                    exist) for this LSP.";
                reference
                    "xxx";
            }

            container statistics {
                description
                    "TBD";
            }
        }
    }
}//module

<CODE ENDS>
```

8. 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.

9. Manageability Considerations

9.1. Control of Function and Policy

9.2. Information and Data Models

9.3. Liveness Detection and Monitoring

9.4. Verify Correct Operations

9.5. Requirements On Other Protocols

9.6. Impact On Network Operations

10. 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-lspdb

Registrant Contact: The MPLS 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-lspdb
Namespace: urn:ietf:params:xml:ns:yang:ietf-lspdb
Prefix: lspdb
Reference: This I-D

11. Acknowledgements

12. References

12.1. Normative References

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

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

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

12.2. Informative References

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

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

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

[I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-01 (work in progress), October 2014.

Appendix A. Contributor Addresses

Authors' Addresses

Xian Zhang
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R.China

EMail: zhang.xian@huawei.com

Dhruv Dhody
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560037
India

EMail: dhruv.ietf@gmail.com