Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

I. Bryskin
Huawei Technologies
X. Liu
Jabil
V. Beeram
Juniper Networks
T. Saad
Cisco Systems Inc

March 13, 2017

ONF/T-API Services vs. IETF/YANG Models and Interfaces
draft-bryskin-teas-yang-ietf-vs-onf-00

Abstract

This document compares IETF YANG TE (Traffic Engineering) data model and ONF/T-API model.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 14, 2017.

Table of Contents

1. Introduction

The success of T-SDN as an architecture depends to a large degree on the quality and widespread adoption of open standardized interfaces to/from T-SDN controllers, linking them flexibly into various hierarchies and confederations. Currently, the two most popular such interfaces are:

1. T-API developed by ONF;

2. RESTCONF/YANG [RFC7950] based on TE Topology and TE Tunnels models defined in [I-D.ietf-teas-yang-te-topo] and [I-D.ietf-teas-yang-te] documents respectively, the product of IETF TEAS WG.

The two interfaces have the close attention of network operators and vendors. There is a lot of confusion about their respective technical merits and "marketing" strengths, applications they can support, use cases they cover, and so on. Do they compete or could they somehow complement each other?

This memo is limited to a strictly technical comparison with the special focus on the models supporting the two interfaces, in particular, the semantics, relationships, informational flows and services they define. Our analysis suggests that the IETF models provide for implementation of powerful hierarchical T-SDN controller systems, supporting a broad range of client systems and use cases, and that in some identifiable respects, T-API appears to fall relatively shorter. This memo is largely organized around considering the identified "gaps".

2. Topology Service

2.1. Constrained Nodes

The T-API Topology service does not support the notion of blocking/ constrained nodes. This means that if a T-API Topology service provider exposes to a client a topology with at least one node with constrained connectivity, e.g. the node can switch a potential TE path/connection, say, from interface (NodeEdge point) A to B, but not from A to C; there is no way for the provider to communicate the connection limitations to the client, thus making the provided TE topology unfit for the client’s path computations. This is a serious issue because many transport physical switches and virtually all abstract composite nodes should be treated as blocking nodes.

Likewise, if a potential path source/destination node is constrained in such a way that the path may leave/enter the source/destination node over a link from a subset of (but not all) same-layer links connected to the node, the T-API Topology service provider has no way of communicating such a circumstance to the client.

The described issue is addressed in the IETF TE Topology model. A TE node’s Connectivity Matrix attribute (Figure 1) fully describes the node’s TE path/connection switching limitations, while a TE Tunnel Termination Point’s (TTP’s) Local Link Connectivity List attribute (Figure 2) describes the node’s TE path/connection termination limitations with respect to each TTP hosted by the node in question.

Basic Connectivity Matrix:	Detailed Connectivity Matrix:
LTP-6/label-x <=> LTP-1/label-y	LTP-6/label-x <=> LTP-1/label-y (Cost c, Delay d, SRLB s, ...)
LTP-5/label-x <=> LTP-2/label-y	LTP-5/label-x <=> LTP-2/label-y (Cost c, Delay d, SRLB s, ...)
LTP-5/label-x <=> LTP-4/label-y	LTP-5/label-x <=> LTP-4/label-y (Cost c, Delay d, SRLB s, ...)
LTP-4/label-x <=> LTP-1/label-y	LTP-4/label-x <=> LTP-1/label-y (Cost c, Delay d, SRLB s, ...)
LTP-3/label-x <=> LTP-2/label-y	LTP-3/label-x <=> LTP-2/label-y
...	...

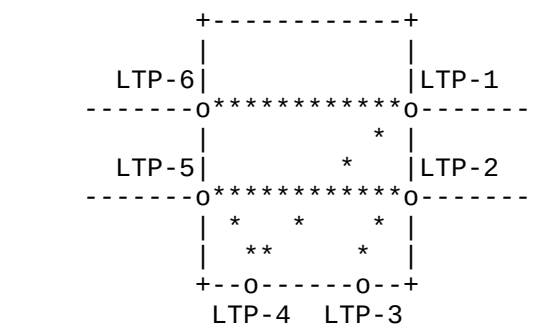


Figure 1: TE Node Connectivity Matrix

```
TTP-1 Basic LLCL:

TTP-1 <=> {LTP-5/label-x,
           LTP-2/label-y}
```

```
TTP-1 Detailed LLCL:

TTP-1 <=> {
 LTP-5/label-x,
  (Cost c, Delay d, SRLB s, ...),
 LTP-2/label-y,
  (Cost c, Delay d, SRLB s, ...)
 }
```

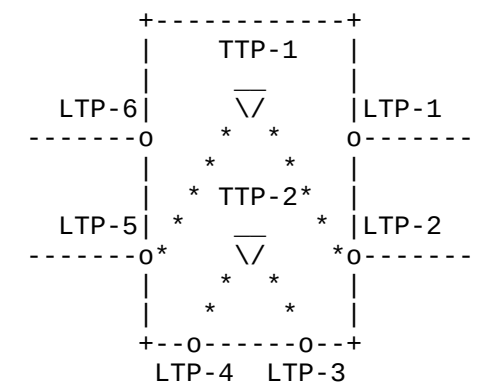


Figure 2: TTP Local Link Connectivity List

2.2. Intra-node Metrics

There is no good way for a T-API Topology service provider to articulate to the client what it would cost for a potential path (e.g., in terms of delay) to cross a node from interface (NodeEdge point) A to interface B. Because nodes (especially composite abstract nodes) may contribute to overall path costs much more than links connecting the nodes along the path, this fact makes the provided topology unfit for the client's path selection optimizations. [Note: To be fair, the T-API Topology service does allow a composite abstract node (representing a group of inter-connected nodes) to refer to the topology describing the abstract node's internals (node's encapTopology attribute). Hence the client may in theory apply path computation algorithms on the abstract node's internal/encapsulated topology to figure out whether the abstract node can switch a path between a given pair of the abstract node's NodeEdge points, as well as the cost penalties the path will accrue by doing so. However, such a technique defeats the whole purpose of creating the abstract node in the first place, which is hiding multiple topological elements behind the abstract node, so that the top level topology becomes smaller and easier to use in path computations. In other words, if the client has to "dive" into the abstract node's internal topology every time the client needs to

understand whether and how a path can cross the abstract node, the client would be better off if the abstract node were not provided, and instead, the node's internals were presented directly in the top level topology.]

This issue does not exist in the IETF TE Topology model. A TE node's Detailed Connectivity Matrix attribute (Figure 1, upper right) associates with each (abstract or physical) node's connectivity matrix entry a vector of costs (in terms of generic TE cost, delay, intra-node SRLGs, etc.) that a potential TE path will have to add to its end-to-end costs should the path select the entry to cross the node. Likewise, a TE path's source/destination TTP's Detailed Local Link Connectivity List attribute (Figure 2, upper right) indicates what it would cost for the path to start/stop on a given first/last link. [Note: In the IETF TE topology model an abstract TE node also points to the encapsulated TE topology describing the node's internals. However, the client is expected to peruse the node's encapsulated TE topology only in exceptional situations (e.g. during trouble shooting), rather than under normal conditions, such as routine path computations.]

2.3. Topology Updates

Suppose that a T-API Topology service client has requested and received a topology from one of its providers (for example, the topology presented in Figure 3). It is imperative that as soon as this done the provider starts updating the client (continuously and in unsolicited way) with changes happening to the topological elements and their attributes that the client has expressed interest in - otherwise, the client would be forced to make decisions on stale information.

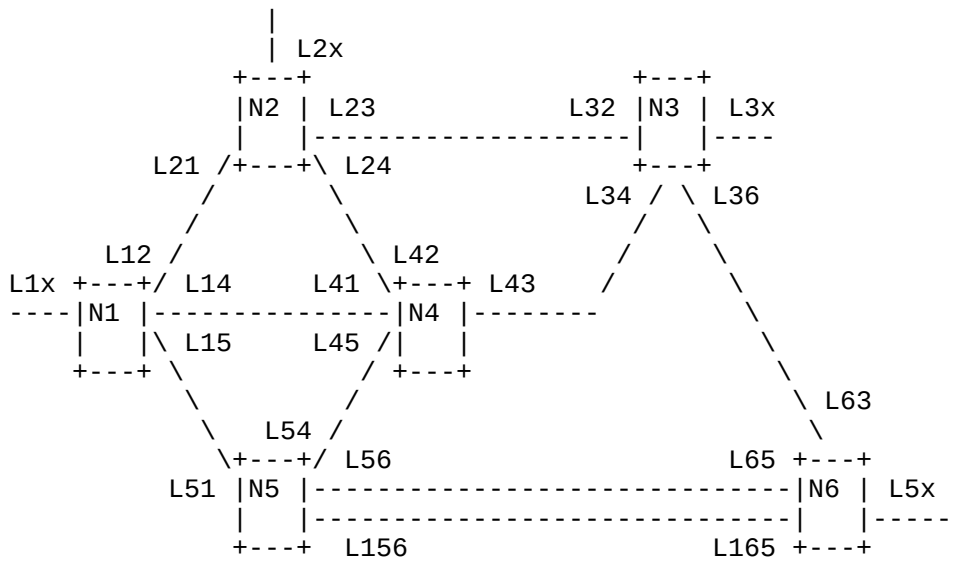


Figure 3: Topology presented to T-API Topology service client

The only way this could be done in T-API is via using T-API Notification service, specifically, the Attribute Value Change (AVC) Notification service, which in a nutshell works as follows:

- Provider registers with the service the types of pre-defined AVC events it is willing and capable of providing notifications for, along with the set of pre-defined object types that may comprise the notification contents;

- Client discovers the registered notifications it can subscribe to and subscribes to some of them, specifying filters to tailor the notifications to its needs.

There are two problems with this paradigm:

1. The client has a very limited way to express which notifications it is interested in, as well as the contents, triggers and frequency of such notifications. Note that even for the same topology element type (e.g., link) different clients may need to know different things, at different scopes and granularities, with respect to the attribute changes. For example, one client may want to hear about links that experienced changes in any attribute, while another client may be interested only in links with changes in specific attribute(s). One client may want to learn about link attribute modifications across all provided topologies, while another client may want to know only about such

links that belong to one or more specific (but not other) topologies. One client may want to receive in the notification the entire set of link attributes, while another client would want to learn only about incremental changes (i.e., changes that happened since the previous notification); some clients are interested not in just any attribute change, but rather, want to know when the attribute has reached a specific threshold, etc. As mentioned, a T-API client has only the option to discover what the provider is willing to offer (without the provider really knowing what their clients want to learn) and to subscribe to a subset of that;

2. In order for the client to understand/interpret the notifications registered by the provider, all notification event types, as well as the types of objects comprising the notification content, must be explicitly pre-defined. Considering the sheer number of, say, link attributes (especially, combinations of them) that different clients may be interested in, and the possible scopes, granularities and triggers of the notifications; explicit pre-definition of notifications is awkward, limited and impractical (if not infeasible).

In sharp contrast, the IETF TE topology model requires no explicit definition of notifications. When the client subscribes to a TE topology update notification it:

a. defines the notification event type by specifying the YANG XPath from the TE topology data store root to the data store node(s) associated with link attribute(s) encompassing the client’s points of interest;

b. specifies another XPath pointing to the data store’s sub-tree, node or group of nodes to identify the content of the notification and whether the entire new state or incremental changes must be provided;

c. defines the trigger for the notification, which could be any change in the node(s) of interest or a specific increment in value or the value hitting a specific threshold;

d. optionally defines the highest notification frequency at which the client wants to receive the notifications.

To illustrate this assume that the IETF TE Topology model client wants to be notified about all TE links whose available capacity has dropped below 10G, with the notification carrying the actual link’s available capacity. In this case the client will:

a. specify root->all TE topologies -> all TE links->linkAttributes->bandwidth XPath as the notification type;

b. specify the same XPath to define the desired notification content;

c. define the notification trigger by specifying the low and high thresholds (e.g. 10G and 15 G respectively);

d. optionally specify the highest frequency of updates the client is capable/willing to consume.

Note that no explicit definitions for the notification were required. After the client registers with the provider the defined subscription, the latter knows exactly what the former wants to be notified about and how. Similar notifications are possible to register with the provider with respect to any TE topology element attribute or combination of thereof.

2.4. Topology Telemetry Collection

Topology service clients (which in the T-SDN context could be various controllers or applications, such as multi-domain coordinators, IP/transport integrators, orchestrators, big data collectors, analytics processors, network planners, etc.) are hungry for accurate real time network state information (a.k.a. network telemetry). This knowledge is instrumental for a client in keeping the network under its control healthy, stable and optimized under conditions of fiber cuts, hardware and software failures. In particular, network telemetry streams provided by the client’s providers allow for the client to identify/predict failing network resources and route the provided transport/connectivity services away from them; to identify/predict points of congestion and eliminate/mitigate the congestion by deploying extra network capacity in a timely manner and so forth. Network telemetry is a valuable source of information useful for network planning, trouble shooting and many other things. Network telemetry is especially important for topology service clients because topologies represent - in an abstracted way - the physical network resources.

[Note: At the time of writing of this memo there were no known TAPI design/modeling activities related to telemetry streaming for any of the T-API services].

Topology telemetry collection is similar in nature to receiving updates on topology attribute changes. Per the description in section 1.3, T-API Notification service, State Change (SC) Notification service is the only mechanism theoretically (i.e. after

all the necessary modeling concepts and attributes, such as statistics counters, are in place) available for the client to subscribe and for the provider to stream the requested network telemetry. T-API SC Notification service has the same drawbacks as the AVC Notification service, specifically:

a. limited capability for the client to articulate what telemetry (event type, content, granularity, etc.) it seeks to receive;

b. necessity for explicit definition of the telemetry events and notification messages.

These issues do not exist in the network telemetry streaming machinery offered by the IETF Topology model. Let’s consider, for example, that the client wants to identify "flipping" TE links (i.e. TE links frequently changing their UP/DOWN operational status) and obtain in the notification the entire state information for such TE links. In order to achieve this the client needs to:

a. specify root->all TE topologies -> all TE links->linkStatistics->linkUPCounter XPath as the notification type;

b. specify root->all TE topologies -> all TE links->linkState XPath to describe the desired notification content;

c. define the notification trigger by specifying the number the model data state node of interest (the linkUPCounter) must increment by for the next notification to be issued;

d. optionally specify the highest frequency of notifications of this type the client is capable/willing to consume.

2.5. Topology Name/Address Spaces

T-API topologies are required to have each node and link assigned a globally unique UUID. This means that all T-API Topology service clients and providers have to resolve potential UUID collisions via allocating the UUIDs from a universal name space governed by a centralized authority (in a similar way to how global IP addresses are assigned in IP networks).

The IETF TE Topology model allows for all TE topologies to have independent name spaces for the TE node, link and SRLG IDs, which not only eliminates the problem of ID collisions, but also greatly simplifies the design and implementation of network applications such as L0/L1 VPNs.

In Figure 4 a TE topology provider exposes its native (i.e. real, physical) TE topology as separate abstract TE topologies to two clients, each one customized separately on per client basis. According to the IETF TE Topology model each of the three depicted TE topologies may have an independent name space for their respective TE node, link and SRLG IDs.

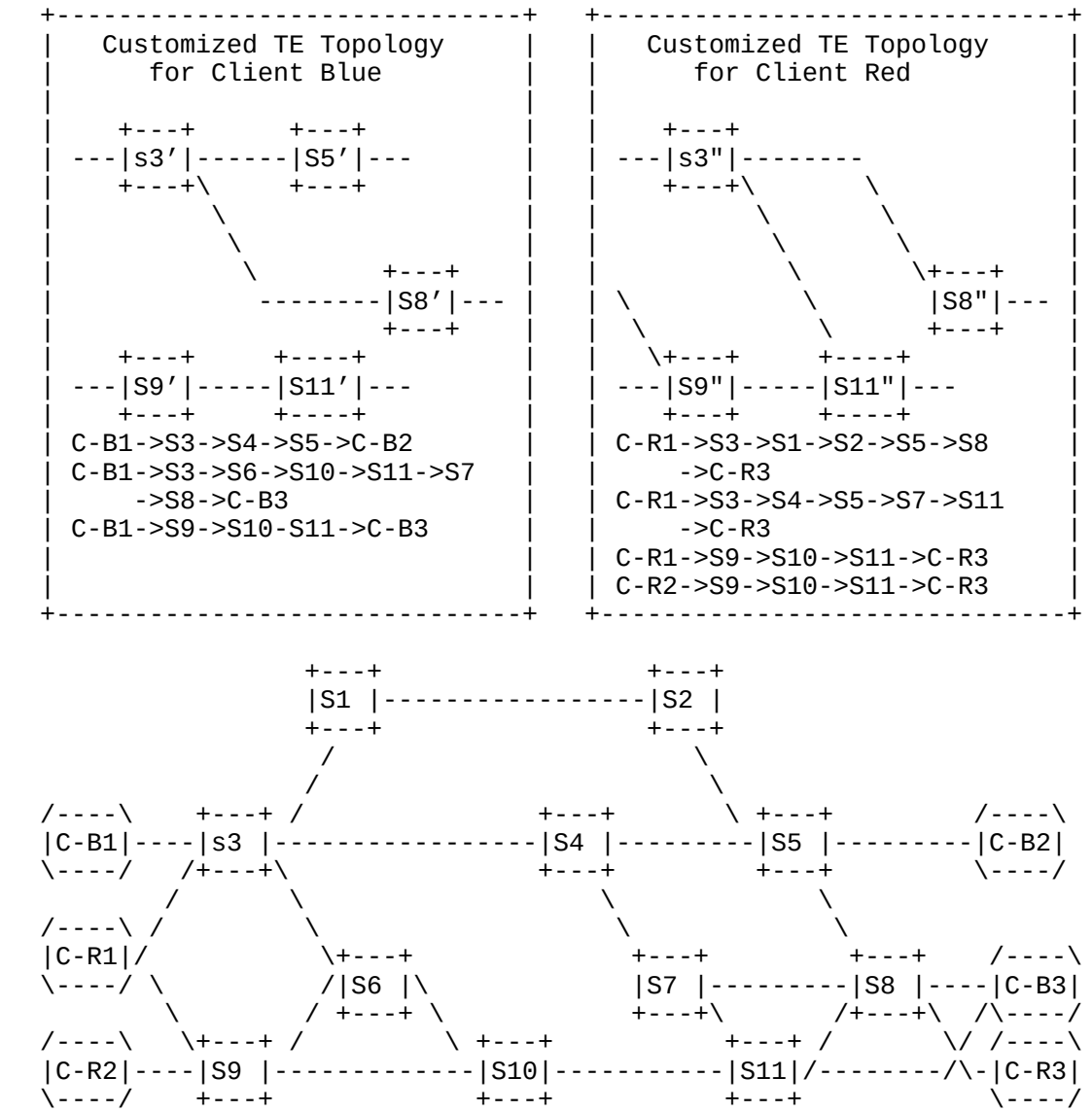


Figure 4: Abstract TE topologies customized for different clients

2.6. Topology Relationships

An IETF TE Topology model provider may expose to the same client multiple TE topologies, which:

- could be native (as known to the provider, unmodified) or abstract (generated by the provider as overlays based on native or lower level abstract TE topologies);
- could describe different layer networks in accordance with distinct layer-specific model augmentations;
- abstract TE topologies could be of a different type (e.g. single node, link mesh, etc.) and of a different hierarchy level;
- abstract TE topologies could be optimized based on different optimization criteria (e.g. smallest cost, shortest delay, best link protection, etc.)

The provider can convey to the client the TE topology optimization criteria, as well as the provider's preference as to the order in which the provided TE topologies are to be used via topology scope attributes specifically designed for this purpose. Furthermore, the TE Topology model defines various inter-topology relationships designed to describe abstract TE topology hierarchies, client-server layer network (vertical) relationships and domain neighboring (horizontal) relationships. The defined inter-topology relationships are as follows:

- TE node underlay topology: A composite abstract TE node of a higher hierarchy level TE topology X, representing a group of inter-connected TE nodes that belong to a lower hierarchy level TE topology Y, has an attribute pointing to Y (i.e., ID of the abstract TE node's internal/encapsulated TE topology);
- TE link underlay topology: A TE link of a TE topology X can point to TE Topology Y which was used by the provider to compute primary and backup TE paths that are (or are to be) used by the actual or potential TE tunnel (transport connectivity) supporting the TE link in question. The TE paths themselves could be provided in the same TE link attribute;
- Supporting node/link topology: A given TE node or link may show up in multiple TE topologies catered by the provider to the client. In order for the provider not to provide/update (and for the client not to consume) multiple identical sets of attributes, the model allows for providing/updating only for one (original) TE node/link, and having the "twins" point to the original TE mode/

link, as well as to the TE topology where the original TE node/ link could be found;

- Source node/link topology: A given TE node or link catered by the provider as a part of a TE topology to the client may be provided to the provider by one of its own providers. In such case the TE node/link in question can point to the original TE node/link, as well as to the TE topology where the original is defined, thus allowing for multi-level multi-provider TE topology hierarchies (see Figure 5);

- Inter-layer lock: This is the relationship/attribute that associates TE links of a higher layer network TE topology with TE Tunnel Termination Points (TTPs) of one or more lower layer network TE topology(ies) to articulate to the client inter-topology /inter-layer adaptation capabilities, to lock the TE topologies describing separate layer networks vertically, thus allowing for client multi-layer path computations and other multi-layer TE applications;

- Inter-domain plug: This is a relationship modeled via an inter-domain TE link attribute that allows for a client managing interconnected multi-domain networks (with each domain served by a separate provider) to identify neighboring domains and to lock the TE topologies provided by all providers horizontally, thus producing TE topologies homogeneously describing the entire multi-domain network and allowing for end-to-end path computations across the network.

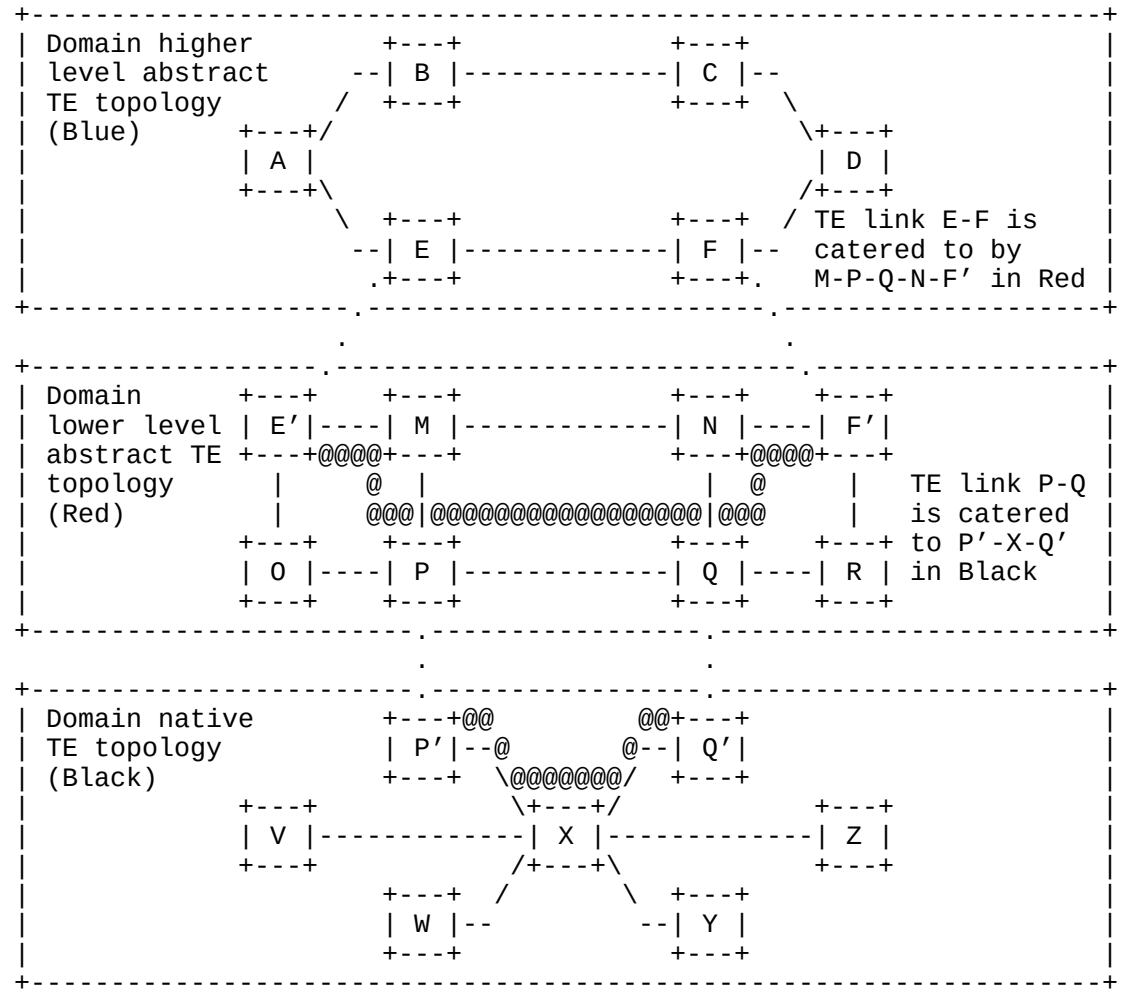


Figure 5: Hierarchical multi-provider abstract TE topologies

A T-API Topology service provider is also allowed to expose multiple topologies to the client. The only inter-topology relationship defined is the Node's encapTopology (which is effectively the same as the IETF's TE node underlay topology relationship described above). Otherwise, all the provided topologies are independent. It is not clear for the client what is the purpose of each of them, what is the provider's preference as to how and in which order they are supposed to be used, and why several same layer topologies, rather than one, were provided to the client in the first place.

2.7. Topology Attributes

Compared to the IETF TE Topology model, T-API Topology nodes and links are missing some important attributes. Specifically, T-API nodes, as mentioned in section 1.1, have no analogs to the Connectivity matrix attribute and the TE TTP container describing nodes switching and termination capabilities/limitations respectively. Furthermore, the T-API Topology service does not have a concept of TTP, which in the context of the IETF TE Topology model conveys to the client various important edge characteristics for a TE tunnel that could be provided by the network described by a given TE topology. Such characteristics include:

- Potential TE tunnel protection capabilities (e.g., whether 1+1 protection could or could not be supported for the tunnel edge);
- Adaptation capacities (i.e., which higher layer network payload types and from which higher layer link termination points can be adopted on the TE tunnel edge, the amount of adaptation bandwidth still available, etc.);
- Technology-specific TTPs describe technology specific properties (e.g. TTP representing an OCh layer transponder can announce whether the transponder’s receiver/transmitter is fixed or tunable, and in the latter case what is the range and resolution of the tunability; supported FECs and signal modulation modes, transmit/acceptable optical signal power levels and OSNRs, etc.)

The T-API Topology link is missing the following attributes:

- Administrative groups (administrative colors) - an attribute describing the link’s association with pre-defined groups of links; such groups could be used as constraints in the client’s path selection/optimization algorithms to mandate/disallow or encourage/discourage the resulting paths to follow/avoid links related to the specified groups;
- Link protection/restoration capability - an attribute that could be also used as a path computation constraint or path optimization criterion, for example, to force or encourage the resulting paths to follow sufficiently protected links;
- Link properties defining whether the link is:
  - A. actual (with committed network resources) or potential;
  - B. static (with pre-established and always-in-place server layer connectivity supporting the link) or dynamic (for which the

connectivity is dynamically put in place if/when the link is used by at least one client connection and is dynamically released as soon as the link is used by none of the client's connections);

- o Link's underlay primary and backup paths and ID of the topology used for their computations.

2.8. Topology Service Relationships with Other Services

IETF TE topology and TE tunnel models are related. For example, a TE link can point via the Supporting Tunnel ID attribute to the lower layer network TE tunnel providing the transport connectivity for the TE link. Likewise, a TE tunnel has an attribute pointing to the TE link it supports, as well as the TE topology which the TE link is part of. These cross-references are instrumental for the client in terms of understanding which network resources a given TE link represents, especially useful at the times of trouble shooting. Additionally, IETF TE tunnel defines and supports the concept of Hierarchical TE links and tunnels. Hierarchical TE tunnels automatically insert dynamic hierarchical TE links into the specified TE topologies as soon as the tunnels are successfully set up (and remove the hierarchical TE links from the respective TE topologies when released). [Note: Hierarchical TE tunnels and links are instrumental in multi-layer traffic engineering].

Furthermore, both TE topology and TE tunnel models are tightly coupled with the IETF YANG based notification machinery, which allows the client to retrieve any telemetry or attribute change updates as long as those telemetry/attribute changes are defined as data state nodes or sub-trees in the respective models.

In contrast, all T-API services (i.e. Topology, Connectivity, Path computation, Virtual Network and Notification) are independent from each other.

2.9. Topology Negotiation and (Re-)configuration

When a client of the IETF TE Topology model/interface receives one or more abstract TE topologies from one of its providers, it may accept the topologies as-is and merge then into one or more of its own native TE topologies. Alternatively, the client may choose to request a re-configuration of one, some or all abstract TE topologies provided by the providers. Specifically, with respect to a given abstract TE topology, some of its TE nodes/links may be requested to be removed, while additional ones may be requested to be added. It is also possible that existing TE nodes/links may be asked to be re-configured (e.g., TE links may be requested to be SRLG disjoint).

Furthermore, the topology-wide optimization criteria may be requested to be changed. For example, underlay TE paths supporting the abstract TE links, currently optimized to be shortest (least-cost) paths, may be requested to be re-optimized based on the minimal delay criteria. Additionally, the client may request the providers to configure entirely new abstract TE topologies and/or to remove existing ones. Furthermore, future periodic or one-time additions, removals and/or re-configurations of abstract TE topologies, topological elements and/or their attributes could be (re-)scheduled by the client ahead of time.

It is the responsibility of the client to implement the logic behind the above-described abstract TE topology negotiation. It is expected that the logic is influenced by the client's local configuration/ templates, policies conveyed by the client's clients, input from the network planning process, telemetry processor, analytics systems and/ or direct human operator commands. Figure 6 exemplifies the abstract TE topology negotiation process. As shown in the Figure, the original abstract TE topology exposed by a provider was requested to be re-configured. Specifically, one of the abstract TE links was asked to be removed, while three new ones were asked to be added to the abstract TE topology.

The ONF T-API Topology service client has no say as to how the abstract topologies exposed to the client by its providers should look like. The only option for the client is to consume the provided topologies as offered. This is a serious disadvantage because it is the client (not providers) that knows which topologies suite best the client's needs.

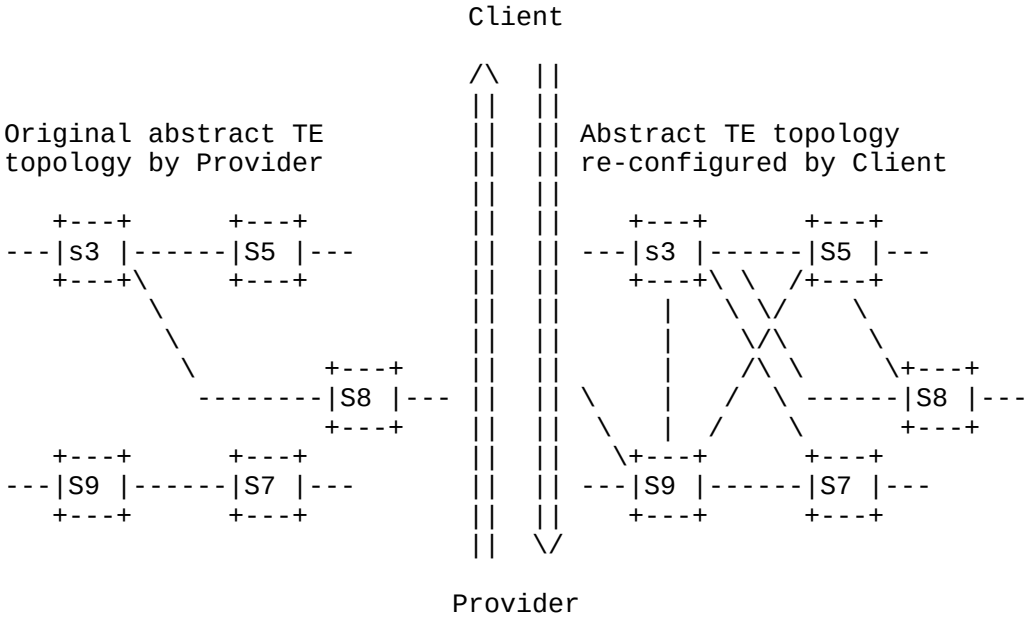


Figure 6: Provider-Client abstract TE topology negotiation

2.10. Integration with IP/MPLS

The IETF TE Topology model is naturally and intimately integrated with IP/MPLS layer models defined for IP/MPLS layer traffic engineering. For example, currently Segment Routing (SR) and Service Function Chaining (SFC) technologies heavily rely on and actively use the TE Topology model. Specifically, SR combines the TE topology model with layer 3 (IP reachability) topology model to facilitate path computations that account for either or both TE and IP reachability information. Likewise, SFC makes use of the TE topology model for computing service function chains optimized according to the combined criteria of real/virtual network function location and best available (possibly in different layers) TE paths to connect the network functions.

It is not clear how the ONF T-API Topology service can fit in and to what extent it can be integrated into the IP/MPLS layer traffic engineering.

3. Connectivity Service

3.1. Connectivity Service Protection

It is not possible for a T-API Connectivity service client to request from a provider a protected service like, for example, the one presented in Figure 7. In the Figure a connectivity service is supported by two disjoint connections - primary (solid blue) and backup (broken yellow), with the client traffic normally carried over the primary connection, but which could be quickly and dynamically switched onto the backup connection as soon as a network failure affecting the primary connection is detected.

The inability to request protected connectivity services from a provider leaves the T-API Connectivity service client with the problem of protecting its own traffic against the network's failures. Admittedly, the client can address this with the following sequence of operations:

1. The client requests a primary connectivity service connecting the desired pair of client device ports over the network managed by the T-API Connectivity service provider;

2. The client requests a secondary connectivity service connecting the same pair of client device ports, which is sufficiently diverse from the primary service (incidentally, this could be problematic due to the independent nature of the path computations carried out by the provider. Specifically, the path selected for the primary service may block disjoint paths for the secondary service. This is a known issue related to sequential/ independent path computations, which could be solved via concurrent path computation for both services);

3. The client binds at both ends the two connectivity services in accordance with the desired protection scheme;

4. From then on the client is constantly monitoring the performance and health of both services;

5. In case the primary service is affected by a network failure (while the secondary service remaining healthy), the client coordinates the protection switchover;

6. In case it is detected that the previously broken primary connectivity service is repaired, the client coordinates the protection reversion (i.e. reversion to the normal forwarding of the client traffic).

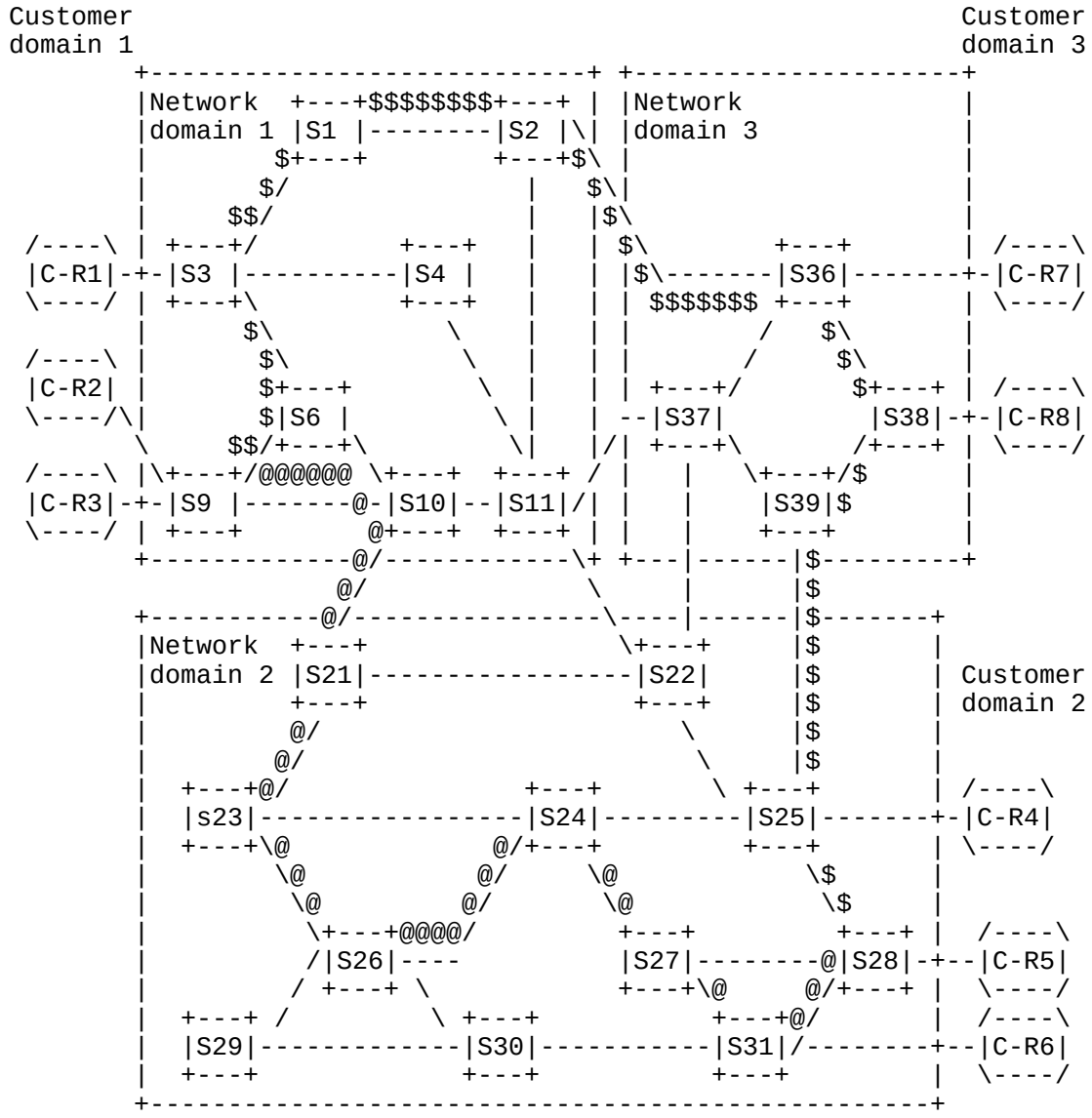


Figure 7: Protected connectivity service

In contrast, an IETF TE tunnel model client normally delegates all the described above operations to the provider by simply configuring the requested transport service (i.e. TE tunnel or a single-domain segment of a multi-domain TE tunnel) to be protected. In doing so the client specifies the required protection type, as well as the level of primary/backup connections disjointedness. Additionally,

the client may specify a set of constraints common for both connections, as well as constraints (e.g. inclusions, exclusions, etc) specific to each connection. Furthermore, the client may even specify, for a given transport service, multiple sets of such constraints in descending preference order for the provider to try before notifying the client about the setup failure. For example, the client may request in this way for a TE tunnel that the primary and backup connections must be SRLG disjoint, and, if this proves to be not possible, to relax the disjointedness criterion to link-disjoint.

3.2. Hierarchical Connectivity Service

A transport network provider may control a multi-layer (e.g. Ethernet/ODUk/OCh) network. On such a network the provider has flexibility to dynamically set up connectivity/transport services in one or more lower layer networks to augment a higher layer topology that is otherwise insufficient for provisioning of a connectivity service requested by the client.

In the top-to-bottom approach the client simply requests a connectivity service in the desired layer network. While processing the request, the provider:

- performs its internal multi-layer path computation,
- identifies one or more lower layer connectivity services required for the successful provisioning of the requested service;
- dynamically (and unknowingly to the client) sets up the so-identified lower layer connections;
- sets up the connection(s) supporting the connectivity service requested by the client.

Both T-API Connectivity service and IETF TE Topology model/interface support the described top-to-bottom multi-layer connectivity services. The approach is simple for the client; however it does not work in many multi-domain use cases. Consider, for example, a multi-domain transport network presented in Figure 8. Consider further that a Multi-Domain Service Coordinator is requested to set up Ethernet layer connectivity service (marked in blue) across three domains, each of which is controlled by a separate provider. Assume also that in order to satisfy the request an underlay ODUk layer TE tunnel (marked as red) also spanning multiple domains needs to be provisioned. This could be achieved via a bottom-to-top multi-layer connectivity service provisioning approach, which includes the following:

- the client (i.e. the Multi-Domain Coordinator) performs its own multi-layer path computation on a network wide TE topology (a product of merging the TE topologies exposed by all providers);
- the client identifies one or more lower layer TE tunnels required for the successful provisioning of the requested service;
- the client coordinates the multi-domain setup of each of the identified lower layer TE tunnels;
- the client instructs each lower layer TE tunnel’s first and last domain provider to add a dynamic TE link in their respective higher layer TE topologies;
- the client triggers and coordinates the setup of the connection(s) supporting the requested connectivity service, constraining the connection path(s) to follow the dynamic TE links supported by the lower layer TE tunnels;
- the client adds into its own (network-wide) TE topology, dynamic TE links supported by the lower layer TE tunnels to make the remaining capacity on the tunnels available for path computations for other higher layer connectivity services.

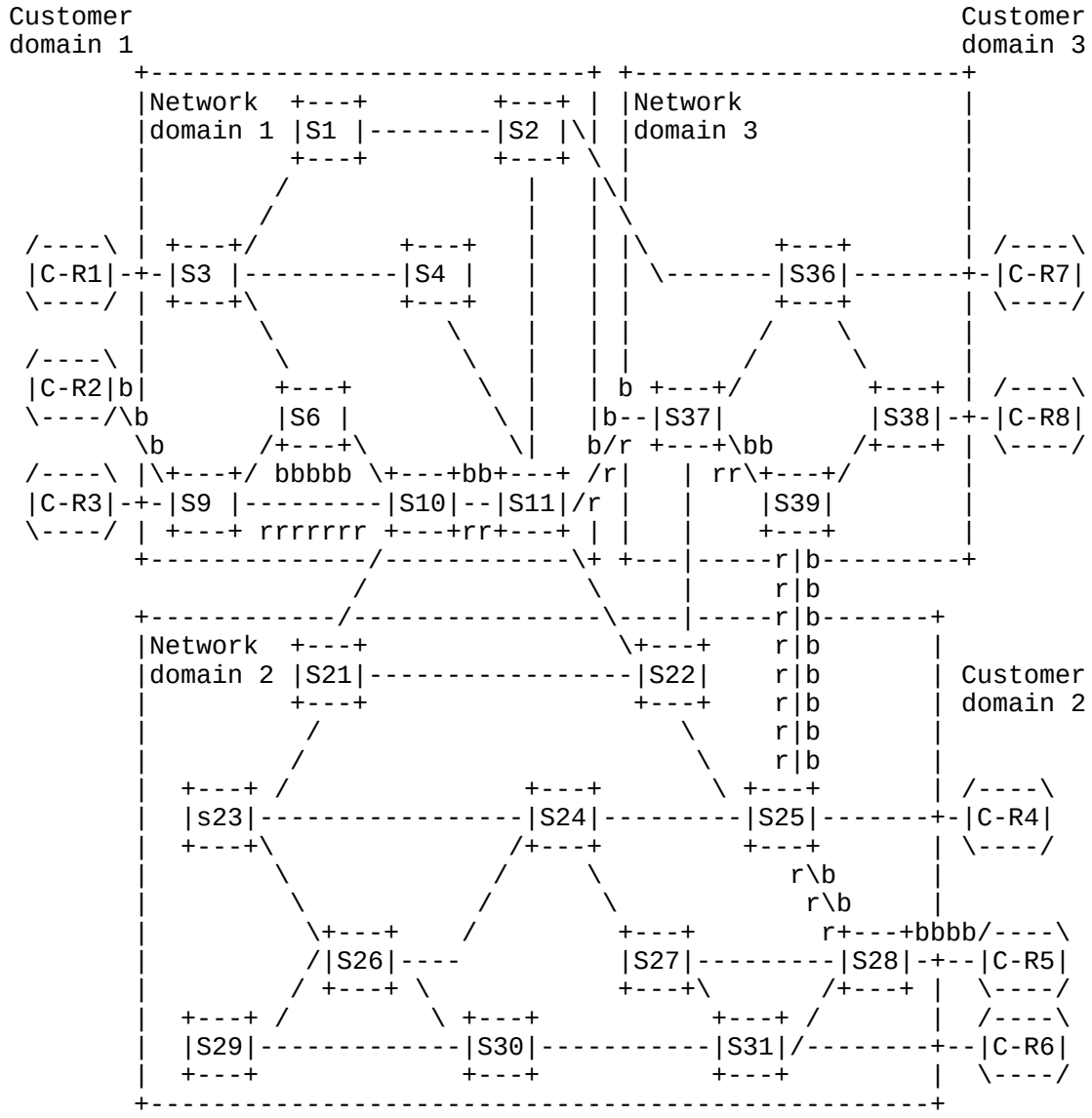


Figure 8: Hierarchical connectivity service

The IETF TE topology model supports the described bottom-to-top multi-layer connectivity service provisioning paradigm via Hierarchy TE tunnels. A hierarchy TE tunnel, once successfully set up, automatically adds into the specified TE topology a TE link it supports and withdraws the TE link from the TE topology if/when released.

T-API Connectivity and Topology services do not support the concept of hierarchical connectivity/dynamic links.

3.3. Connectivity Service Re-optimization

An IETF TE tunnel model/interface client, when requesting a transport service from a provider, can control - via a designed for this purpose knob (lockDown attribute) - whether the connection(s) supporting the service must be "pinned" to their respective original paths (the paths selected at the setup stage), or whether the provider may occasionally perform a service re-optimization, resulting in service connection replacement toward more optimal paths. This knob is especially useful in conjunction with a connectivity scheduling service (see section 2.6), allowing for the client to specify time intervals at which the re-optimization of a given transport service (and subsequent potential traffic hits) is acceptable for the client. For example, the client may configure a transport service to get "unpinned" every Saturday at 1 am for service re-optimization procedures and to get "re-pinned" after that for another week.

T-API Connectivity service clients have no way of controlling of connectivity service re-optimization operations.

3.4. Connectivity Service Templates

The IETF TE tunnel model defines containers of named transport service configuration sets that could be shared by multiple services. This not only simplifies for the client the process of transport service configuration, but also allows manipulation of multiple services by a single configuration change. For example, a client may define a set of constraints named Foo that forces a transport service primary path to go through a node X. If, later, the client modifies Foo by substituting node X with node Y, all transport services configured with the constraint set Foo will (be attempted to) be re-placed onto path(s) going through node Y.

The T-API Connectivity service model does not have a similar concept.

3.5. Connectivity Service Attribute Change Update Notifications and Telemetry Streaming

Both T-API and IETF modeling rely on respective notification tools universal across all interfaces. Therefore, connectivity service attribute change notifications and telemetry streaming is no different from the topology notifications and telemetry streaming discussed in sections 2.3 and 2.4

3.6. Connectivity Scheduling

T-API Connectivity service has the _schedule attribute that includes just two parameters: startTime and endTime. This allows for a client to schedule at a specified time and for a specified period of time a one-time kickoff of a service configured initially (presumably) as disabled. It is not possible to schedule multi-time (periodic) kickoffs. Furthermore, the scheduling granularity is connectivity service as a whole. In particular, it is not possible to schedule re-configurations of one or several service parameters (e.g. bandwidth requirement, inclusion/exclusion path, etc.).

There is an ongoing effort in IETF to produce a generic scheduling tool that could be applied to any of YANG models. Similar to the notification subscription tool - allowing for the client to subscribe on notifications with respect to any data state (CONFIG=FALSE) node defined in any supported by the provider data store - the scheduling tool will allow for the client to schedule periodic and/or one-time modification of any configuration (CONFIG=TRUE) leaf of any supported data store. For example, if it is required to schedule a re-configuration of the bandwidth requirement for one or more selected services, the client will specify an XPath pointing to the configured bandwidth attribute of the services of interest and convey the new bandwidth requirement and the timetable for the service bandwidth re-configuration. [Note: At time intervals outside of the scheduled range, the service configured bandwidth will remain/be restored to the value provided during initial service configuration.]

3.7. Potential Connectivity Service

The IETF TE topology model defines a number of "unconventional" configuration modes to be specified by a client and supported by a provider of transport services. One of those modes is the COMPUTE_ONLY mode. When a provider processes a request for a transport service configured in the COMPUTE_ONLY mode, it performs the normal path computation for the service, but does not trigger setup of the connection(s) supporting the service. Instead, the computed paths are returned to the client as a part of normal service attribute change notification. Furthermore, when the provider detects a change in the managed network potentially affecting the returned paths, it may re-evaluate the paths and notify the client if they have become infeasible or more optimal paths are available.

The concept of COMPUTE_ONLY transport services makes a good foundation for Path computation service/interface between the Client and the Provider (see more in section 4).

4. Path Computation Service

A client of a transport network can discover the network resources available for the client in one of the two ways:

- by requesting from the network provider , via a topology interface, one or more topologies describing the network with respect to its availability to the client;

or

- by requesting, via a path computation interface, that the provider identify potential paths that could connect various client device ports across the network.

To support the latter option, ONF T-API has introduced a Path computation service dedicated to the purpose. A T-API Path computation service client can issue a path computation request specifying the identities of the required path source and destination end points, the layer network in which the paths are to be determined, the required mutual diversity of the resulting paths, various path computation constraints (e.g., bandwidth requirements, inclusions, exclusions, etc.) and path selection optimization criteria (e.g., smallest cost, shortest delay, etc.). A T-API Path computation service provider is expected to satisfy the request by running a path computation algorithm and responding to the client with zero, one or more resulting paths.

In contrast, IETF modeling does not offer a dedicated mechanism/model to support the Client<=>Provider path computation interface. Instead, it is suggested to use the YANG TE tunnel model and request and manipulate path computations in the form of COMPUTE_ONLY TE tunnels as described in section 2.7. This approach has some important advantages as compared to the T-API Path computation service:

Simplicity: provided that both the client and the provider know how to request, manipulate and support transport services, there is no additional interface/model for the client to learn how to use and functionality for the provider to support;

Accuracy: T-API Path computation and Connectivity services are not related. It cannot be guaranteed that the set of path computation constraints conveyed by a T-API Path computation service client will match the set of path computation constraints internally generated by a T-API Connectivity service provider even when the configuration parameters - source/destination, layer network,

bandwidth and others - match. There are many reasons for that, including:

A. additional constraints could be imposed by the provider based on some internal and possibly proprietary knowledge about the network (unknown to the client);

B. various internal policies could relax, harden or overwrite other constraints;

C. various internal policies could modify or overwrite the requested optimization criteria;

D. etc.

Furthermore, the provider may even use different path computation engines to provide the Path computation and connectivity services. All this may result in the paths returned to the Path computation service client being different from the paths taken by the corresponding (same source/destination and other constraints) connectivity services. The difference may be in path costs, delay and fate sharing characteristics, etc. In extreme cases the Path computation service client may even receive unprovisionable and hence useless paths.

IETF COMPUTE_ONLY TE tunnels, on the other hand, do not have such problems. It is inherently guaranteed that the client will be notified/updated with paths which are exactly the same as the ones that would be taken by connections of "conventional" TE tunnels for the same configuration inputs;

Path staleness: paths returned to the T-API Path computation service client may become unfeasible at some later time because of changes in the network’s state. There is no way for the Path computation service provider to convey this fact to the client. In contrast, IETF COMPUTE_ONLY TE tunnel provider can use the intrinsic attribute change notifications to let the client know that previously provided paths have changed, have become unfeasible or that better, more optimal paths have become available.

5. Virtual Network Service

A client of a transport network may want to limit the transport network connectivity of a particular type and quality to defined subsets of its device ports interconnected across the network. Furthermore, a given transport network may serve more than one client. In this case some or all clients may want to ensure the availability of transport network resources in case dynamic

(re-)connection of their device ports across the network is envisioned. In all such cases a client may want to set up one or more Virtual Networks over the provided transport network.

ONF T-API has introduced a dedicated service for this purpose - the Virtual Network service (VNS). A VNS client can request creation of a VNS specifying the layer network of the VNS and the Traffic Matrix requirement. The client has no control over the requested VN beyond that. In particular, it is up to the provider to decide which network resources will support the VN in question. The client has no say as to how the underlying network topology should look, how the topology needs to be optimized for the VN (e.g. shortest delay rather than smallest cost), what is the required level of the topology link protection and mutual diversity, and so forth.

As in case of the path computation interface, IETF modeling does not offer a separate model to support VNS. Instead, it encourages using the TE topology model - leveraging the IETF abstract TE topology's ability to be configured by the client. In a nutshell, the client configures and manipulates a VN as a customized abstract TE topology based on the TE topologies already exposed by the provider. In the simplest case the client requests a single node ("black box") abstract TE topology with desired attributes. In more complex cases the client may opt to construct, for the VN, a separate multi-node/ link arbitrary abstract TE topology. In doing so, the client may "borrow" into the VN's topology TE nodes and links from other topologies. Additionally the client may add new composite abstract TE nodes specifying the IDs of TE topologies the nodes will encapsulate, connected by abstract TE links pointing to the respective underlay TE topologies to be used for computation and provisioning of the TE tunnels supporting them. The client/provider negotiation of a"so-cooked" TE topology is described in 1.9. In short, the client is able to manipulate the VN's topology at the granularity of individual topological elements (such as TE nodes and links).

6. Data Modeling Language

Today YANG is a very popular data modeling language. It is a product of IETF NETMOD WG. It is not the only data modeling language produced by IETF (for example, FORCES WG has developed one of its own, arguably - in some aspects - superior to YANG). YANG is neither stable nor perfect. It is constantly evolving with the sole objective to make IETF models more scalable, efficient, inclusive, information-rich: better in all aspects. Supporting non-IETF (e.g. ONF) data models is not a priority. Therefore It is not clear why ONF, while investing a lot of effort in designing Core Information Models, is devoting no effort to designing a data modeling language

of its own that would closely suit support of its CIM. Nor it is clear what would happen if the IETF NETMOD WG decides, for whatever reason to obsolete some of the YANG features/properties/capabilities that ONF models rely upon.

Furthermore, writing CIMs in UML and having them mechanically translated into YANG has its own issues, which includes the following:

- Many useful YANG features that do not have analogs in UML are not used. For example, T-API YANG models use only non-extendible enumeration type, rather than extendible identity type. This prevents T-API YANG models from being easily extendible via augmentation;

- T-API YANG models heavily overuse and often misuse YANG RPCs for operations that could be handled simpler and more efficiently by NETCONF/RESTCONF protocol via native edit-config and get operations;

- T-API YANG models unnecessarily define their own notification subscription/streaming and scheduling mechanisms, instead of leveraging the NETCONF/RESTCONF machinery easily applicable to all YANG models;

- T-API YANG models make no use of YANG templates and defaults designed to simplify for the client the provider’s data store (re-)configuration;

- T-API YANG models follow the conventions inherited from UML and previously defined REST APIs. As a consequence. the models sometimes are not compatible with the current best practices recommended for YANG model writers and do not always follow YANG model guidelines defined in [I-D.ietf-netmod-rfc6087bis]

7. Security Framework

ONF T-API does not have a security framework of its own. It simply assumes that the proper security could be inherently provided by the underlying protocols. IETF TEAS interfaces, on the other hand, take the security considerations very seriously. They rely on the generic framework ([RFC6241], [RFC8040], [RFC6536], and [I-D.ietf-netconf-rfc6536bis]) allowing for the provider to configure in a universal way various strength AAA protection for any YANG modeled data store accessible via NETCONF or RESTCONF protocol. In particular, said framework allows for the client authentication, identification of the client’s privileges with respect to the

information access, required filtering and scoping of the provided information, as well as secure client-provider communication.

8. IANA Considerations

This document has no actions for IANA.

9. Security Considerations

This document does not define networking protocols and data, hence are not directly responsible for security risks.

This document compares two interface technologies of T-SDN controllers. For each specific technology discussed in the document, security framework has been described and compared in the corresponding section.

10. Acknowledgements

The authors would like to thank Christopher Jenz, Diego Caviglia, Aihua Guo, Fatai Zhang, and Italo Busi for their helpful comments and valuable contributions.

11. References

11.1. Normative References

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <http://www.rfc-editor.org/info/rfc8040>.

[I-D.ietf-teas-yang-te-topo]
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for TE Topologies", draft-ietf-teas-yang-te-topo-06 (work in progress), October 2016.

[I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., Bryskin, I., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-05 (work in progress), October 2016.

[I-D.ietf-netconf-rfc6536bis]
Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", draft-ietf-netconf-rfc6536bis-01 (work in progress), March 2017.

11.2. Informative References

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-12 (work in progress), March 2017.

Authors' Addresses

Igor Bryskin
Huawei Technologies

EMail: Igor.Bryskin@huawei.com

Xufeng Liu
Jabil

EMail: Xufeng_Liu@jabil.com

Vishnu Pavan Beeram
Juniper Networks

EMail: vbeeram@juniper.net

Tarek Saad
Cisco Systems Inc

EMail: tsaad@cisco.com

TEAS Working Group
Internet-Draft
Intended Status: Standards Track
Expires: September 11, 2017

R. Gandhi, Ed.
Cisco Systems, Inc.
H. Shah
Ciena
Jeremy Whittaker
Verizon

March 10, 2017

Fast Reroute Procedures For Associated Bidirectional Label Switched Paths (LSPs)
draft-gandhishah-teas-assoc-corouted-bidir-04

Abstract

Resource Reservation Protocol (RSVP) association signaling can be used to bind two unidirectional LSPs into an associated bidirectional LSP. When an associated bidirectional LSP is co-routed, the reverse LSP follows the same path as its forward LSP. This document describes Fast Reroute (FRR) procedures for both single-sided and double-sided provisioned associated bidirectional LSPs. The FRR procedures are applicable to co-routed and non co-routed LSPs. For co-routed LSPs, the FRR procedures can ensure that traffic flows on co-routed paths in the forward and reverse directions after a failure event.

Status of this Memo

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

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

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

Table of Contents

1. Introduction

The Resource Reservation Protocol (RSVP) (Extended) ASSOCIATION Object is specified in [RFC6780] which can be used generically to associate (G)Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs). [RFC7551] defines mechanisms for binding two point-to-point unidirectional LSPs [RFC3209] into an associated bidirectional LSP. There are two models described in [RFC7551] for provisioning an associated bidirectional LSP, single-sided and double-sided. In both models, the reverse LSP of the bidirectional LSP may or may not be co-routed and follow the same path as its forward LSP.

[GMPLS-FRR] defines Fast Reroute (FRR) procedure for GMPLS signaled LSPs to co-ordinate bypass tunnel assignments in the forward and reverse directions. The mechanisms defined in [GMPLS-FRR] are applicable to FRR of associated bidirectional LSPs.

In packet transport networks, there are requirements where the reverse LSP of a bidirectional LSP needs to follow the same path as its forward LSP [RFC6373]. The MPLS Transport Profile (TP) [RFC6370] architecture facilitates the co-routed bidirectional LSP by using the GMPLS extensions [RFC3473] to achieve congruent paths. However, the RSVP association signaling allows to enable co-routed bidirectional LSPs without having to deploy GMPLS extensions in the existing networks. The association signaling also allows to take advantage of the existing Traffic Engineering (TE) and FRR mechanisms in the network.

This document describes FRR procedures for both single-sided and double-sided provisioned associated bidirectional LSPs. The FRR procedures are applicable to co-routed and non co-routed LSPs. For co-routed LSPs, the FRR procedures can ensure that traffic flows on co-routed paths in the forward and reverse directions after a failure event.

2. Conventions Used in This Document

2.1. Key Word Definitions

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

2.2. Terminology

The reader is assumed to be familiar with the terminology in [RFC2205], [RFC3209], [RFC4090] and [RFC7551].

2.2.1. Reverse Co-routed Unidirectional LSPs

Two reverse unidirectional point-to-point (P2P) LSPs are setup in the opposite directions between a pair of source and destination nodes to form an associated bidirectional LSP. A reverse unidirectional LSP originates on the same node where the forward unidirectional LSP terminates, and it terminates on the same node where the forward unidirectional LSP originates. A reverse co-routed unidirectional LSP traverses along the same path of the forward direction unidirectional LSP in the opposite direction.

3. Overview

As specified in [RFC7551], in the single-sided provisioning case, the RSVP TE tunnel is configured only on one endpoint node of the bidirectional LSP. An LSP for this tunnel is initiated by the originating endpoint with (Extended) ASSOCIATION Object containing Association Type set to "single-sided associated bidirectional LSP" and REVERSE_LSP Object inserted in the Path message. The remote endpoint then creates the corresponding reverse TE tunnel and signals the reverse LSP in response using the information from the REVERSE_LSP Object and other objects present in the received Path message. As specified in [RFC7551], in the double-sided provisioning case, the RSVP TE tunnel is configured on both endpoint nodes of the bidirectional LSP. Both forward and reverse LSPs are initiated independently by the two endpoints with (Extended) ASSOCIATION Object containing Association Type set to "double-sided associated bidirectional LSP". In both single-sided and double-sided provisioned bidirectional LSPs, the reverse LSP may or may not be congruent (i.e. co-routed) and follow the same path as its forward LSP.

In the case of single-sided provisioned LSP, the originating LSP with REVERSE_LSP Object is identified as a forward LSP. In the case of double-sided provisioned LSP, the LSP originating from the higher node address (as source) and terminating on the lower node address (as destination) is identified as a forward LSP. The reverse LSP of the bidirectional LSP traverses in the opposite direction of the forward LSP.

Both single-sided and double-sided associated bidirectional LSPs require solutions to the following issues for fast reroute.

3.1. Fast Reroute Bypass Tunnel Assignment

In order to ensure that the traffic flows on a co-routed path after a link or node failure on the protected LSP path, the mid-point Point

of Local Repair (PLR) nodes need to assign matching bidirectional bypass tunnels for fast reroute. Even for a non co-routed bidirectional LSP, it is desired that the same bidirectional bypass tunnel is used in both directions of the protected LSP. Such bypass assignment requires co-ordination between the forward and reverse direction PLR nodes when more than one bypass tunnels are present on a PLR node.

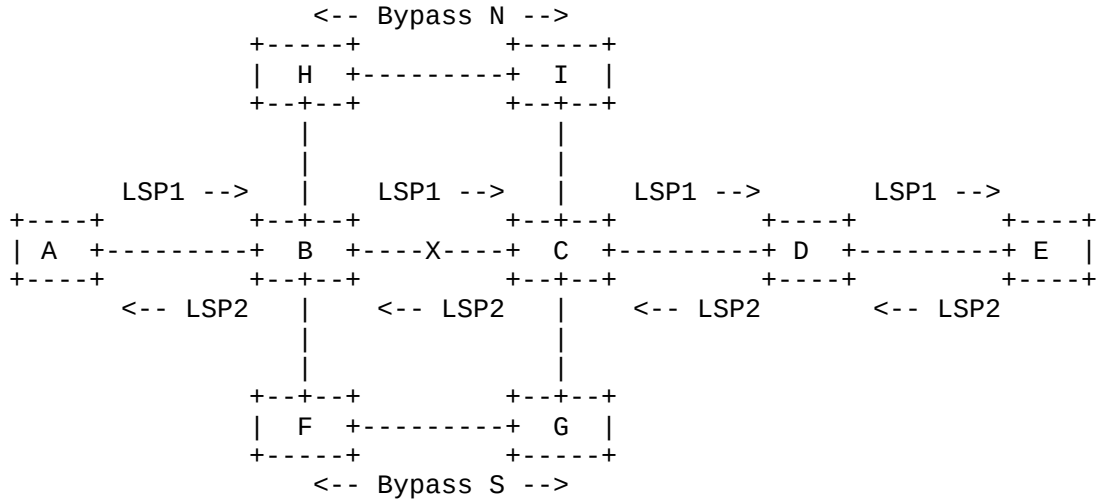


Figure 1: Multiple Bidirectional Bypass Tunnels

As shown in Figure 1, there are two bypass tunnels available, Bypass N on path B-H-I-C and Bypass S on path B-F-G-C. The mid-point PLR nodes B and C need to co-ordinate bypass tunnel assignment to ensure that traffic in both directions flow through either on the Bypass N path B-H-I-C or the Bypass S path B-F-G-C, after the link B-C failure.

3.2. Bidirectional LSP Association At Mid-Points

In packet transport networks, a restoration LSP is signaled after a link failure on the protected LSP and the protected LSP may or may not be torn down [GMPLS-REST]. In this case, multiple forward and reverse LSPs of a bidirectional LSP may be present at mid-point nodes with identical (Extended) ASSOCIATION Objects. This creates an ambiguity at mid-point nodes to identify the correct associated LSP pair for fast reroute bypass assignment (e.g. during the recovery phase of RSVP graceful restart procedure).

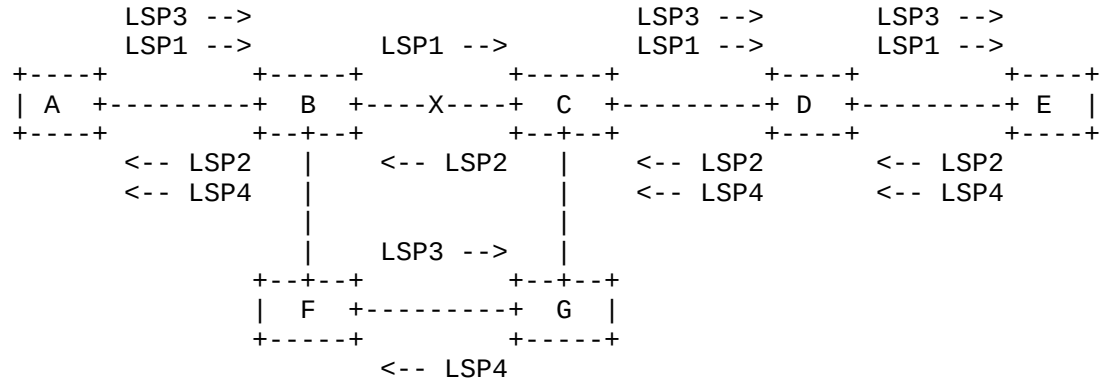


Figure 2: Restoration LSP Set-up After Link Failure

As shown in Figure 2, protected LSPs LSP1 and LSP2 are an associated LSP pair, similarly restoration LSPs LSP3 and LSP4 are an associated LSP pair, both pairs belong to the same associated bidirectional LSP and carry identical (Extended) ASSOCIATION Objects. In this example, mid-point node D may mistakenly associate LSP1 with reverse LSP4 instead of reverse LSP3 due to the matching (Extended) ASSOCIATION Objects. This may cause the bidirectional LSP to become non co-routed. Since a reverse LSP reflects the bypass tunnel assignment received in the forward LSP, this can also lead to undesired bypass tunnel assignments.

4. Signaling Procedure

4.1. Bidirectional LSP Fast Reroute

The mechanisms defined in [GMPLS-FRR] are used for fast reroute of both single-sided and double-sided associated bidirectional LSPs as following.

- As described in [GMPLS-FRR], BYPASS_ASSIGNMENT subobject is signaled in the RRO of the Path message to co-ordinate bypass tunnel assignment between the forward and reverse direction PLR nodes. A BYPASS_ASSIGNMENT subobject MUST be added by the forward direction PLR node in the Path message of the forward LSP to indicate the bypass tunnel assigned.

- The forward direction PLR node always initiates the bypass tunnel assignment for the forward LSP. The reverse direction PLR (forward direction LSP Merge Point (MP)) node simply reflects the bypass tunnel assignment for the reverse direction LSP.

- o After a link or node failure, the PLR nodes in both forward and reverse directions trigger fast reroute independently using the procedures defined in [RFC4090].

- o When using a node protection bypass tunnel, asymmetry of paths can occur in the forward and reverse directions of the bidirectional LSP after a link failure when using co-routed LSPs [GMPLS-FRR]. This can be corrected using the re-corouting procedure defined in [GMPLS-FRR]. Unlike GMPLS LSPs, the asymmetry of paths in the forward and reverse directions does not result in RSVP soft-state time-out with the associated bidirectional LSPs.

4.2. Bidirectional LSP Association At Mid-points

In order to associate the correct LSPs at a mid-point node, an endpoint node MUST signal Extended ASSOCIATION Object and add unique Extended Association ID for each associated forward and reverse LSP pair forming the bidirectional LSP. As an example, an endpoint node MAY set the Extended Association ID to the value specified in Section 5.1 of this document.

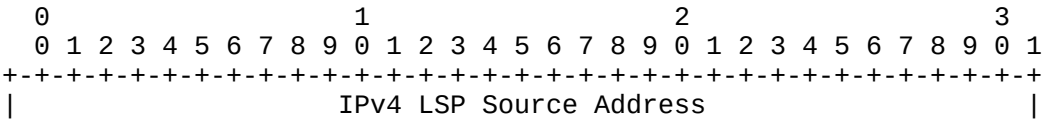
- o For single-sided provisioned bidirectional LSPs [RFC7551], the originating endpoint signals the Extended ASSOCIATION Object with a unique Extended Association ID. The remote endpoint copies the contents of the received Extended ASSOCIATION Object including the Extended Association ID in the RSVP Path message of the reverse LSP's Extended ASSOCIATION Object.

- o For double-sided provisioned bidirectional LSPs [RFC7551], both endpoints need to ensure that the bidirectional LSP has a unique Extended ASSOCIATION Object for each forward and reverse LSP pair by provisioning appropriate Extended Association IDs signaled by them.

5. Message and Object Definitions

5.1. Extended ASSOCIATION Object

The Extended Association ID in the Extended ASSOCIATION Object can be set to the value specified as following to uniquely identify associated forward and reverse LSP pair of a bidirectional LSP.



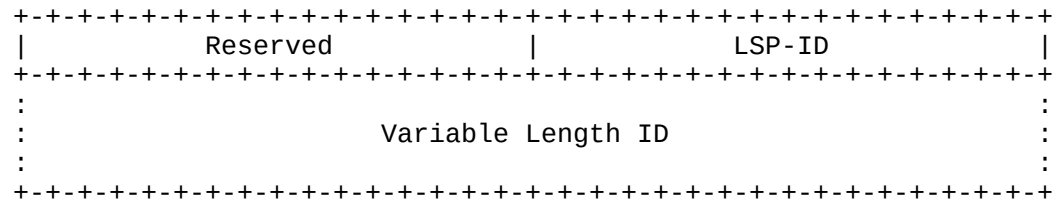


Figure 3: IPv4 Extended Association ID

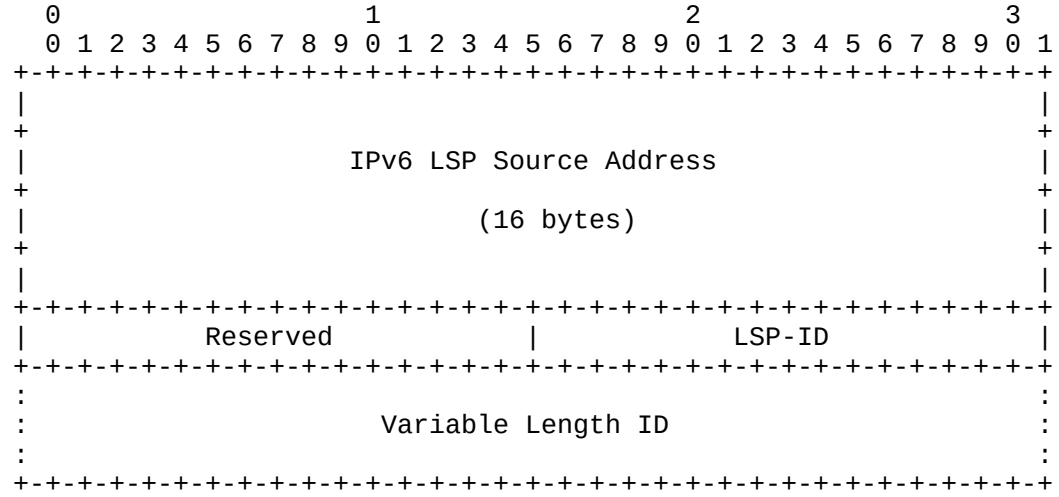


Figure 4: IPv6 Extended Association ID

LSP Source Address

IPv4/IPv6 source address of the forward LSP.

LSP-ID

16-bits LSP-ID of the forward LSP.

Variable Length ID

Variable length ID inserted by the endpoint node of the associated bidirectional LSP [RFC6780].

6. Compatibility

This document describes the procedures for fast reroute for associated bidirectional LSPs. Operators wishing to use this function SHOULD ensure that it is supported on the nodes on the LSP path.

7. Security Considerations

This document uses signaling mechanisms defined in [RFC7551] and [GMPLS-FRR] and does not introduce any additional security considerations other than already covered in [RFC7551], [GMPLS-FRR] and the MPLS/GMPLS security framework [RFC5920].

8. IANA Considerations

This document does not make any request for IANA action.

9. References

9.1. Normative References

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

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

[RFC4090] Pan, P., Ed., Swallow, G., Ed., and A. Atlas, Ed., "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, May 2005.

[RFC6780] Berger, L., Le Faucheur, F., and A. Narayanan, "RSVP Association Object Extensions", RFC 6780, October 2012.

[RFC7551] Zhang, F., Ed., Jing, R., and Gandhi, R., Ed., "RSVP-TE Extensions for Associated Bidirectional LSPs", RFC 7551, May 2015.

[GMPLS-FRR] Taillon, M., Saad, T., Ed., Gandhi, R., Ed., Ali, Z., Bhatia, M., "Extensions to Resource Reservation Protocol For Fast Reroute of Traffic Engineering GMPLS LSPs", draft-ietf-teas-gmpls-lsp-fastreroute, work in progress.

9.2. Informative References

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

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

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

[RFC6370] Bocci, M., Swallow, G., and E. Gray, "MPLS Transport Profile (MPLS-TP) Identifiers", RFC 6370, September 2011.

[RFC6373] Andersson, L., Berger, L., Fang, L., Bitar, N., and E. Gray, "MPLS Transport Profile (MPLS-TP) Control Plane Framework", RFC 6373, September 2011.

[GMPLS-REST] Zhang, X., Zheng, H., Ed., Gandhi, R., Ed., Ali, Z.,

Brzozowski, P., "RSVP-TE Signaling Procedure for End-to-End GMPLS Restoration and Resource Sharing", draft-ietf-teas-gmpls-resource-sharing-proc, work in progress.

Authors' Addresses

Rakesh Gandhi (editor)
Cisco Systems, Inc.

EMail: rgandhi@cisco.com

Himanshu Shah
Ciena

EMail: hshah@ciena.com

Jeremy Whittaker
Verizon

EMail: jeremy.whittaker@verizon.com

TEAS Working Group
Internet Draft
Intended status: Informational
Expires: December 2017

Daniele Ceccarelli (Ed)
Ericsson
Young Lee (Ed)
Huawei

June 13, 2017

Framework for Abstraction and Control of Traffic Engineered Networks

draft-ietf-teas-actn-framework-06

Abstract

Traffic Engineered networks have a variety of mechanisms to facilitate the separation of the data plane and control plane. They also have a range of management and provisioning protocols to configure and activate network resources. These mechanisms represent key technologies for enabling flexible and dynamic networking.

Abstraction of network resources is a technique that can be applied to a single network domain or across multiple domains to create a single virtualized network that is under the control of a network operator or the customer of the operator that actually owns the network resources.

This document provides a framework for Abstraction and Control of Traffic Engineered Networks (ACTN).

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire on December 13, 2017.

Table of Contents

1. Introduction

Traffic Engineered 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.

The term Traffic Engineered network is used in this document to refer to a network that uses any connection-oriented technology under the control of a distributed or centralized control plane to support dynamic provisioning of end-to-end connectivity. Some examples of networks that are in scope of this definition are optical networks, MPLS Transport Profile (MPLS-TP) networks [RFC5654], and MPLS Traffic Engineering (MPLS-TE) networks [RFC2702].

One of the main drivers for Software Defined Networking (SDN) [RFC7149] 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 the Path Computation Element (PCE) [RFC4655] for TE-based networks. One of the advantages of SDN is its logically centralized control regime that allows a global view of the underlying networks. Centralized control in SDN helps improve network resource utilization compared with distributed network control. For TE-based networks, PCE is essentially equivalent to a logically centralized path computation function.

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

. Separation of service requests from service delivery so that the orchestration of a network is transparent from the point of view of the customer but remains responsive to the customer's services and business needs.

. Network abstraction: As described in [RFC7926], abstraction is the process of applying policy to a set of information about a TE network to produce selective information that represents the potential ability to connect across the domain. The process of abstraction presents the connectivity graph in a way that is independent of the underlying network technologies, capabilities, and topology so that it can be used to plan and deliver network services in a uniform way

. Coordination of resources across multiple domains and multiple layers to provide end-to-end services regardless of whether the domains use SDN or not.

As networks evolve, the need to provide separated service request/orchestration and resource abstraction has emerged as a key requirement for operators. In order to support multiple clients each with its own view of and control of the server network, a network operator needs to partition (or "slice") the network resources. The resulting slices can be assigned to each client for guaranteed usage which is a step further than shared use of common network resources.

Furthermore, each network represented to a client can be built from abstractions of the underlying networks so that, for example, a link in the client's network is constructed from a path or collection of paths in the underlying network.

We call the set of management and control functions used to provide these features Abstraction and Control of Traffic Engineered Networks (ACTN).

Particular attention needs to be paid to the multi-domain case, 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.

The ACTN framework described in this document facilitates:

. Abstraction of the underlying network resources to higher-layer applications and customers [RFC7926].

. Virtualization of particular underlying resources, whose selection criterion is the allocation of those resources to a particular customer, application or service [ONF-ARCH].

. Network slicing of infrastructure to meet specific customers' service requirements.

. Creation of a virtualized environment allowing operators to view and control multi-domain networks as a single virtualized network.

. The presentation to customers of networks as a virtual network via open and programmable interfaces.

1.1. Terminology

The following terms are used in this document. Some of them are newly defined, some others reference existing definition:

. Network Slicing: In the context of ACTN, network slicing is a collection of resources that are used to establish logically dedicated virtual networks over TE networks. It allows a network provider to provide dedicated virtual networks for application/customer over a common network infrastructure. The logically dedicated resources are a part of the larger common network infrastructures that are shared among various network slice instances which are the end-to-end realization of network slicing, consisting of the combination of physically or logically dedicated resources.

. Node: A node is a vertex on the graph representation of a TE topology. In a physical network topology, a node corresponds to a physical network element (NE). In an abstract network topology, a node (sometimes called an abstract node) is a representation as a single vertex of one or more physical NEs and their connecting physical connections. The concept of a node represents the ability to connect from any access to the node (a link end) to any other access to that node, although "limited cross-connect capabilities" may also be defined to restrict this functionality. Just as network slicing and network abstraction may be applied recursively, so a node in a topology may be created by applying slicing or abstraction on the nodes in the underlying topology.

. Link: A link is an edge on the graph representation of a TE topology. Two nodes connected by a link are said to be "adjacent" in the TE topology. In a physical network topology, a link corresponds to a physical connection. In an abstract network topology, a link (sometimes called an abstract link) is a representation of the potential to connect a pair of points with certain TE parameters (see RFC 7926 for details). Network slicing/virtualization and network abstraction may be applied recursively, so a link in a topology may be created by applying slicing and/or abstraction on the links in the underlying topology.

. CNC: A Customer Network Controller is responsible for communicating customer’s virtual network service requirements to network provider. It has knowledge of the end-point associated with virtual network service, service policy, and other QoS information related to the service it is responsible for instantiating.

. PNC: A Physical Network Controller is responsible for controlling devices or NEs under its direct control. The PNC functions can be implemented as part of an SDN domain controller, a Network Management System (NMS), an Element Management System (EMS), an active PCE-based controller or any other means to dynamically control a set of nodes and that is implementing an NBI compliant with ACTN specification.

. PNC domain: A PNC domain includes all the resources under the control of a single PNC. It can be composed of different routing domains and administrative domains, and the resources may come from different layers. The interconnection between PNC domains can be a link or a node.

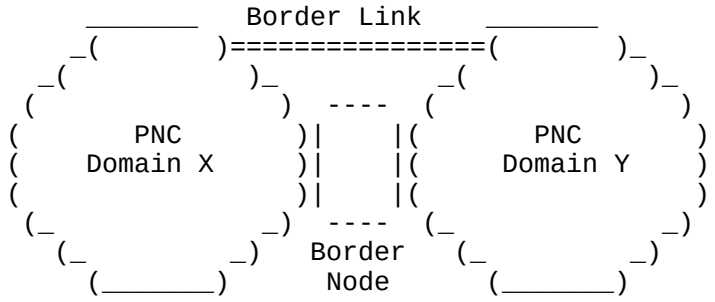


Figure 1: PNC Domain Borders

. MDSC: A multi-domain Service Coordinator is a functional block that implements all four ACTN main functions, i.e., multi domain coordination, virtualization/abstraction, customer mapping/translation, and virtual service coordination. The first two functions of the MDSC, namely, multi domain coordination and virtualization/abstraction are referred to as network-related functions while the last two functions, namely, customer mapping/translation and virtual service coordination are referred to as service-related functions. See details on these functions in Section 4.2. In some implementation, PNC and MDSC functions can be co-located and implemented in the same box.

. A Virtual Network (VN) is a customer view of the TE network. Depending on the agreement between client and provider various VN operations and VN views are possible as follows:

  o VN Creation - VN could be pre-configured and created via offline negotiation between customer and provider. In other cases, the VN could also be created dynamically based on a request from the customer with given SLA attributes which satisfy the customer’s objectives.

  o Dynamic Operations - The VN could be further modified or deleted based on a customer request. The customer can further act upon the virtual network resources to perform end-to-end tunnel management (set-up/release/modify).

These changes will result in subsequent LSP management at the operator's level.

o VN Type:

a. The VN can be seen as set of end-to-end tunnels from a customer point of view, where each tunnel is referred as a VN member. Each VN member can then be formed by recursive slicing or abstraction of paths in underlying networks. Such end-to-end tunnels may comprise of customer end points, access links, intra-domain paths, and inter-domain links. In this view, VN is thus a set of VN members (which is referred to as Type 1 VN)

b. The VN can also be seen as a topology comprising of physical, sliced, and abstract nodes and links. This VN is referred to as Type 2 VN. The nodes in this case include physical customer end points, border nodes, and internal nodes as well as abstracted nodes. Similarly the links include physical access links, inter-domain links, and intra-domain links as well as abstract links. With VN type 2, it is still possible to view VN member-level.

. Virtual Network Service (VNS) is requested by the customer and negotiated with the provider. There are three types of VNS defined in this document. Type 1 VNS refers to VNS in which customer is allowed to create and operate a Type 1 VN. Type 2a and 2b VNS refers to the VNS in which customer is allowed to create and operates a Type 2 VN. With Type 2a VNS, once the VN is statically created at service configuration time, the customer is not allowed to change the topology (i.e., adding or deleting abstract nodes/links). Type 2b VNS is the same as Type 2a VNS except that the customer is allowed to change topology dynamically from the initial topology created at service configuration time. See Section 3 for details.

. Abstraction. This process is defined in [RFC7926].

. Abstract Link: The term "abstract link" is defined in [RFC7926].

. Abstract Topology: The topology of abstract nodes and abstract links presented through the process of abstraction by a lower layer network for use by a higher layer network.

. Access link: A link between a customer node and a provider node.

. Inter-domain link: A link between domains managed by different PNCs. The MDSC is in charge of managing inter-domain links.

. Access Point (AP): An access point is used to keep confidentiality between the customer and the provider. It is a logical identifier shared between the customer and the provider, used to map the end points of the border node in both the customer and the provider NW. The AP can be used by the customer when requesting VN service to the provider.

. VN Access Point (VNAP): A VNAP is defined as the binding between an AP and a given VN and is used to identify the portion of the access and/or inter-domain link dedicated to a given VN.

2. Business Model of ACTN

The Virtual Private Network (VPN) [RFC4026] and Overlay Network (ON) models [RFC4208] are built on the premise that the network provider provides all virtual private or overlay networks to its customers. These models are simple to operate but have some disadvantages in accommodating the increasing need for flexible and dynamic network virtualization capabilities.

There are three key entities in the ACTN model:

- Customers
- Service Providers
- Network Providers

These are described in the following sections.

2.1. Customers

Within the ACTN framework, different types of customers may be taken into account depending on the type of their resource needs, and on

their number and type of access. For 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 for a service provider 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 a proxy and the services generally have very few dedicated resources (such as for subscriber drop), with everything else shared on the basis of some Service Level Agreement (LSA), 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 demands varying significantly in time and from customer to customer. This is one of the reasons why a bundled service offering is not enough and it is desirable to provide each advanced customer with a customized virtual network service.

Advanced customers may own dedicated virtual resources, or share resources. They may also have the ability to modify their service parameters within the scope of their virtualized environments. The primary focus of ACTN is Advanced Customers.

As customers are geographically spread over multiple network provider domains, they have to interface to multiple providers and may have to support multiple virtual network services with different underlying objectives set by the network providers. To enable these customers to support flexible and dynamic applications they need to control their allocated virtual network resources in a dynamic fashion, and that means that they need a view of the topology that spans all of the network providers. Customers of a given service provider can in turn offer a service to other customers in a recursive way.

2.2. Service Providers

Service providers are the providers of virtual network services (see Section 3 for details) to their customers. Service providers may or may not own physical network resources (i.e, may or may not be network providers as described in Section 2.3). When a service provider is the same as the network provider, this is similar to existing VPN models applied to a single provider. This approach

works well when the customer maintains a single interface with a single provider. When customer spans 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 distinct service providers interface to the customers. In this case, service providers are, themselves 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.

The ACTN network model is predicated upon this three tier model and is summarized in Figure 2:

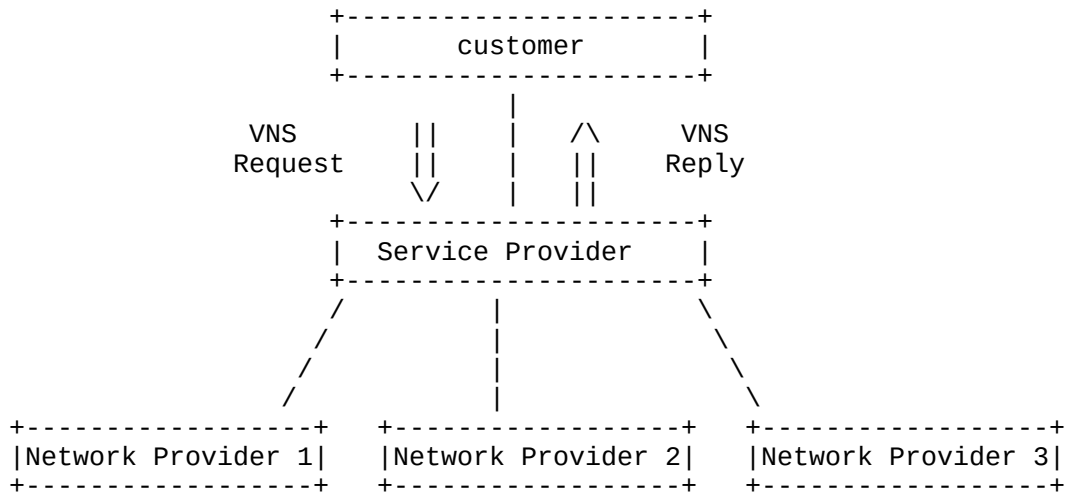


Figure 2: Three tier model.

There can be multiple service providers to which a customer may interface.

There are multiple types of service providers, for example:

- Data Center providers can be viewed as a service provider type as they own and operate data center resources for various WAN customers, and they can lease physical network resources from network providers.
- Internet Service Providers (ISP) are service providers 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.

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 described in this architecture separates the concerns of network providers and customers, with service providers acting as aggregators of customer requests.

3. Virtual Network Service

Virtual Network Service (VNS) is requested by the customer and negotiated with the provider. There are three types of VNS defined in this document.

Type 1 VNS refers to VNS in which customer is allowed to create and operate a Type 1 VN. Type 1 VN is a VN that comprises a set of end-to-end tunnels from a customer point of view, where each tunnel is referred as a VN member. With Type 1 VNS, the network operator does not need to provide additional abstract VN topology associated with the Type 1 VN.

Type 2a VNS refer to VNS in which customer is allowed to create and operates a Type 2 VN, but not allowed to change topology once it is configured at service configuration time. Type 2 VN is an abstract VN topology that may comprise of virtual/abstract nodes and links. The nodes in this case may include physical customer end points, border nodes, and internal nodes as well as abstracted nodes. Similarly, the links may include physical access links, inter-domain links, and intra-domain links as well as abstract links.

Type 2b VNS refers to VNS in which customer is allowed to create and operate a Type 2 VN and the customer is allowed to dynamically change abstract VN topology from the initially configured abstract VN topology at service configuration time.

From an implementation standpoint, Type 2a VNS and Type 2b VNS differentiation might be fulfilled via local policy.

In all types of VNS, customer can specify a set of service related parameters such as connectivity type, VN traffic matrix (e.g., bandwidth, latency, diversity, etc.), VN survivability, VN service policy and other characteristics.

4. ACTN Base Architecture

This section provides a high-level model of ACTN showing the interfaces and the flow of control between components.

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

- Multi-domain coordination function: This function oversees the specific aspects of the different domains and builds a single abstracted end-to-end network topology in order to coordinate end-to-end path computation and path/service provisioning. Domain sequence path calculation/determination is also a part of this function.

- Virtualization/Abstraction function: This function provides an abstracted view of the underlying network resources for use by the customer - a customer may be the client or a higher level controller entity. This function includes network path computation based on customer service connectivity request constraints, path computation based on the global network-wide abstracted topology, and the creation of an abstracted view of network resources allocated to each customer. These operations depend on customer-specific network objective functions and customer traffic profiles.

- Customer mapping/translation function: This function is to map customer requests/commands into network provisioning requests that can be sent to the Physical Network Controller (PNC) according to business policies provisioned statically or dynamically at the OSS/NMS. Specifically, it provides mapping and translation of a customer’s service request into a set of parameters that are specific to a network type and technology such that network configuration process is made possible.

. Virtual service coordination function: This function translates customer service-related information into virtual network service operations in order to seamlessly operate virtual networks while meeting a customer's service requirements. In the context of ACTN, service/virtual service coordination includes a number of service orchestration functions such as multi-destination load balancing, guarantees of service quality, bandwidth and throughput. It also includes notifications for service fault and performance degradation and so forth.

Figure 3 depicts the base ACTN architecture with three controller types and the corresponding interfaces between these controllers. The types of controller defined in the ACTN architecture are shown in Figure 3 below and are as follows:

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

Figure 3 also shows the following interfaces:

. CMI - CNC-MDSC Interface
. MPI - MDSC-PNC Interface
. SBI - South Bound Interface

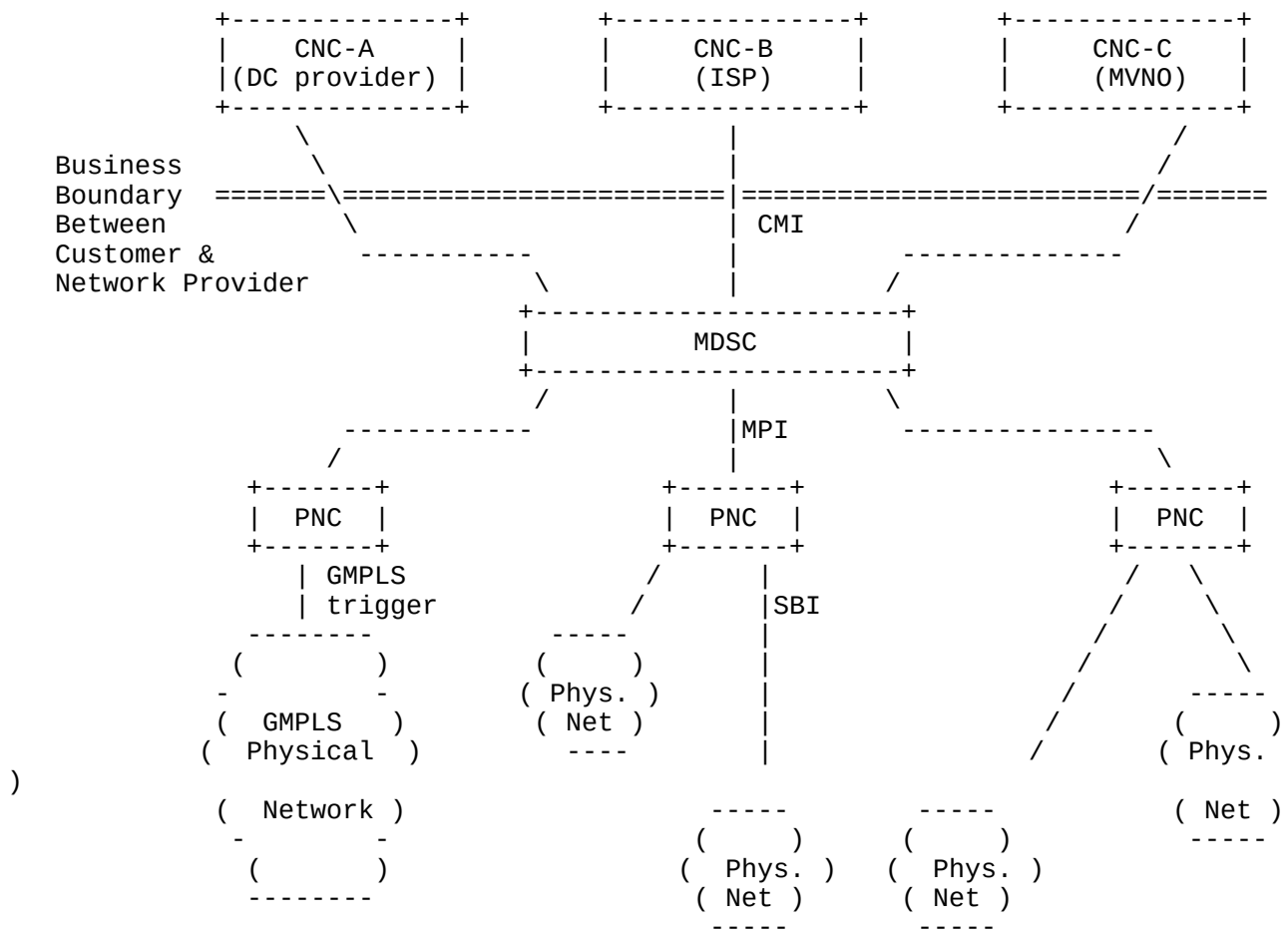


Figure 3: ACTN Base Architecture

4.1. Customer Network Controller

A Virtual Network Service is instantiated by the Customer Network Controller via the CNC-MDSC Interface (CMI). As the Customer Network Controller directly interfaces to the applications, it understands multiple application requirements and their service needs. It is assumed that the Customer Network Controller and the MDSC have a common knowledge of the end-point interfaces based on their business negotiations prior to service instantiation. End-point interfaces

refer to customer-network physical interfaces that connect customer premise equipment to network provider equipment.

4.2. Multi Domain Service Coordinator

The Multi Domain Service Coordinator (MDSC) sits between the CNC that issues connectivity requests and the Physical Network Controllers (PNCs) that manage the physical network resources. The MDSC can be collocated with the PNC.

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 [RFC7491] and the ONF SDN architecture [ONF-ARCH].

The MDSC is the only building block of the architecture that can implement all four ACTN main functions, i.e., multi domain coordination, virtualization/abstraction, customer mapping/translation, and virtual service coordination. The first two functions of the MDSC, namely, multi domain coordination and virtualization/abstraction are referred to as network-related functions while the last two functions, namely, customer mapping/translation and virtual service coordination are referred to as service-related functions.
The key point of the MDSC (and of the whole ACTN framework) is detaching the network and service control from underlying technology to help the customer express the network as desired by business needs. The MDSC envelopes the instantiation of the right technology and network control to meet business criteria. In essence it controls and manages the primitives to achieve functionalities as desired by the CNC.

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 also be possible to have an M:1 relationship between MDSCs and PNC to allow for network resource partitioning/sharing among different customers not necessarily connected to the same MDSC (e.g., different service providers).

4.3. Physical Network Controller

The Physical Network Controller (PNC) oversees configuring the network elements, monitoring the topology (physical or virtual) of the network, and passing information about the topology (either raw or abstracted) to the MDSC.

The internal architecture of the PNC, its 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 [RFC7491] 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 main ACTN main functions: multi domain coordination and virtualization/abstraction function.
Note that from an implementation point of view it is possible to integrate one or more MDSC functions and one or more PNC functions within the same controller.

4.4. ACTN Interfaces

The network has to provide open, programmable interfaces, through 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, the network control plane for transport networks has been separated from the data plane and as such it is not viable for the customer to directly interface with transport network elements.

. CMI Interface: The CNC-MDSC Interface (CMI) is an interface between a CNC and an MDSC. As depicted in Figure 3, the CMI is a business boundary between customer and network provider. It is used to request virtual network services required for the applications. Note that all service related information such as specific service properties, including virtual network service type, topology, bandwidth, and constraint information are conveyed over this interface. Most of the information over this interface is technology agnostic; however, there are some cases, e.g., access link configuration, where it should be

possible to explicitly request for a VN to be created at a given layer in the network (e.g. ODU VN or MPLS VN).

. MPI Interface: The MDSC-PNC Interface (MPI) is an interface between an MDSC and a PNC. It communicates the creation requests for new connectivity or for bandwidth changes in the physical network. In multi-domain environments, the MDSC needs to establish multiple MPIs, one for each PNC, as there is one PNC responsible for control of each domain. The MPI could have different degrees of abstraction and present an abstracted topology hiding technology specific aspects of the network or convey technology specific parameters to allow for path computation at the MDSC level. Please refer to CCAMP Transport NBI work for the latter case [Transport NBI].

. SBI Interface: This interface is out of the scope of ACTN. It is shown in Figure 3 for reference reason only.

Please note that for all the three interfaces, when technology specific information needs to be included, this info will be add-ons on top of the general abstract topology. As far as general topology abstraction standpoint, all interfaces are still recursive in nature.

5. Advanced ACTN architectures

This section describes advanced forms of ACTN architectures as possible implementation choices.

5.1. MDSC Hierarchy for scalability

A hierarchy of MDSCs can be foreseen for many reasons, among which are scalability, administrative choices or putting together different layers and technologies in the network. In the case where there is a hierarchy of MDSCs, we introduce the higher-level MDSC (MDSC-H) the lower-level MDSC (MDSC-L) and the interface between them is basically of a recursive nature of the MPI. An implementation choice could foresee the usage of an MDSC-L for all the PNCs related to a given network layer or technology (e.g. IP/MPLS) a different MDSC-L for the PNCs related to another layer/technology (e.g. OTN/WDM) and an MDSC-H to coordinate them.

Figure 4 shows this case.

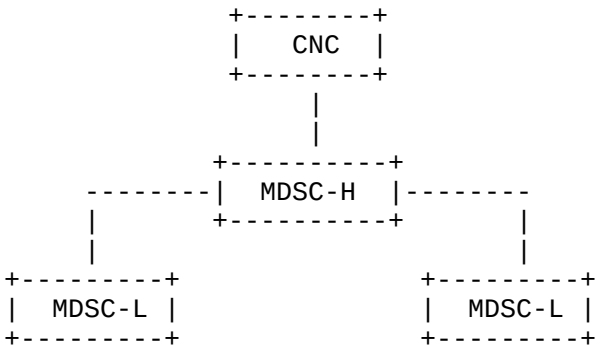


Figure 4: MDSC Hierarchy

Note that both the MDSC-H and the MDSC-L in general cases implement all four functions of the MDSC discussed in Section 3.2.

5.2. Functional Split of MDSC Functions in Orchestrators

Another implementation choice could foresee the separation of MDSC functions into two groups (i.e., one group for service-related functions and another group for network-related functions) which will result in a service orchestrator for providing service-related functions of MDSC and other non-ACTN functions and a network orchestrator for providing network-related functions of MDSC and other non-ACTN functions. Figure 5 shows this case and it also depicts the mapping between ACTN architecture and the YANG service model architecture described in [Service-YANG]. This mapping is helpful for the readers who are not familiar with some TEAS specific terminology used in this document. A number of key ACTN interfaces exist for deployment and operation of ACTN-based networks. These are highlighted in Figure 5 (ACTN Interfaces).

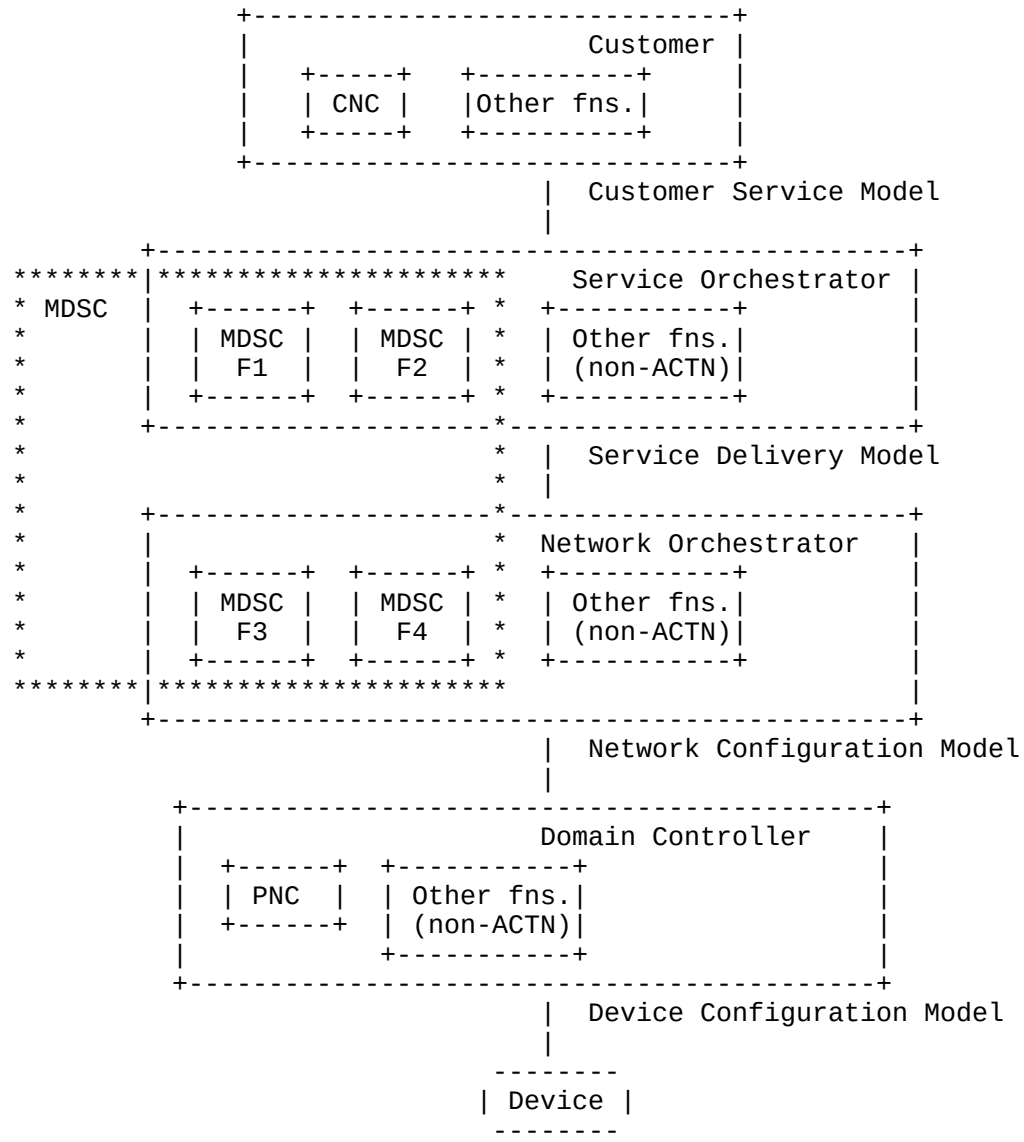


Figure 5: ACTN Architecture in the context of YANG Service Models

In Figure 5, MDSC F1 and F2 correspond to customer mapping/translation, and virtual service coordination, respectively, which are the MDSC service-related functions as defined in Section 4. MDSC F3 and F4 correspond to multi domain coordination, virtualization/abstraction, respectively, which are the MDSC network-related functions as defined in Section 4. In some implementation, MDSC F1 and F2 can be implemented as part of a Service Orchestrator which may support other non-ACTN functions. Likewise, the MDSC F3 and F4 can be implemented as part of a Network Orchestrator which may support other non-ACTN MDSC functions.

Also note that the PNC is not same as domain controller. Domain controller in general has a larger set of functions than that of PNC. The main functions of PNC are explained in Section 3.3. Likewise, Customer has a larger set of functions than that of the CNC.

Customer service model describes a service as offer or delivered to a customer by a network operator as defined in [Service-YANG]. The CMI is a subset of a customer service model to support VNS. This model encompasses other non-TE/non-ACTN models to control non-ACTN services (e.g., L3SM).

Service delivery model is used by a network operator to define and configure how a service is provided by the network as defined in [Service-YANG]. This model is similar to the MPI model as the network-related functions of the MDSC, i.e., F3 and F4, provide an abstract topology view of the E2E network to the service-related functions of the MDSC, i.e., F1 and F2, which translate customer’s request at the CMI into the network configuration at the MPI.

Network configuration model is used by a network orchestrator to provide network-level configuration model to a controller as defined in [Service-YANG]. The MPI is a subset of network configuration model to support TE configuration. This model encompasses the MPI model plus other non-TE/non-ACTN models to control non-ACTN functions of the domain controller (e.g., L3VPN).

Device configuration model is used by a controller to configure physical network elements.

6. Topology Abstraction Method

This section discusses topology abstraction types and their context in ACTN architecture. Topology abstraction is useful in ACTN architecture as a way to scale multi-domain network operation. As

the MDSC needs to coordinate with the PNCs in multi-domain networks for determining a feasible domain sequence and provisioning the end-to-end tunnel based on the determined domain sequence, topology abstraction provides an efficient network scaling mechanism. Note that this is the abstraction performed by the PNC to the MDSC or by the MDSC-L to the MDSC-H, and that this is different from the VN Type 2 topology (that is created and negotiated between the CNC and the MDSC as part of the VNS). The purpose of topology abstraction discussed in this section is for an efficient internal network operation based on abstraction principle.

Refer to [ACTN-Abstraction] for detailed discussions on factors that affect topology abstraction, how to build topology abstraction, various considerations and which method to pick based on abstraction types and the nature of underlying transport technology (e.g., MPLS, OTN, WSON, etc.).

6.1. No abstraction (native/white topology)

This is a case where the PNC provides the actual network topology to the MDSC without any hiding or filtering of information as shown in Figure 6. In this case, the MDSC has the full knowledge of the underlying network topology and as such there is no need for the MDSC to send a path computation request to the PNC. The computation burden will fall on the MDSC to find an optimal end-to-end path and optimal per domain paths.

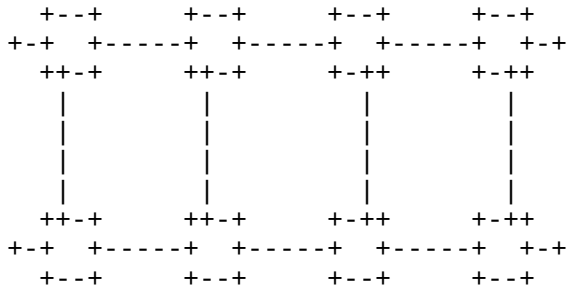


Figure 6a: The native/white topology

6.2. One Virtual Node (black topology)

The entire domain network is abstracted as a single virtual node (see the definition of virtual node in [RFC7926]) with the access/egress links without disclosing any node internal connectivity information.

Figure 6b depicts a native topology with the corresponding black topology with one virtual node and inter-domain links. In this case, the MDSC has to make path computation requests to the PNCs before it can determine an end-to-end path. If there are a large number of inter-connected domains, this abstraction method may impose a heavy coordination load at the MDSC level in order to find an optimal end-to-end path.

The black topology would not give the MDSC any critical network resource information other than the border nodes/links information and as such it is likely to have a need for complementary communications between the MDSC and the PNCs (e.g., Path computation Request/Reply).

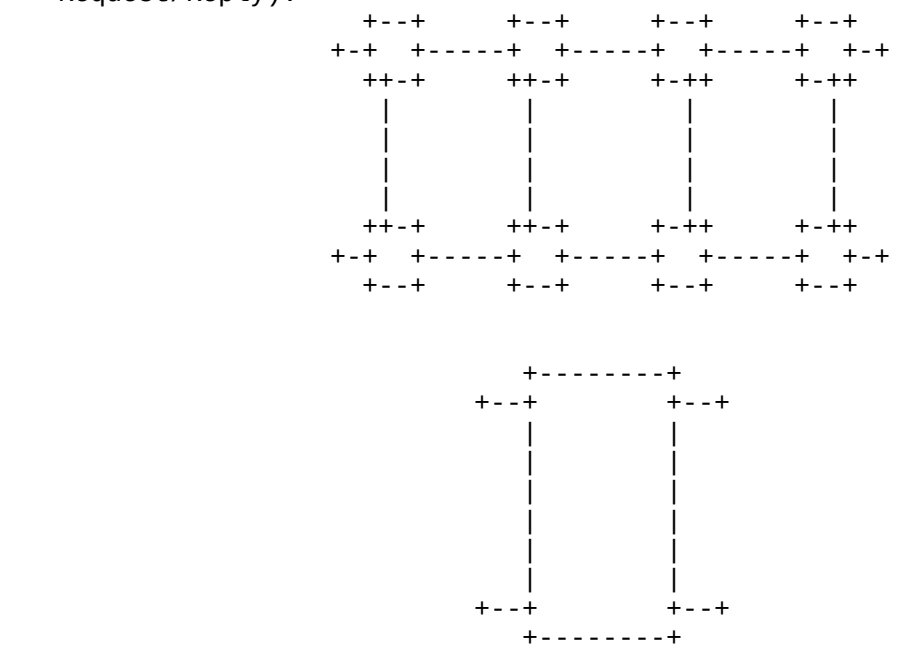


Figure 6b: The native topology and the corresponding black topology with one virtual node and inter-domain links

6.3. Abstraction of TE tunnels for all pairs of border nodes (grey topology)

This abstraction level, referred to a grey topology is between black topology and white topology from a granularity point of view. As shown in Figures 7a and 7b, we may further differentiate from a perspective of how to abstract internal TE resources between the pairs of border nodes:

- Grey topology type A: border nodes with a TE links between them in a full mesh fashion (See Figure 7a)
- Grey topology type B: border nodes with some internal abstracted nodes and abstracted links (See Figure 7b)

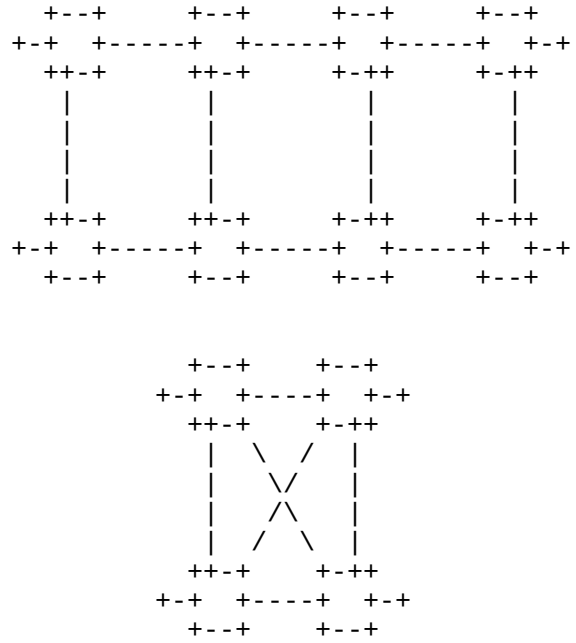
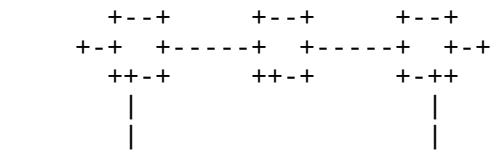


Figure 7a: The native topology and the corresponding grey topology type A with TE links between border nodes



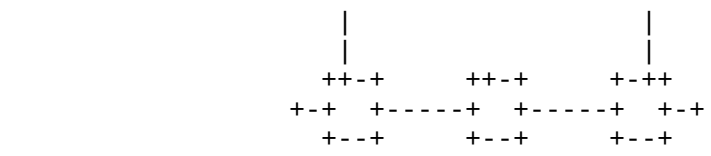


Figure 7b: The grey topology type B with abstract nodes/links between border nodes

6.3.1. Grey topology type A: border nodes with a TE links between them in a full mesh fashion

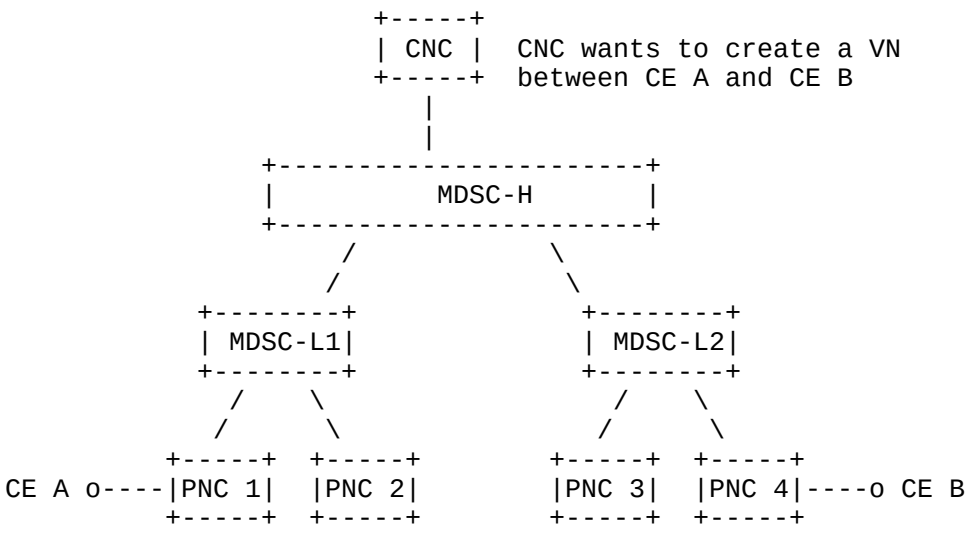
For each pair of ingress and egress nodes (i.e., border nodes to/from the domain), TE link metric is provided with TE attributes such as max bandwidth available, link delay, etc. This abstraction depends on the underlying TE networks.
Note that this grey topology can also be represented as a single abstract node with the connectivity matrix defined in [TE-Topology], abstracting the internal connectivity information. The only thing might be different is some additional information about the end points of the links of the border nodes (i.e., links outward customer-facing) as they cannot be included in the connectivity matrix's termination points.

6.3.2. Grey topology Type B

The grey abstraction type B would allow the MDSC to have more information about the internals of the domain networks by the PNCs so that the MDSC can flexibly determine optimal paths. The MDSC may configure some of the internal virtual nodes (e.g., cross-connect) to redirect its traffic as it sees changes from the domain networks.

6.4. Topology Abstraction Granularity Level example

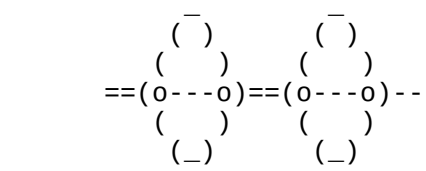
This section illustrates how topology abstraction operates in different level of granularity over a hierarchy of MDSCs which is shown in Figure 8 below.



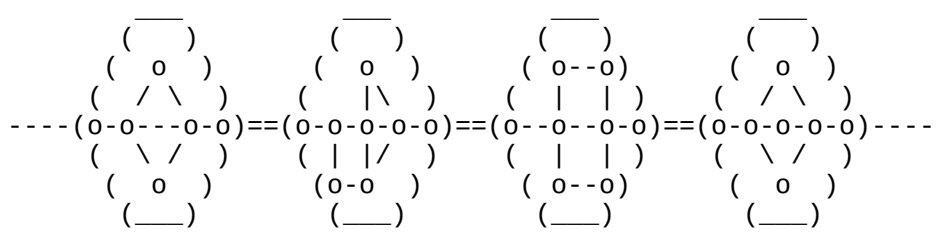
Topology operated by MDSC-H

Topology operated by MDSC-L1

Topology operated by MDSC-L2



Actual Topology



Where o is a node and -- is a link and === a border link

Figure 8: Illustration of topology abstraction granularity levels

In the example depicted in Figure 8, there are four domains under control of the respective PNCs, namely, PNC 1, PNC 2, PNC3 and PNC4. Assume that MDSC L-1 is controlling PNC 1 and PNC 2 while MDSC L-2 is controlling PNC 3 and PNC 4. Let us assume that each of the PNCs provides a grey topology abstraction in which to present only border nodes and links within and outside the domain. The abstract topology MDSC-L1 would operate is basically a combination of the two topologies the PNCs (PNC 1 and PNC 2) provide. Likewise, the abstract topology MDSC-L2 would operate is shown in Figure 8. Both MDSC-L1 and MDSC-L2 provide a black topology abstraction in which each PNC domain is presented as one virtual node to its top level MDSC-H. Then the MDSC-H combines these two topologies updated by MDSC-L1 and MDSC-L2 to create the abstraction topology to which it operates. MDSC-H sees the whole four domain networks as four virtual nodes connected via virtual links. This illustrates the point discussed in Section 5.1: The top level MDSC may operate on a higher level of abstraction (i.e., less granular level) than the lower level MSDCs.

7. Access Points and Virtual Network Access Points

In order not to share unwanted topological information between the customer domain and provider domain, a new entity is defined which is referred to as the Access Point (AP). See the definition of AP in Section 1.1.

A customer node will use APs as the end points for the request of VNS as shown in Figure 9.

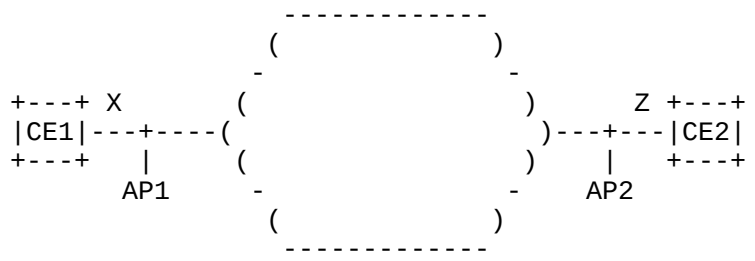


Figure 9: APs definition customer view

Let’s take as an example a scenario shown in Figure 7. CE1 is connected to the network via a 10Gb link and CE2 via a 40Gb link. Before the creation of any VN between AP1 and AP2 the customer view can be summarized as shown in Table 1:

	End Point	Access Link Bandwidth	
AP id	CE,port	MaxResBw	AvailableBw
AP1	CE1,portX	10Gb	10Gb
AP2	CE2,portZ	40Gb	40Gb

Table 1: AP - customer view

On the other hand, what the provider sees is shown in Figure 10.

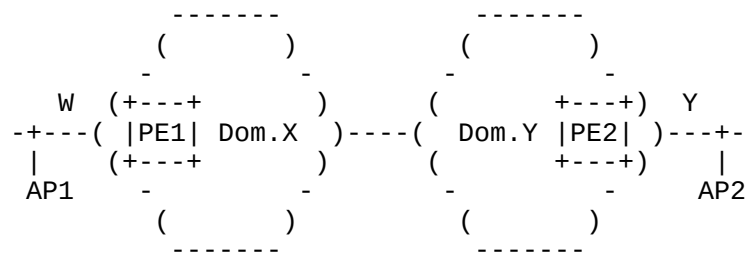


Figure 10: Provider view of the AP

Which results in a summarization as shown in Table 2.

	End Point	Access Link Bandwidth	
AP id	PE,port	MaxResBw	AvailableBw
AP1	PE1,portW	10Gb	10Gb
AP2	PE2,portY	40Gb	40Gb

Table 2: AP - provider view

A Virtual Network Access Point (VNAP) needs to be defined as binding between the AP that is linked to a VN and that is used to allow for different VNs to start from the same AP. It also allows for traffic

engineering on the access and/or inter-domain links (e.g., keeping track of bandwidth allocation). A different VNAP is created on an AP for each VN.

In the simple scenario depicted above we suppose we want to create two virtual networks. The first with VN identifier 9 between AP1 and AP2 with bandwidth of 1Gbps, while the second with VN id 5, again between AP1 and AP2 and with bandwidth 2Gbps.

The provider view would evolve as shown in Table 3.

	End Point	Access Link/VNAP Bw		
AP/VNAPid	PE,port	MaxResBw	AvailableBw	
AP1	PE1,portW	10Gbps	7Gbps	
-VNAP1.9		1Gbps	N.A.	
-VNAP1.5		2Gbps	N.A	
AP2	PE2,portY	40Gbps	37Gbps	
-VNAP2.9		1Gbps	N.A.	
-VNAP2.5		2Gbps	N.A	

Table 3: AP and VNAP - provider view after VNS creation

7.1. Dual homing scenario

Often there is a dual homing relationship between a CE and a pair of PEs. This case needs to be supported by the definition of VN, APs and VNAPs. Suppose CE1 connected to two different PEs in the operator domain via AP1 and AP2 and that the customer needs 5Gbps of bandwidth between CE1 and CE2. This is shown in Figure 11.

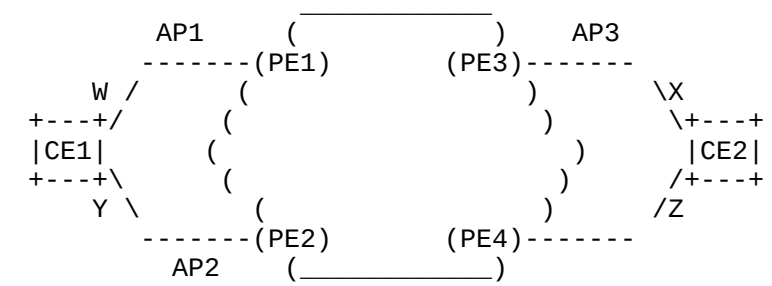


Figure 11: Dual homing scenario

In this case, the customer will request for a VN between AP1, AP2 and AP3 specifying a dual homing relationship between AP1 and AP2. As a consequence no traffic will flow between AP1 and AP2. The dual homing relationship would then be mapped against the VNAPs (since other independent VNs might have AP1 and AP2 as end points).

The customer view would be shown in Table 4.

	End Point	Access Link/VNAP Bw			
AP/VNAPid	CE,port	MaxResBw	AvailableBw		Dual Homing
AP1 -VNAP1.9	CE1,portW 	10Gbps 5Gbps	5Gbps N.A.		VNAP2.9
AP2 -VNAP2.9	CE1,portY 	40Gbps 5Gbps	35Gbps N.A.		VNAP1.9
AP3 -VNAP3.9	CE2,portX 	40Gbps 5Gbps	35Gbps N.A.		NONE

Table 4: Dual homing - customer view after VN creation

8. Advanced ACTN Application: Multi-Destination Service

A further advanced application of ACTN is in the case of Data Center selection, where the customer requires the Data Center selection to be based on the network status; this is referred to as Multi-Destination in [ACTN-REQ]. In terms of ACTN, a CNC could request a connectivity service (virtual network) between a set of source Aps and destination APs and leave it up to the network (MDSC) to decide which source and destination access points to be used to set up the connectivity service (virtual network). The candidate list of source and destination APs is decided by a CNC (or an entity outside of ACTN) based on certain factors which are outside the scope of ACTN.

Based on the AP selection as determined and returned by the network (MDSC), the CNC (or an entity outside of ACTN) should further take

care of any subsequent actions such as orchestration or service setup requirements. These further actions are outside the scope of ACTN.

Consider a case as shown in Figure 12, where three data centers are available, but the customer requires the data center selection to be based on the network status and the connectivity service setup between the AP1 (CE1) and one of the destination APs (AP2 (DC-A), AP3 (DC-B), and AP4 (DC-C)). The MDSC (in coordination with PNCs) would select the best destination AP based on the constraints, optimization criteria, policies, etc., and setup the connectivity service (virtual network).

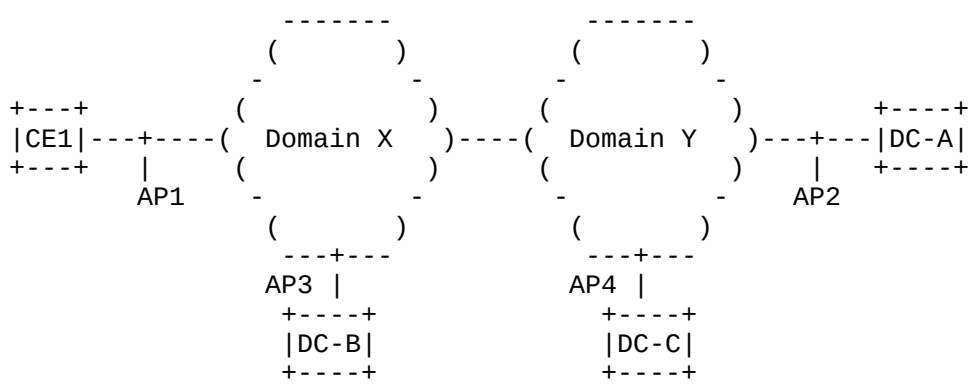


Figure 12: End point selection based on network status

8.1. Pre-Planned End Point Migration

Further in case of Data Center selection, customer could request for a backup DC to be selected, such that in case of failure, another DC site could provide hot stand-by protection. As shown in Figure 13 DC-C is selected as a backup for DC-A. Thus, the VN should be setup by the MDSC to include primary connectivity between AP1 (CE1) and AP2 (DC-A) as well as protection connectivity between AP1 (CE1) and AP4 (DC-C).

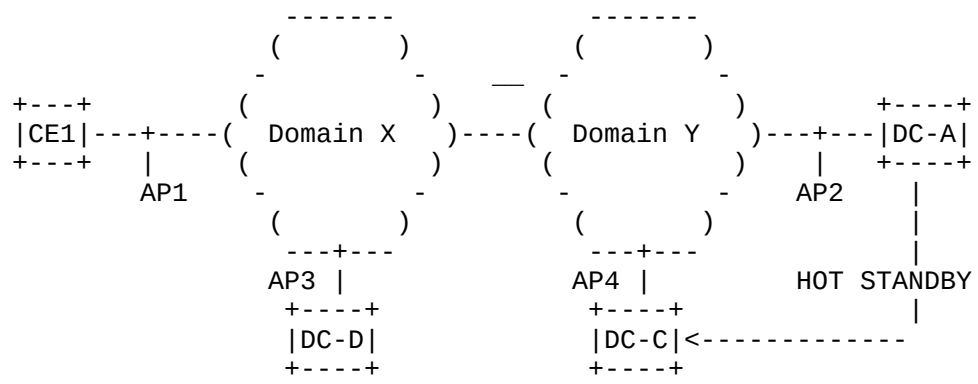


Figure 13: Pre-planned end point migration

8.2. On the Fly End Point Migration

Compared to pre-planned end point migration, on the fly end point selection is dynamic in that the migration is not pre-planned but decided based on network condition. Under this scenario, the MDSC would monitor the network (based on the VN SLA) and notify the CNC in case where some other destination AP would be a better choice based on the network parameters. The CNC should instruct the MDSC when it is suitable to update the VN with the new AP if it is required.

9. Advanced Topic

This section describes how ACTN architecture supports some deployment scenarios. See Appendix A for details on MDSC and PNC functions integrated in Service/Network Orchestrator and Appendix B for IP + Optical with L3VPN service.

10. Manageability Considerations

The objective of ACTN is to manage traffic engineered resources, and provide a set of mechanism to allow clients to request virtual connectivity across server network resources. ACTN will support multiple clients each with its own view of and control of the server network, the network operator will need to partition (or "slice") their network resources, and manage them resources accordingly.

The ACTN platform will, itself, need to support the request, response, and reservations of client and network layer connectivity. It will also need to provide performance monitoring and control of traffic engineered resources. The management requirements may be categorized as follows:

- Management of external ACTN protocols
- Management of internal ACTN protocols
- Management and monitoring of ACTN components
- Configuration of policy to be applied across the ACTN system

10.1. Policy

It is expected that a policy will be an important aspect of ACTN control and management. Typically, policies are used via the components and interfaces, during deployment of the service, to ensure that the service is compliant with agreed policy factors (often described in Service Level Agreements - SLAs), these include, but are not limited to: connectivity, bandwidth, geographical transit, technology selection, security, resilience, and economic cost.

Depending on the deployment the ACTN deployment architecture, some policies may have local or global significance. That is, certain policies may be ACTN component specific in scope, while others may have broader scope and interact with multiple ACTN components. Two examples are provided below:

- A local policy might limit the number, type, size, and scheduling of virtual network services a customer may request via its CNC. This type of policy would be implemented locally on the MDSC.

- A global policy might constrain certain customer types (or specific customer applications) to only use certain MDSCs, and be restricted to physical network types managed by the PNCs. A global policy agent would govern these types of policies.

This objective of this section is to discuss the applicability of ACTN policy: requirements, components, interfaces, and examples. This section provides an analysis and does not mandate a specific method for enforcing policy, or the type of policy agent that would be responsible for propagating policies across the ACTN components. It does highlight examples of how policy may be applied in the

context of ACTN, but it is expected further discussion in an applicability or solution specific document, will be required.

10.2. Policy applied to the Customer Network Controller

A virtual network service for a customer application will be requested from the CNC. It will reflect the application requirements and specific service policy needs, including bandwidth, traffic type and survivability. Furthermore, application access and type of virtual network service requested by the CNC, will be need adhere to specific access control policies.

10.3. Policy applied to the Multi Domain Service Coordinator

A key objective of the MDSC is to help the customer express the application connectivity request via its CNC as set of desired business needs, therefore policy will play an important role.

Once authorised, the virtual network service will be instantiated via the CNC-MDSC Interface (CMI), it will reflect the customer application and connectivity requirements, and specific service transport needs. The CNC and the MDSC components will have agreed connectivity end-points, use of these end-points should be defined as a policy expression when setting up or augmenting virtual network services. Ensuring that permissible end-points are defined for CNCs and applications will require the MDSC to maintain a registry of permissible connection points for CNCs and application types.

It may also be necessary for the MDSC to resolve policy conflicts, or at least flag any issues to administrator of the MDSC itself. Conflicts may occur when virtual network service optimization criterion are in competition. For example, to meet objectives for service reachability a request may require an interconnection point between multiple physical networks; however, this might break a confidentially policy requirement of specific type of end-to-end service. This type of situation may be resolved using hard and soft policy constraints.

10.4. Policy applied to the Physical Network Controller

The PNC is responsible for configuring the network elements, monitoring physical network resources, and exposing connectivity (direct or abstracted) to the MDSC. It is therefore expected that policy will dictate what connectivity information will be exported between the PNC, via the MDSC-PNC Interface (MPI), and MDSC.

Policy interactions may arise when a PNC determines that it cannot compute a requested path from the MDSC, or notices that (per a locally configured policy) the network is low on resources (for example, the capacity on key links become exhausted). In either case, the PNC will be required to notify the MDSC, which may (again per policy) act to construct a virtual network service across another physical network topology.

Furthermore, additional forms of policy-based resource management will be required to provide virtual network service performance, security and resilience guarantees. This will likely be implemented via a local policy agent and subsequent protocol methods.

11. Security Considerations

The ACTN framework described in this document defines key components and interfaces for managed traffic engineered networks. Securing the request and control of resources, confidentially of the information, and availability of function, should all be critical security considerations when deploying and operating ACTN platforms.

Several distributed ACTN functional components are required, and as a rule implementations should consider encrypting data that flow between components, especially when they are implemented at remote nodes, regardless if these are external or internal network interfaces.

The ACTN security discussion is further split into two specific categories described in the following sub-sections:

- Interface between the Customer Network Controller and Multi Domain Service Coordinator (MDSC), CNC-MDSC Interface (CMI)

- Interface between the Multi Domain Service Coordinator and Physical Network Controller (PNC), MDSC-PNC Interface (MPI)

From a security and reliability perspective, ACTN may encounter many risks such as malicious attack and rogue elements attempting to connect to various ACTN components. Furthermore, some ACTN components represent a single point of failure and threat vector, and must also manage policy conflicts, and eavesdropping of communication between different ACTN components.

The conclusion is that all protocols used to realize the ACTN framework should have rich security features, and customer, application and network data should be stored in encrypted data

stores. Additional security risks may still exist. Therefore, discussion and applicability of specific security functions and protocols will be better described in documents that are use case and environment specific.

11.1. Interface between the Customer Network Controller and Multi Domain Service Coordinator (MDSC), CNC-MDSC Interface (CMI)

The role of the MDSC is to detach the network and service control from underlying technology to help the customer express the network as desired by business needs. It should be noted that data stored by the MDSC will reveal details of the virtual network services, and which CNC and application is consuming the resource. The data stored must therefore be considered as a candidate for encryption.

CNC Access rights to an MDSC must be managed. MDSC resources must be properly allocated, and methods to prevent policy conflicts, resource wastage and denial of service attacks on the MDSC by rogue CNCs, should also be considered.

A CNC-MDSC protocol interface will likely be an external protocol interface. Again, suitable authentication and authorization of each CNC connecting to the MDSC will be required, especially, as these are likely to be implemented by different organizations and on separate functional nodes. Use of the AAA-based mechanisms would also provide role-based authorization methods, so that only authorized CNC’s may access the different functions of the MDSC.

11.2. Interface between the Multi Domain Service Coordinator and Physical Network Controller (PNC), MDSC-PNC Interface (MPI)

The function of the Physical Network Controller (PNC) is to configure network elements, provide performance and monitoring functions of the physical elements, and export physical topology (full, partial, or abstracted) to the MDSC.

Where the MDSC must interact with multiple (distributed) PNCs, a PKI-based mechanism is suggested, such as building a TLS or HTTPS connection between the MDSC and PNCs, to ensure trust between the physical network layer control components and the MDSC.

Which MDSC the PNC exports topology information to, and the level of detail (full or abstracted) should also be authenticated and specific access restrictions and topology views, should be configurable and/or policy-based.

12. References

12.1. Informative References

[RFC2702] Awduche, D., et. al., "Requirements for Traffic Engineering Over MPLS", RFC 2702, September 1999.

[RFC4026] L. Andersson, T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, March 2005.

[RFC4208] G. Swallow, J. Drake, H.Ishimatsu, 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.

[RFC4655] Farrel, A., Vasseur, J.-P., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", IETF RFC 4655, August 2006.

[RFC5654] Niven-Jenkins, B. (Ed.), D. Brungard (Ed.), and M. Betts (Ed.), "Requirements of an MPLS Transport Profile", RFC 5654, September 2009.

[RFC7149] Boucadair, M. and Jacquenet, C., "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, March 2014.

[RFC7926] A. Farrel (Ed.), "Problem Statement and Architecture for Information Exchange between Interconnected Traffic-Engineered Networks", RFC 7926, July 2016.

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

[ONF-ARCH] Open Networking Foundation, "SDN architecture", Issue 1.1, ONF TR-521, June 2016.

[RFC7491] King, D., and Farrel, A., "A PCE-based Architecture for Application-based Network Operations", RFC 7491, March 2015.

[Transport NBI] Busi, I., et al., "Transport North Bound Interface Use Cases", draft-tnbidt-ccamp-transport-nbi-use-cases, work in progress.

[ACTN-Abstraction] Y. Lee, et al., "Abstraction and Control of TE Networks (ACTN) Abstraction Methods", draft-lee-teas-actn-abstraction, work in progress.

13. Contributors

Adrian Farrel
Old Dog Consulting
Email: adrian@olddog.co.uk

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

Khuzema Pithewan
Infinera
Email: kpithewan@infinera.com

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

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
Microsoft
Email: luyuanf@gmail.com

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

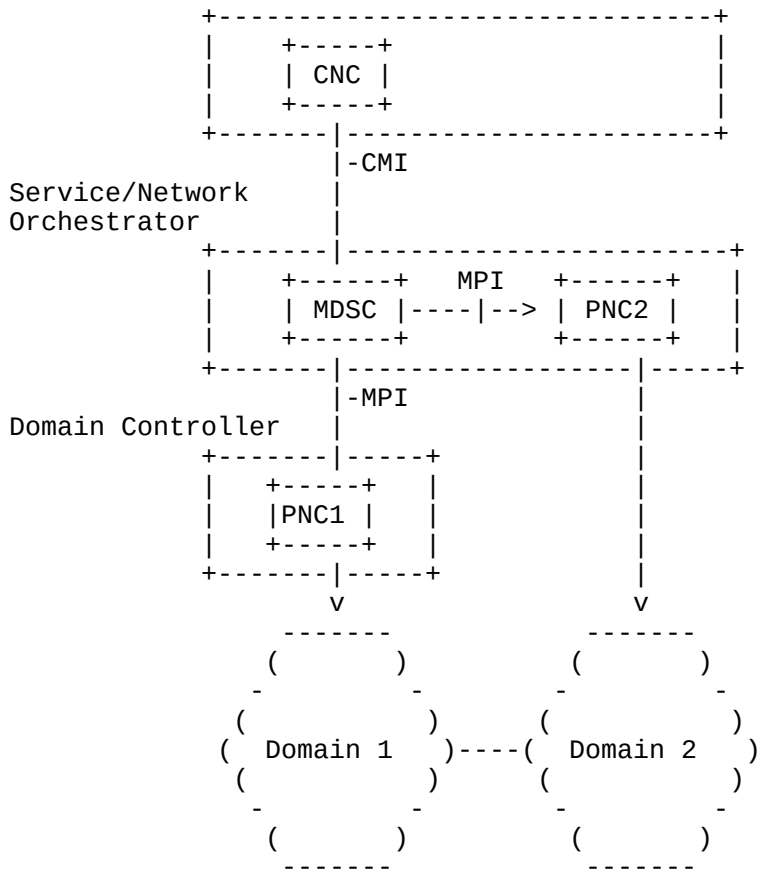
Sergio Belotti
Alcatel Lucent
Via Trento, 30
Vimercate, Italy
Email: sergio.belotti@nokia.com
Daniel King
Lancaster University
Email: d.king@lancaster.ac.uk

Dhruv Dhody
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India
Email: dhruv.ietf@gmail.com

Gert Grammel
Juniper Networks
Email: ggrammel@juniper.net

APPENDIX A - Example of MDSC and PNC functions integrated in Service/Network Orchestrator

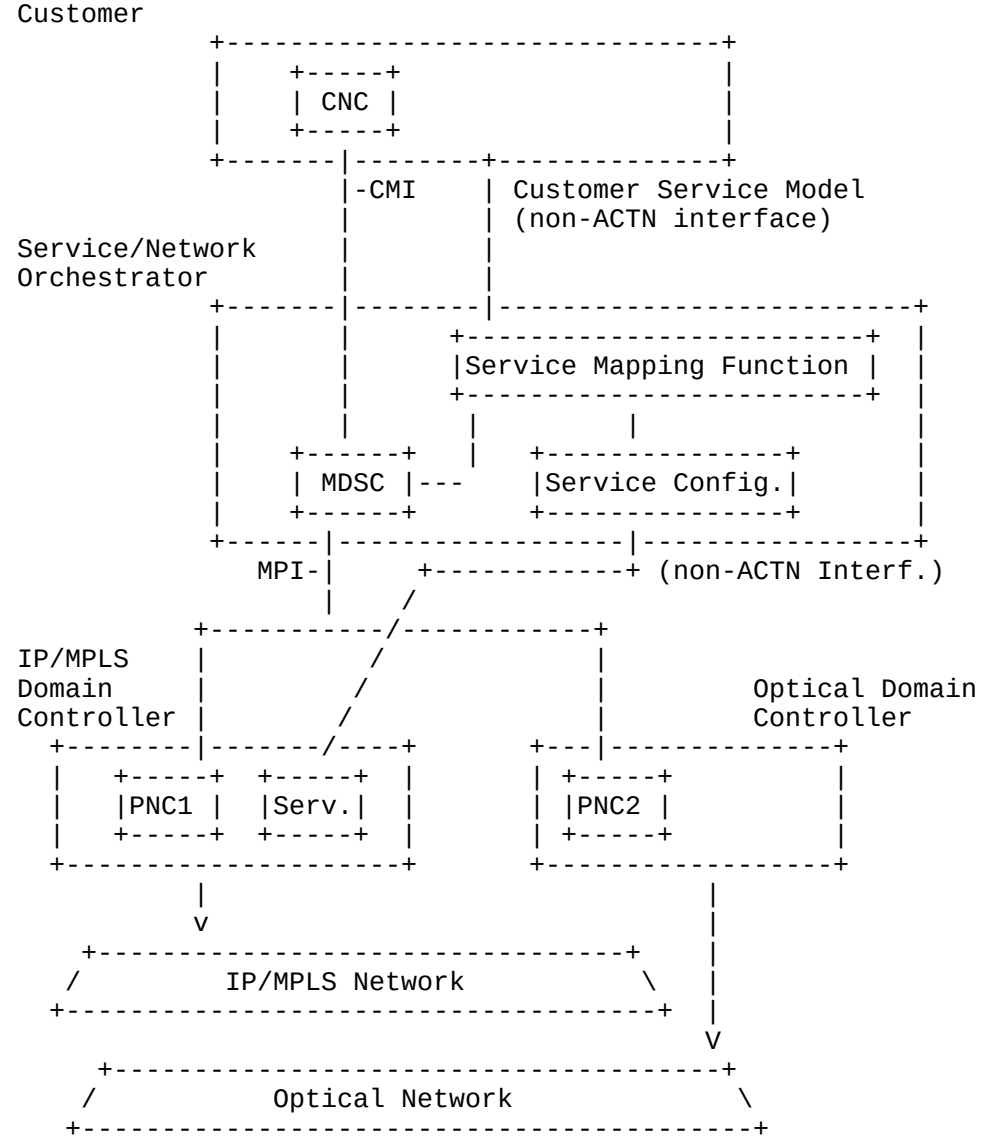
This section provides an example of a possible deployment scenario, in which Service/Network Orchestrator can include a number of functionalities, among which, in the example below, PNC functionalities for domain 2 and MDSC functionalities to coordinate the PNC1 functionalities (hosted in a separate domain controller) and PNC2 functionalities (co-hosted in the network orchestrator).



APPENDIX B - Example of IP + Optical network with L3VPN service

This section provides a more complex deployment scenario in which ACTN hierarchy is deployed to control a multi-layer network via an IP/MPLS PNC and an Optical PNC. The scenario is further enhanced by the introduction of an upper layer service configuration (e.g.

L3VPN). The provisioning of the L3VPN service is outside ACTN scope but it is worth showing how the two parts are integrated for the end to end service fulfilment. An example of service configuration function in the Service/Network Orchestrator is discussed in [I-D.dhjain-bess-bgp-l3vpn-yang].



TEAS Working Group

Young Lee (Editor)
Dhruv Dhody
Huawei

Internet Draft

Intended status: Informational

Sergio Belotti
Nokia

Expires: November 2017

Khuzema Pithewan
Infinera

Daniele Ceccarelli
Ericsson

Takuya Miyasaka
KDDI

Jong Yoon Shin
SKT

May 12, 2017

Requirements for Abstraction and Control of TE Networks

draft-ietf-teas-actn-requirements-05.txt

Abstract

This document provides a set of requirements for abstraction and control of Traffic Engineering networks to 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.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that

other groups may also distribute working documents as Internet-Drafts.

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

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

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

This Internet-Draft will expire on November 12, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document provides a set of requirements for Abstraction and Control of Traffic Engineering (TE) Networks (ACTN) identified in various use-cases specified by the operators. [ACTN-frame] defines the base reference architecture and terminology.

ACTN refers to the set of virtual network service operations needed to orchestrate, control and manage large-scale multi-domain TE networks so as to facilitate network programmability, automation, efficient resource sharing, and end-to-end virtual service aware connectivity.

These operations are summarized as follows:

- Abstraction and coordination of underlying network resources independent of how these resources are managed or controlled, so that higher-layer entities can dynamically control virtual networks based on those resources. Control includes creating, modifying, monitoring, and deleting virtual networks.

- Collation of the resources from multiple TE networks (multiple technologies, equipment from multiple vendors, under the control of multiple administrations) through a process of hierarchical abstraction to present a customer with a single virtual network. This is achieved by presenting the network domain as an abstracted topology to the customer via open and programmable interfaces. Hierarchical abstraction allows for the recursion of controllers in a customer-provider relationship.

- Orchestration of end-to-end virtual network services and applications via allocation of network resources to meet specific service, application and customer requirements.

- Adaptation of customer requests (to control virtual resources) to the physical network resources performing the necessary mapping, translation, isolation and, policy that allows conveying, managing and enforcing customer policies with respect to the services and the network of the customer.

- Provision via a data model of a computation scheme and virtual control capability to customers who request virtual network

services. Note that these customers could, themselves, be service providers.

ACTN solutions will build on, and extend, existing TE constructs and TE mechanisms wherever possible and appropriate. Support for controller-based approaches is specifically included in the possible solution set.

2. High-level ACTN requirements

This section provides a summary of use-cases in terms of two categories: (i) service-specific requirements; (ii) network-related requirements. All these requirements are specified by operators that are interested in implementing ACTN.

Service-specific requirements listed below are uniquely applied to the work scope of ACTN. Service-specific requirements are related to the virtual service coordination function. 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 and necessary for coherent/seamless for the virtual network operation function. These requirements are related to multi-domain and multi-layer signaling, routing, protection/restoration and synergy, re-optimization/re-grooming, etc.

2.1. Service-Specific Requirements

1. Requirement 1: Virtual Network Service (VNS) creation

Customer MUST be able to request/instantiate the VNS to the network within the confines of mutual agreement between customer and network operator and network operator's capability. There are different types of VNS in terms of the VN types the customer is allowed to operate (e.g., a VN type can be simply a set of end-to-end tunnels, or it can comprise of virtual nodes and links in mesh fashion, etc.). The customer MUST be able to express VNS policy that captures Service Level Agreements (SLA) associated with virtual network

service (e.g., Endpoint selection policy, routing policy, time-related policy, etc.)

Reference: [KLEE], [LOPEZ], [SHIN], [DHODY], [FANG].

2. Requirement 2: Virtual Network Service Query

Customer SHOULD be able to request VNS Query ("Can you give me these VN(s)?") that include the following parameters:

- VN type: various VN types defined by the customer (e.g., path, graph, etc.)
- VN end-points (Customer Edge interface information)
- VN Topology Service-specific Objective Functions (e.g., maximum bandwidth, minimum latency, minimum hops, etc. and any combination of them).
- VN constraints requirement (e.g., Maximum Latency threshold, Minimum Bandwidth, etc.)

Reference: [KUMAKI], [FANG], [CHENG].

3. Requirement 3: VNS Instantiation ("Please create a VNS for me")

Customer MUST be able to instantiate VNS that includes various VNS related parameters:

- VN type: various VN types defined by the customer (e.g., path, graph, etc.)
- VN end-points (Customer Edge interface information)
- VN Topology Service-specific Objective Functions (e.g., maximum bandwidth, minimum latency, minimum hops, etc. and any combination of them).
- VN constraints requirement (e.g., Maximum Latency threshold, Minimum Bandwidth, etc.)
- VN Topology diversity when there are multiple instances of VNS (e.g., VN1 and VN2 must be disjoint; Node/link disjoint from other VNs)

Reference: [KUMAKI], [FANG], [CHENG].

4. Requirement 4: VNS Lifecycle Management & Operation (M&O)

Customer MUST be able to perform the following VNS operations:

- VNS Delete: Customer MUST be able to delete VNS.
- VNS Modify: Customer MUST be able to modify VNS related parameters during the lifecycle of the instantiated VNS.

Reference: [FANG], [KUMAKI], [LOPEZ], [DHODY], [FANG], [KLEE].

5. Requirement 5: VNS Isolation

Customer's VN should be able to use arbitrary network topology, routing, or forwarding functions as well as customized control mechanisms independent of the underlying physical network and of other coexisting virtual networks. Other customers' VNS operation MUST not impact a particular customer's VNS network operation.

Reference: [KUMAKI], [FANG], [LOPEZ]

6. Requirement 6: Multi-Destination Coordination

Customer MUST be able to define and convey service/preference requirements for multi-destination applications (e.g., set of candidate sources/destinations, thresholds for load balancing, disaster recovery policy, etc.)

Reference: [FANG], [LOPEZ], [SHIN].

7. Requirement 7: VNS Performance Monitoring

The customer MUST be able to define performance monitoring parameters and its associated policy such as frequency of report, abstraction/aggregation level of performance data (e.g., VN level, tunnel level, virtual link/node level, etc.) with dynamic feedback loop from the network.

Reference: [XU], [XU2], [DHODY], [CHENG]

8. Requirement 8: VNS Confidentiality and Security Requirements

The following confidentiality/security requirements MUST be supported in all interfaces:

- Securing the request and control of resources, confidentially of the information, and availability of function.
- Trust domain verification (external entity versus internal entity)
- Encrypting data that flow between components, especially when they are implemented at remote nodes, regardless if these are external or internal network interfaces.

2.2. Network-Related Requirements

1. Requirement 1: Virtual Network Service Coordination

Network MUST be able to support the following VNS operations:

- VNS Delete: Upon customer’s VNS deletion request, network MUST be able to delete VNS.
- VNS Modify: Upon customer’s VNS modification request, network MUST be able to modify VNS related parameters during the lifecycle of the instantiated VNS.
- VNS Update: Upon customer’s VNS performance monitoring setup, the network MUST be able to support VNS level Operations, Administration and Management (OAM) Monitoring under policy agreement.

Reference: [FANG], [KUMAKI], [LOPEZ], [DHODY], [FANG], [KLEE].

2. Requirement 2: Topology Abstraction Capability

The network MUST be capable of managing its networks based on the principle of topology abstraction to be able to scale multi-layer, multi-domain networks.

Reference: [KLEE], [LOPEZ], [DHODY], [CHENG].

3. Requirement 3: Multi-Domain & Multi-layer Coordination

Network coordination for multi-domain and multi-layer path computation and path setup operation MUST be provided:

- End-to-end path computation across multi-domain networks (based on abstract topology from each domain)
- Domain sequence determination
- Request for path signaling to each domain controller
- Alternative path computation if any of the domain controllers cannot find its domain path

Reference: [CHENG], [DHODY], [KLEE], [LOPEZ], [SHIN], [SUZUKI].

4. Requirement 4: End-to-End Path Restoration

End-to-end Path Restoration Operations MUST be provided with seamless coordination between domain-level recovery schemes and cross-domain recovery schemes.

Reference: [CHENG], [KLEE], [DHODY], [LOPEZ], [SHIN].

5. Requirement 5: Dynamicity of virtual network control operations

Dynamic virtual network control operations MUST be supported. This includes, but is not limited to, the following:

- Real-time VNS control (e.g., fast recovery/reroute upon network failure).
- Fast convergence of abstracted topologies upon changes due to failure or reconfiguration across the network domain view, the multi-domain network view and the customer view.
- Large-scale VNS operation (e.g., the ability to query tens of thousands of nodes, and to examine tens of thousands of connectivity requests) for time-sensitive applications.

Reference: [SHIN], [XU], [XU2], [KLEE], [KUMAKI], [SUZUKI].

3. References

3.1. Normative References

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

3.2. Informative References

[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 E2E Network Services in Multiple Vendor Domain Transport Networks", draft-klee-teas-actn-connectivity-multi-domain, work-in-progress.

[KUMAKI] K. Kumaki, T. Miyasaka, "ACTN : Use case for Multi Tenant VNO", draft-kumaki-teas-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.

[XU2] Y. Xu, et. al., "Requirements of Abstract Alarm Report in ACTN architecture", draft-xu-teas-actn-abstract-alarm-report, work-in-progress.

[SUZUKI] T. Suzuki, et. al., "Use-case and Requirements for Multi-domain Operation Plane Change", draft-suzuki-teas-actn-multidomain-opc, work-in-progress.

4. Contributors

Kwangkook Lee
KT
Email: kwangkooglee@gmail.com

Yunbin Xu
CATR
Email: xuyunbin@mail.ritt.com.cn

Toshiaki Suzuki
Hitachi
Email: toshiaki.suzuki.cs@hitachi.com

Haomian Zheng
Huawei
Email: zhenghaomian@huawei.com

Authors' Addresses

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

Dhruv Dhody
Huawei Technologies

Email: dhruv.ietf@gmail.com

Sergio Belotti
Nokia
Via Trento, 30
Vimercate, Italy
Email: sergio.belotti@nokia.com

Khuzema Pithewan
Infinera
Email: kpithewan@infinera.com

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

Takuya Miyasaka
KDDI
Email: ta-miyasaka@kddi.com

Jong Yoon Shin
SKT
Email: jongyoon.shin@sk.com

TEAS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2017

V. Beeram
Juniper Networks
T. Saad, Ed.
R. Gandhi
Cisco Systems, Inc.
X. Liu
Jabil
I. Bryskin
Huawei Technologies
H. Shah
Ciena

March 10, 2017

A YANG Data Model for Resource Reservation Protocol (RSVP)
draft-ietf-teas-yang-rsvp-07

Abstract

This document defines a YANG data model for the configuration and management of RSVP Protocol. The model covers the building blocks of the RSVP protocol that can be augmented and used by other RSVP extension models such as RVSP extensions to Traffic-Engineering (RSVP-TE). The model covers the configuration, operational state, remote procedural calls, and event notifications data.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2017.

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 interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage the RSVP protocol [RFC2205]. This model covers RSVP protocol building blocks that can be augmented and used by other RSVP extension models- such as for signaling RSVP-TE MPLS (or other technology specific) Label Switched Paths (LSP)s.

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]
rt-type	ietf-routing-types	XX
key-chain	ietf-key-chain	XX

Table 1: Prefixes and corresponding YANG modules

2. Design Considerations

2.1. Module Hierarchy

The RSVP base YANG module augments the "control-plane-protocol" list in ietf-routing [RFC8022] module with specific RSVP parameters in an "rsvp" container. It also defines an extensiion identity "rsvp" of base "rt:routing-protocol" to identify the RSVP protocol.

During modeling discussion, some RSVP features are categorized as core to the functionality of the protocol, and hence, are supported by all vendors claiming the support for RSVP. These features' configuration and state were grouped in the RSVP base module.

Other extended RSVP features are categorized as either optional or providing additional knobs to provide better tune basic functionality of the RSVP protocol. The support for extended RSVP features by all vendors was considered optional. Such features were grouped in a separate RSVP extended module.

The augmentation of the RSVP model by other models (e.g. RSVP-TE for MPLS or other technologies) are considered outside the scope of this document and discussed in separate document(s).

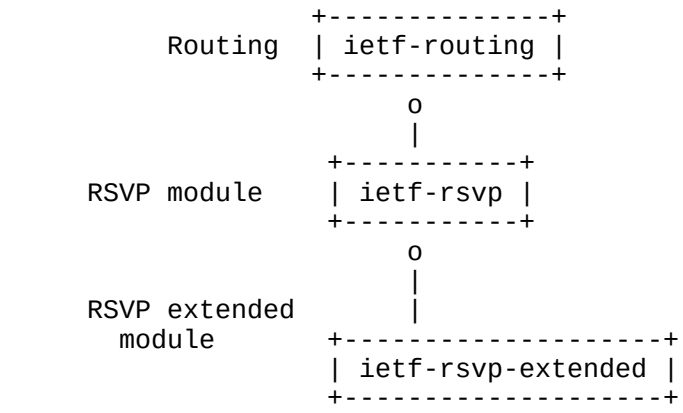


Figure 1: Relationship of RSVP and RSVP extended modules with other protocol modules

The RSVP base model does not aim to be feature complete. The primary intent is to cover a set of standard core features (listed below) that are commonly in use.

- Authentication ([RFC2747])
- Refresh Reduction ([RFC2961])
- Hellos ([RFC3209])
- Graceful Restart ([RFC3473], [RFC5063])

The extended RSVP YANG model covers non-basic configuration(s) for RSVP feature(s) as well as optional RSVP feature that are not a must for basic RSVP operation.

2.2. Data Organization

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.

The approach described in [I-D.openconfig-netmod-opstate] allows for modeling the respective intended and applied configuration and derived state. The state data can be categorized into one of the following:

- State corresponding to applied configuration
- State corresponding to derived state, counters, stats, etc.

Pure state data (for example, protocol derived data) is placed inside the "state" sub-container, as shown in Figure 2.

2.3. Configuration Inheritance

The defined data model supports configuration inheritance for neighbors, and interfaces. Data elements defined in the main container (e.g. the container that encompasses the list of interfaces, or neighbors) are assumed to apply equally to all elements of the list, unless overridden explicitly for a certain element (e.g. interface). Vendors are expected to augment the above container(s) to provide the list of inheritance command for their implementations.

3. Model Organization

This document divides the RSVP model into two modules: the RSVP base and extended. Each module covers the configuration, state, notification and RPCs of data. The relationship between the different modules is depicted in Figure 1.

3.1. RSVP Base YANG Model

This section describes the RSVP base YANG data model. The container "rsvp" is the top level container in this data model. The presence of this container enables the RSVP protocol functionality.

Data for such state is contained under the respective "state" sub-container of the intended object (e.g. interface) as shown in Figure 2.

```
module: ietf-rsvp
   +--rw rsvp!
      +--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 neighbors
               +-- rw config
                  <<intended configuration>>
               .
               +-- ro state
                  <<applied configuration>>
                  <<derived state associated with the tunnel>>
            .
            .
         +--rw sessions
               +-- rw config
                  <<intended configuration>>
               .
               +-- ro state
                  <<applied configuration>>
                  <<derived state associated with the tunnel>>
            .
      rpcs:
         +--x global-rpc
         +--x interfaces-rpc
         +--x neighbors-rpc
         +--x sessions-rpc
      notifications:
         +--n global-notif
         +--n interfaces-notif
         +--n neighbors-notif
         +--n sessions-notif
```

Figure 2: RSVP high-level tree model view

The following subsections provide overview of the parts of the model pertaining to configuration and state data.

Configuration and state data are organized into those applicable globally (node scope), per interfaces, per neighbors, or per session.

3.1.1. Global Data

This branch of the data model covers global configuration and states that control RSVP protocol behavior.

3.1.2. Interface Data

This branch of the data model covers configuration and state elements relevant to one or all RSVP interfaces. Any data configuration applied at the "interfaces" container level are equally applicable to all interfaces - unless overridden by explicit configuration under a specific interface.

3.1.3. Neighbor Data

This branch of the data model covers configuration of elements relevant to RSVP neighbors. This would be discussed in detail in future revisions.

3.1.4. Session Data

This branch of the data model covers configuration of elements relevant to RSVP sessions. This would be discussed in detail in future revisions.

3.1.5. Tree Diagram

```
module: ietf-rsvp
  augment
  /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
    +--rw rsvp!
       +--rw globals
       |  +--rw sessions
       |  |  +--ro session* [local-index]
       |  |     +--ro local-index    -> ../state/local-index
       |  |     +--ro state
       |  |        +--ro local-index?        uint64
       |  |        +--ro destination-port?   inet:port-number
       |  |        +--ro source?             inet:ip-address
       |  |        +--ro destination?        inet:ip-address
       |  |        +--ro session-name?       string
       |  |        +--ro session-state?      enumeration
```

```
|  |        +--ro session-type?       identityref
|  |        +--ro psbs
|  |        |  +--ro psb*
|  |        |     +--ro source-port?   inet:port-number
|  |        |     +--ro expires-in?    uint32
|  |        +--ro rsbs
|  |           +--ro rsb*
|  |              +--ro source-port?         inet:port-number
|  |              +--ro reservation-style?   identityref
|  |              +--ro expires-in?          uint32
|  +--rw statistics
|  |  +--ro state
|  |     +--ro messages
|  |     |  +--ro ack-sent?                       yang:counter64
|  |     |  +--ro ack-received?                   yang:counter64
|  |     |  +--ro bundle-sent?                    yang:counter64
|  |     |  +--ro bundle-received?                yang:counter64
|  |     |  +--ro hello-sent?                     yang:counter64
|  |     |  +--ro hello-received?                 yang:counter64
|  |     |  +--ro integrity-challenge-sent?       yang:counter64
|  |     |  +--ro integrity-challenge-received?   yang:counter64
|  |     |  +--ro integrity-response-sent?        yang:counter64
|  |     |  +--ro integrity-response-received?    yang:counter64
|  |     |  +--ro notify-sent?                    yang:counter64
|  |     |  +--ro notify-received?                yang:counter64
|  |     |  +--ro path-sent?                      yang:counter64
|  |     |  +--ro path-received?                  yang:counter64
|  |     |  +--ro path-err-sent?                  yang:counter64
|  |     |  +--ro path-err-received?              yang:counter64
|  |     |  +--ro path-tear-sent?                 yang:counter64
|  |     |  +--ro path-tear-received?             yang:counter64
|  |     |  +--ro resv-sent?                      yang:counter64
|  |     |  +--ro resv-received?                  yang:counter64
|  |     |  +--ro resv-confirm-sent?              yang:counter64
|  |     |  +--ro resv-confirm-received?          yang:counter64
|  |     |  +--ro resv-err-sent?                  yang:counter64
|  |     |  +--ro resv-err-received?              yang:counter64
|  |     |  +--ro resv-tear-sent?                 yang:counter64
|  |     |  +--ro resv-tear-received?             yang:counter64
|  |     |  +--ro summary-refresh-sent?           yang:counter64
|  |     |  +--ro summary-refresh-received?       yang:counter64
|  |     |  +--ro unknown-messages-received?      yang:counter64
|  |     +--ro packets
|  |     |  +--ro sent?       yang:counter64
|  |     |  +--ro received?   yang:counter64
|  |     +--ro errors
|  |        +--ro authenticate?   yang:counter64
|  |        +--ro checksum?       yang:counter64
```

```
|  |        +--ro packet-len?     yang:counter64
|  +--rw graceful-restart
|     +--rw config
|     |  +--rw enabled?   boolean
|     +--ro state
|        +--ro enabled?   boolean
+--rw interfaces
|  +--rw config
|  +--ro state
|  +--rw refresh-reduction
|  |  +--rw config
|  |  |  +--rw enabled?   boolean
|  |  +--ro state
|  |     +--ro enabled?   boolean
|  +--rw hellos
|  |  +--rw config
|  |  |  +--rw enabled?   boolean
|  |  +--ro state
|  |     +--ro enabled?   boolean
|  +--rw authentication
|  |  +--rw config
|  |  |  +--rw enabled?              boolean
|  |  |  +--rw authentication-key?   string
|  |  |  +--rw crypto-algorithm      identityref
|  |  +--ro state
|  |     +--ro enabled?              boolean
|  |     +--ro authentication-key?   string
|  |     +--ro crypto-algorithm      identityref
|  +--rw statistics
|  |  +--ro state
|  |     +--ro messages
|  |     |  +--ro ack-sent?                        yang:counter64
|  |     |  +--ro ack-received?                    yang:counter64
|  |     |  +--ro bundle-sent?                     yang:counter64
|  |     |  +--ro bundle-received?                 yang:counter64
|  |     |  +--ro hello-sent?                      yang:counter64
|  |     |  +--ro hello-received?                  yang:counter64
|  |     |  +--ro integrity-challenge-sent?        yang:counter64
|  |     |  +--ro integrity-challenge-received?    yang:counter64
|  |     |  +--ro integrity-response-sent?         yang:counter64
|  |     |  +--ro integrity-response-received?     yang:counter64
|  |     |  +--ro notify-sent?                     yang:counter64
|  |     |  +--ro notify-received?                 yang:counter64
|  |     |  +--ro path-sent?                       yang:counter64
|  |     |  +--ro path-received?                   yang:counter64
|  |     |  +--ro path-err-sent?                   yang:counter64
|  |     |  +--ro path-err-received?               yang:counter64
|  |     |  +--ro path-tear-sent?                  yang:counter64
```

```
|  |     |  +--ro path-tear-received?            yang:counter64
|  |     |  +--ro resv-sent?                     yang:counter64
|  |     |  +--ro resv-received?                 yang:counter64
|  |     |  +--ro resv-confirm-sent?             yang:counter64
|  |     |  +--ro resv-confirm-received?         yang:counter64
|  |     |  +--ro resv-err-sent?                 yang:counter64
|  |     |  +--ro resv-err-received?             yang:counter64
|  |     |  +--ro resv-tear-sent?                yang:counter64
|  |     |  +--ro resv-tear-received?            yang:counter64
|  |     |  +--ro summary-refresh-sent?          yang:counter64
|  |     |  +--ro summary-refresh-received?      yang:counter64
|  |     |  +--ro unknown-messages-received?     yang:counter64
|  |     +--ro packets
|  |     |  +--ro sent?       yang:counter64
|  |     |  +--ro received?   yang:counter64
|  |     +--ro errors
|  |        +--ro authenticate?   yang:counter64
|  |        +--ro checksum?       yang:counter64
|  |        +--ro packet-len?     yang:counter64
|  +--rw interface* [interface]
|     +--rw interface            if:interface-ref
|     +--rw config
|     +--ro state
|     +--rw refresh-reduction
|     |  +--rw config
|     |  |  +--rw enabled?   boolean
|     |  +--ro state
|     |     +--ro enabled?   boolean
|     +--rw hellos
|     |  +--rw config
|     |  |  +--rw enabled?   boolean
|     |  +--ro state
|     |     +--ro enabled?   boolean
|     +--rw authentication
|     |  +--rw config
|     |  |  +--rw enabled?              boolean
|     |  |  +--rw authentication-key?   string
|     |  |  +--rw crypto-algorithm      identityref
|     |  +--ro state
|     |     +--ro enabled?              boolean
|     |     +--ro authentication-key?   string
|     |     +--ro crypto-algorithm      identityref
|     +--rw statistics
|        +--ro state
|           +--ro messages
|           |  +--ro ack-sent?                    yang:counter64
|           |  +--ro ack-received?                yang:counter64
|           |  +--ro bundle-sent?                 yang:counter64
```

```
|           |  +--ro bundle-received?             yang:counter64
|           |  +--ro hello-sent?                  yang:counter64
|           |  +--ro hello-received?              yang:counter64
|           |  +--ro integrity-challenge-sent?    yang:counter64
|           |  +--ro integrity-challenge-received? yang:counter64
|           |  +--ro integrity-response-sent?     yang:counter64
|           |  +--ro integrity-response-received? yang:counter64
|           |  +--ro notify-sent?                 yang:counter64
|           |  +--ro notify-received?             yang:counter64
|           |  +--ro path-sent?                   yang:counter64
|           |  +--ro path-received?               yang:counter64
|           |  +--ro path-err-sent?               yang:counter64
|           |  +--ro path-err-received?           yang:counter64
|           |  +--ro path-tear-sent?              yang:counter64
|           |  +--ro path-tear-received?          yang:counter64
|           |  +--ro resv-sent?                   yang:counter64
|           |  +--ro resv-received?               yang:counter64
|           |  +--ro resv-confirm-sent?           yang:counter64
|           |  +--ro resv-confirm-received?       yang:counter64
|           |  +--ro resv-err-sent?               yang:counter64
|           |  +--ro resv-err-received?           yang:counter64
|           |  +--ro resv-tear-sent?              yang:counter64
|           |  +--ro resv-tear-received?          yang:counter64
|           |  +--ro summary-refresh-sent?        yang:counter64
|           |  +--ro summary-refresh-received?    yang:counter64
|           |  +--ro unknown-messages-received?   yang:counter64
|           +--ro packets
|           |  +--ro sent?       yang:counter64
|           |  +--ro received?   yang:counter64
|           +--ro errors
|              +--ro authenticate?   yang:counter64
|              +--ro checksum?       yang:counter64
|              +--ro packet-len?     yang:counter64
|
+--rw neighbors
   +--rw neighbor* [address]
      +--rw address    -> ../config/address
      +--rw config
      |  +--rw address?   inet:ip-address
      +--ro state
         +--ro address?                    inet:ip-address
         +--ro epoch?                      uint32
         +--ro expiry-time?                uint32
         +--ro graceful-restart
         |  +--ro enabled?                 boolean
         |  +--ro local-restart-time?      uint32
         |  +--ro local-recovery-time?     uint32
         |  +--ro neighbor-restart-time?   uint32
         |  +--ro neighbor-recovery-time?  uint32
```

```
               |  +--ro helper-mode
               |     +--ro enabled?                              boolean
               |     +--ro max-helper-restart-time?              uint32
               |     +--ro max-helper-recovery-time?             uint32
               |     +--ro neighbor-restart-time-remaining?      uint32
               |     +--ro neighbor-recovery-time-remaining?     uint32
               +--ro hello-status?                 enumeration
               +--ro interface?                    if:interface-ref
               +--ro neighbor-state?               enumeration
               +--ro refresh-reduction-capable?    boolean
               +--ro restart-count?                yang:counter32
               +--ro restart-time?                 yang:date-and-time
```

Figure 3: RSVP model tree diagram

3.1.6. YANG Module

```
<CODE BEGINS> file "ietf-rsvp@2017-03-10.yang"
module ietf-rsvp {

  namespace "urn:ietf:params:xml:ns:yang:ietf-rsvp";

  /* Replace with IANA when assigned */
  prefix "rsvp";

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-inet-types {
    prefix inet;
  }

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

  import ietf-routing {
    prefix "rt";
  }

  import ietf-key-chain {
    prefix "key-chain";
  }

  organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
     Working Group";
```

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

        WG Chair: Lou Berger
                  <mailto:lberger@labn.net>

        WG Chair: Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Tarek Saad
                  <mailto:tsaad@cisco.com>

        Editor:   Rakesh Gandhi
                  <mailto:rgandhi@cisco.com>

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

        Editor:   Xufeng Liu
                  <mailto:Xufeng_Liu@jabil.com>

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

        Editor:   Raqib Jones
                  <mailto:raqib@Brocade.com>

        Editor:   Bin Wen
                  <mailto:Bin_Wen@cable.comcast.com>";

     description
       "This module contains the RSVP YANG data model.";

     revision "2017-03-10" {
       description "Latest revision of RSVP yang module.";
       reference "RFC2205";
     }

     identity rsvp {
       base "rt:routing-protocol";
       description "RSVP protocol";
     }

     identity rsvp-session-type {
```

```
      description "Base RSVP session type";
    }

    identity rsvp-session-ipv4 {
      base rsvp-session-type;
      description "RSVP IPv4 session type";
    }

    identity rsvp-session-ipv6 {
      base rsvp-session-type;
      description "RSVP IPv4 session type";
    }

    identity reservation-style {
      description "Base identity for reservation style";
    }

    identity reservation-wildcard-filter {
      base reservation-style;
      description "Wildcard-Filter (WF) Style";
      reference "RFC2205";
    }

    identity reservation-fixed-filter {
      base reservation-style;
      description "Fixed-Filter (FF) Style";
      reference "RFC2205";
    }

    identity reservation-shared-explicit {
      base reservation-style;
      description "Shared Explicit (SE) Style";
      reference "RFC2205";
    }

    grouping graceful-restart_config {
      description
        "Base configuration parameters relating to RSVP
         Graceful-Restart";
      leaf enabled {
        type boolean;
        description
          "'true' if RSVP Graceful Restart is enabled.
           'false' if RSVP Graceful Restart is disabled.";
      }
    }

    grouping graceful-restart {
```

```
      description
        "RSVP graceful restart parameters grouping";
      container graceful-restart {
        description
          "RSVP graceful restart parameters container";
        container config {
          description
            "Configuration parameters for graceful restart
             properties";
          uses graceful-restart_config;
        }
        container state {
          config false;
          description
            "State parameters for graceful restart
             properties";
          uses graceful-restart_config;
        }
      }
    }

    grouping refresh-reduction_config {
      description
        "Configuration parameters relating to RSVP
         refresh reduction";

      leaf enabled {
        type boolean;
        description
          "'true' if RSVP Refresh Reduction is enabled.
           'false' if RSVP Refresh Reduction is disabled.";
      }
    }

    grouping refresh-reduction {
      description
        "Top level grouping for RSVP refresh reduction
         parameters";
      container refresh-reduction {
        description
          "Top level container for RSVP refresh reduction
           parameters";
        container config {
          description
            "Configuration parameters relating to
             RSVP refresh reduction parameters";
          uses refresh-reduction_config;
        }
```

```
        container state {
          config false;
          description
            "State information associated with RSVP
             refresh reduction parameters";
          uses refresh-reduction_config;
        }
      }
    }

    grouping authentication_config {
      description
        "Configuration parameters relating to RSVP
         authentication";
      leaf enabled {
        type boolean;
        description
          "'true' if RSVP Authentication is enabled.
           'false' if RSVP Authentication is disabled.";
      }
      leaf authentication-key {
        type string;
        description
          "An authentication key string";
        reference
          "RFC 2747: RSVP Cryptographic Authentication";
      }
      leaf crypto-algorithm {
       type identityref {
         base key-chain:crypto-algorithm;
       }
       mandatory true;
       description
         "Cryptographic algorithm associated with key.";
      }
    }

    grouping authentication {
      description
        "Top level grouping for RSVP authentication parameters";
      container authentication {
        description
          "Top level container for RSVP authentication
           parameters";
        container config {
          description
            "Configuration parameters relating to
             RSVP authentication parameters";
```

```
          uses authentication_config;
        }
        container state {
          config false;
          description
            "State information associated with RSVP
             authentication parameters";
          uses authentication_config;
        }
      }
    }

    grouping hellos_config {
      description
        "Configuration parameters relating to RSVP
        hellos";
      leaf enabled {
        type boolean;
        description
          "'true' if RSVP Hello is enabled.
           'false' if RSVP Hello is disabled.";
      }
    }

    grouping hellos {
      description
        "Top level grouping for RSVP hellos parameters";
      container hellos {
        description
          "Top level container for RSVP hello parameters";
        container config {
          description
            "Configuration parameters relating to
             RSVP hello parameters";
          uses hellos_config;
        }
        container state {
          config false;
          description
            "State information associated with RSVP
             hello parameters";
          uses hellos_config;
        }
      }
    }

    grouping signaling-parameters_config {
      description
```

```
        "Configuration parameters relating to RSVP
        signaling";
    }

    grouping signaling-parameters {
      description
        "Top level grouping for RSVP signaling  parameters";
      container config {
        description
          "Configuration parameters relating to
           RSVP signaling parameters";
        uses signaling-parameters_config;
      }
      container state {
        config false;
        description
          "State information associated with RSVP
           signaling parameters";
        uses signaling-parameters_config;
      }
    }

    grouping session-attributes_state {
      description
        "Top level grouping for RSVP session properties";
      leaf local-index {
        type uint64;
        description
          "The index used to identify the RSVP session
           on the local network element. This index is
           generated by the device and is unique only
           to the local network element.";
      }
      leaf destination-port {
        type inet:port-number;
        description "RSVP destination port";
        reference "RFC2205";
      }
      leaf source {
        type inet:ip-address;
        description "RSVP source address";
        reference "RFC2205";
      }
      leaf destination {
        type inet:ip-address;
        description "RSVP destination address";
        reference "RFC2205";
      }
```

```
        leaf session-name {
          type string;
          description
            "The signaled name of this RSVP session.";
        }
        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 identityref {
            base rsvp-session-type;
          }
          description "RSVP session type";
        }
        container psbs {
          description "Path State Block container";
          list psb {
            description "List of path state blocks";
            leaf source-port {
              type inet:port-number;
              description "RSVP source port";
              reference "RFC2205";
            }
            leaf expires-in {
              type uint32;
              units seconds;
              description "Time to reservation expiry (in seconds)";
            }
          }
        }
        container rsbs {
          description "Reservation State Block container";
          list rsb {
            description "List of reservation state blocks";
            leaf source-port {
              type inet:port-number;
              description "RSVP source port";
```

```
            reference "RFC2205";
          }
          leaf reservation-style {
            type identityref {
              base reservation-style;
            }
            description "RSVP reservation style";
          }
          leaf expires-in {
            type uint32;
            units seconds;
            description "Time to reservation expiry (in seconds)";
          }
        }
      }
    }

    grouping neighbor-attributes {
      description
        "Top level grouping for RSVP neighbor properties";
      container config {
        description
          "Configuration for neighbor properties";
        leaf address {
          type inet:ip-address;
          description
            "Address of RSVP neighbor";
        }
      }
      container state {
        config false;
        description
          "State information associated with RSVP
           neighbor properties";
        uses neighbor-derived_state;
      }
    }

    grouping packets_state {
      description
        "Packet statistics grouping";
      container packets {
        description
          "Packet statistics container";
        leaf sent {
          type yang:counter64;
          description
            "Packet sent count";
```

```
        }

        leaf received {
          type yang:counter64;
          description
            "Packet received count";
        }
      }
    }

    grouping protocol_state {
      description
        "RSVP protocol statistics grouping";
      container messages {
        description
          "RSVP protocol statistics container";
        leaf ack-sent {
          type yang:counter64;
          description
            "Hello sent count";
        }

        leaf ack-received {
          type yang:counter64;
          description
            "Hello received count";
        }

        leaf bundle-sent {
          type yang:counter64;
          description
            "Bundle sent count";
        }

        leaf bundle-received {
          type yang:counter64;
          description
            "Bundle received count";
        }

        leaf hello-sent {
          type yang:counter64;
          description
            "Hello sent count";
        }

        leaf hello-received {
          type yang:counter64;
```

```
          description
            "Hello received count";
        }

        leaf integrity-challenge-sent {
          type yang:counter64;
          description
            "Integrity Challenge sent count";
        }

        leaf integrity-challenge-received {
          type yang:counter64;
          description
            "Integrity Challenge received count";
        }

        leaf integrity-response-sent {
          type yang:counter64;
          description
            "Integrity Response sent count";
        }

        leaf integrity-response-received {
          type yang:counter64;
          description
            "Integrity Response received count";
        }

        leaf notify-sent {
          type yang:counter64;
          description
            "Notify sent count";
        }

        leaf notify-received {
          type yang:counter64;
          description
            "Notify received count";
        }

        leaf path-sent {
          type yang:counter64;
          description
            "Path sent count";
        }

        leaf path-received {
          type yang:counter64;
```

```
          description
            "Path received count";
        }

        leaf path-err-sent {
          type yang:counter64;
          description
            "Path error sent count";
        }

        leaf path-err-received {
          type yang:counter64;
          description
            "Path error received count";
        }

        leaf path-tear-sent {
          type yang:counter64;
          description
            "Path tear sent count";
        }

        leaf path-tear-received {
          type yang:counter64;
          description
            "Path tear received count";
        }

        leaf resv-sent {
          type yang:counter64;
          description
            "Resv sent count";
        }

        leaf resv-received {
          type yang:counter64;
          description
            "Resv received count";
        }

        leaf resv-confirm-sent {
          type yang:counter64;
          description
            "Confirm sent count";
        }

        leaf resv-confirm-received {
          type yang:counter64;
```

```
          description
            "Confirm received count";
        }

        leaf resv-err-sent {
          type yang:counter64;
          description
            "Resv error sent count";
        }

        leaf resv-err-received {
          type yang:counter64;
          description
            "Resv error received count";
        }

        leaf resv-tear-sent {
          type yang:counter64;
          description
            "Resv tear sent count";
        }

        leaf resv-tear-received {
          type yang:counter64;
          description
            "Resv tear received count";
        }

        leaf summary-refresh-sent {
          type yang:counter64;
          description
            "Summary refresh sent count";
        }

        leaf summary-refresh-received {
          type yang:counter64;
          description
            "Summary refresh received count";
        }

        leaf unknown-messages-received {
          type yang:counter64;
          description
            "Unknown packet received count";
        }
      }
    }
```

```
    grouping errors_state {
      description
        "Error statistics state grouping";
      container errors {
        description
          "Error statistics state container";
        leaf authenticate {
          type yang:counter64;
          description
            "The total number of packets received with an
            authentication failure.";
        }

        leaf checksum {
          type yang:counter64;
          description
            "The total number of packets received with an invalid
            checksum value.";
        }

        leaf packet-len {
          type yang:counter64;
          description
            "The total number of packets received with an invalid
            packet length.";
        }
      }
    }

    grouping statistics_state {
      description "RSVP statistic attributes.";
      container statistics {
        description
          "statistics state container";
        container state {
          config false;
          description
            "State information associated with RSVP
             hello parameters";
          uses protocol_state;
          uses packets_state;
          uses errors_state;
        }
      }
    }

    grouping neighbor-derived_state {
      description
```

```
          "Derived state at neighbor level.";

        leaf address {
          type inet:ip-address;
          description
            "Address of RSVP neighbor";
        }

        leaf epoch {
          type uint32;
          description
            "Neighbor epoch.";
        }

        leaf expiry-time {
          type uint32;
          units seconds;
          description
            "Neighbor expiry time after which the neighbor state
            is purged if no states associated with it";
        }

        container graceful-restart {
          description
            "Graceful restart information.";

          leaf enabled {
            type boolean;
            description
              "'true' if graceful restart is enabled for the
              neighbor.";
          }

          leaf local-restart-time {
            type uint32;
            units seconds;
            description
              "Local node restart time";
          }

          leaf local-recovery-time {
            type uint32;
            units seconds;
            description
              "Local node recover time";
          }

          leaf neighbor-restart-time {
```

```
            type uint32;
            units seconds;
            description
              "Neighbor restart time";
          }

          leaf neighbor-recovery-time {
            type uint32;
            units seconds;
            description
              "Neighbor recover time";
          }

          container helper-mode {
            description
              "Helper mode information ";

            leaf enabled {
              type boolean;
              description
                "'true' if helper mode is enabled.";
            }

            leaf max-helper-restart-time {
              type uint32;
              units seconds;
              description
                "The time the router or switch waits after it
                 discovers that a neighboring router has gone down
                 before it declares the neighbor down";
            }

            leaf max-helper-recovery-time {
              type uint32;
              units seconds;
              description
                "The amount of time the router retains the state of its
                 RSVP neighbors while they undergo a graceful restart";
            }

            leaf neighbor-restart-time-remaining {
              type uint32;
              units seconds;
              description
                "Number of seconds remaining for neighbor to send
                 Hello message after restart.";
            }
```

```
            leaf neighbor-recovery-time-remaining {
              type uint32;
              units seconds;
              description
                "Number of seconds remaining for neighbor to
                refresh.";
            }
          } // helper-mode
        } // graceful-restart

        leaf hello-status {
          type enumeration {
            enum "enabled" {
              description
                "Enabled";
            }
            enum "disabled" {
              description
                "Disabled";
            }
            enum "restarting" {
              description
                "Restarting";
            }
          }
          description
            "Hello status";
        }

        leaf interface {
          type if:interface-ref;
          description
            "Interface where RSVP neighbor was detected";
        }

        leaf neighbor-state {
          type enumeration {
            enum "up" {
              description
                "up";
            }
            enum "down" {
              description
                "down";
            }
            enum "hello-disable" {
              description
                "hello-disable";
```

```
            }
            enum "restarting" {
              description
                "restarting";
            }
          }
          description
            "Neighbor state";
        }

        leaf refresh-reduction-capable {
          type boolean;
          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";
        }

        leaf restart-count {
          type yang:counter32;
          description
            "Number of times this neighbor restart";
        }

        leaf restart-time {
          type yang:date-and-time;
          description
            "Last restart time of the neighbor";
        }
      }

      grouping global-attributes {
        description
          "Top level grouping for RSVP global properties";
        container sessions {
          description
            "RSVP sessions container";
          list session {
            key "local-index";
            config false;
            description
              "List of RSVP sessions";

            leaf local-index {
              type leafref {
```

```
              path "../state/local-index";
            }
            description
              "Reference to the local index for the RSVP
              session";
          }
          container state {
            config false;
            description
              "State information associated with RSVP
               session parameters";
            uses session-attributes_state;
          }
        }
      }
      uses statistics_state;
    }

    grouping intf-attributes {
      description
        "Top level grouping for RSVP interface properties";
      uses signaling-parameters;
      uses refresh-reduction;
      uses hellos;
      uses authentication;
      uses statistics_state;
    }

    augment "/rt:routing/rt:control-plane-protocols/"
      + "rt:control-plane-protocol" {
      when "rt:type = 'rsvp:rsvp'" {
      description
       "This augment is only valid when routing protocol
        instance type is RSVP.";
      }
      description
        "RSVP protocol augmentation";
      container rsvp {
        presence "Enable RSVP feature";
        description "RSVP feature container";
        container globals {
          description "RSVP global properties.";
          uses global-attributes;
          uses graceful-restart;
        }

        container interfaces {
          description
```

```
            "RSVP interfaces container";
          uses intf-attributes;

          list interface {
            key "interface";
            description
              "RSVP interfaces.";
            leaf interface {
              type if:interface-ref;
              description
                "RSVP interface.";
            }
            uses intf-attributes;
          }
        }

        container neighbors {
          description
            "RSVP neighbors container";
          list neighbor {
            key "address";
            description
              "List of RSVP neighbors";
            leaf address {
              type leafref {
                path "../config/address";
              }
              description
                "Address of RSVP neighbor";
            }
            uses neighbor-attributes;
          }
        }
      }
    }
  }
  <CODE ENDS>
```

3.2. RSVP Extended YANG Model

The RSVP extended YANG model covers optional or non-core RSVP feature(s). It also covers feature(s) that are not necessarily supported by all vendors, and hence, guarded with "if-feature" checks.

3.2.1. Tree Diagram

Figure 4 shows the YANG tree representation for configuration and state data that is augmenting the RSVP basic module:

```
module: ietf-rsvp
  augment
    /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
    +--rw rsvp!
       +--rw globals
       |  +--rw sessions
       |  |  +--ro session* [local-index]
       |  |     +--ro local-index    -> ../state/local-index
       |  |     +--ro state
       |  |        +--ro local-index?        uint64
       |  |        +--ro destination-port?   inet:port-number
       |  |        +--ro source?             inet:ip-address
       |  |        +--ro destination?        inet:ip-address
       |  |        +--ro session-name?       string
       |  |        +--ro session-state?      enumeration
       |  |        +--ro session-type?       identityref
       |  |        +--ro psbs
       |  |        |  +--ro psb*
       |  |        |     +--ro source-port?   inet:port-number
       |  |        |     +--ro expires-in?    uint32
       |  |        +--ro rsbs
       |  |           +--ro rsb*
       |  |              +--ro source-port?         inet:port-number
       |  |              +--ro reservation-style?   identityref
       |  |              +--ro expires-in?          uint32
       |  +--rw statistics
       |  |  +--ro state
       |  |     +--ro messages
       |  |     |  +--ro ack-sent?                       yang:counter64
       |  |     |  +--ro ack-received?                   yang:counter64
       |  |     |  +--ro bundle-sent?                    yang:counter64
       |  |     |  +--ro bundle-received?                yang:counter64
       |  |     |  +--ro hello-sent?                     yang:counter64
       |  |     |  +--ro hello-received?                 yang:counter64
       |  |     |  +--ro integrity-challenge-sent?       yang:counter64
       |  |     |  +--ro integrity-challenge-received?   yang:counter64
       |  |     |  +--ro integrity-response-sent?        yang:counter64
       |  |     |  +--ro integrity-response-received?    yang:counter64
       |  |     |  +--ro notify-sent?                    yang:counter64
       |  |     |  +--ro notify-received?                yang:counter64
       |  |     |  +--ro path-sent?                      yang:counter64
       |  |     |  +--ro path-received?                  yang:counter64
       |  |     |  +--ro path-err-sent?                  yang:counter64
```

```
|  |     |  +--ro path-err-received?              yang:counter64
|  |     |  +--ro path-tear-sent?                 yang:counter64
|  |     |  +--ro path-tear-received?             yang:counter64
|  |     |  +--ro resv-sent?                      yang:counter64
|  |     |  +--ro resv-received?                  yang:counter64
|  |     |  +--ro resv-confirm-sent?              yang:counter64
|  |     |  +--ro resv-confirm-received?          yang:counter64
|  |     |  +--ro resv-err-sent?                  yang:counter64
|  |     |  +--ro resv-err-received?              yang:counter64
|  |     |  +--ro resv-tear-sent?                 yang:counter64
|  |     |  +--ro resv-tear-received?             yang:counter64
|  |     |  +--ro summary-refresh-sent?           yang:counter64
|  |     |  +--ro summary-refresh-received?       yang:counter64
|  |     |  +--ro unknown-messages-received?      yang:counter64
|  |     +--ro packets
|  |     |  +--ro sent?                           yang:counter64
|  |     |  +--ro received?                       yang:counter64
|  |     |  +--ro rsvp-ext:discontinuity-time? yang:date-and-time
|  |     |  +--ro rsvp-ext:out-dropped?           yang:counter64
|  |     |  +--ro rsvp-ext:in-dropped?            yang:counter64
|  |     |  +--ro rsvp-ext:out-error?             yang:counter64
|  |     |  +--ro rsvp-ext:in-error?              yang:counter64
|  |     +--ro errors
|  |        +--ro authenticate?   yang:counter64
|  |        +--ro checksum?       yang:counter64
|  |        +--ro packet-len?     yang:counter64
|  +--rw graceful-restart
|     +--rw config
|     |  +--rw enabled?                  boolean
|     |  +--rw rsvp-ext:restart-time?    uint32
|     |  +--rw rsvp-ext:recovery-time?   uint32
|     +--ro state
|        +--ro enabled?                  boolean
|        +--ro rsvp-ext:restart-time?    uint32
|        +--ro rsvp-ext:recovery-time?   uint32
+--rw interfaces
|  +--rw config
|  |  +--rw rsvp-ext:refresh-interval?        uint32
|  |  +--rw rsvp-ext:refresh-misses?          uint32
|  |  +--rw rsvp-ext:checksum?                boolean
|  |  +--rw rsvp-ext:patherr-state-removal?   empty
|  +--ro state
|  |  +--ro rsvp-ext:refresh-interval?        uint32
|  |  +--ro rsvp-ext:refresh-misses?          uint32
|  |  +--ro rsvp-ext:checksum?                boolean
|  |  +--ro rsvp-ext:patherr-state-removal?   empty
|  +--rw refresh-reduction
|  |  +--rw config
```

```
|  |  |  +--rw enabled?                                boolean
|  |  |  +--rw rsvp-ext:bundle-message-max-size?    uint32
|  |  |  +--rw rsvp-ext:reliable-ack-hold-time?     uint32
|  |  |  +--rw rsvp-ext:reliable-ack-max-size?      uint32
|  |  |  +--rw rsvp-ext:reliable-retransmit-time?   uint32
|  |  |  +--rw rsvp-ext:reliable-srefresh?          empty
|  |  |  +--rw rsvp-ext:summary-max-size?           uint32
|  |  +--ro state
|  |     +--ro enabled?                             boolean
|  |     +--ro rsvp-ext:bundle-message-max-size?    uint32
|  |     +--ro rsvp-ext:reliable-ack-hold-time?     uint32
|  |     +--ro rsvp-ext:reliable-ack-max-size?      uint32
|  |     +--ro rsvp-ext:reliable-retransmit-time?   uint32
|  |     +--ro rsvp-ext:reliable-srefresh?          empty
|  |     +--ro rsvp-ext:summary-max-size?           uint32
|  +--rw hellos
|  |  +--rw config
|  |  |  +--rw enabled?                    boolean
|  |  |  +--rw rsvp-ext:interface-based?   empty
|  |  |  +--rw rsvp-ext:hello-interval?    uint32
|  |  |  +--rw rsvp-ext:hello-misses?      uint32
|  |  +--ro state
|  |     +--ro enabled?                    boolean
|  |     +--ro rsvp-ext:interface-based?   empty
|  |     +--ro rsvp-ext:hello-interval?    uint32
|  |     +--ro rsvp-ext:hello-misses?      uint32
|  +--rw authentication
|  |  +--rw config
|  |  |  +--rw enabled?                boolean
|  |  |  +--rw authentication-key?     string
|  |  |  +--rw crypto-algorithm        identityref
|  |  |  +--rw rsvp-ext:lifetime?      uint32
|  |  |  +--rw rsvp-ext:window-size?   uint32
|  |  |  +--rw rsvp-ext:challenge?     empty
|  |  |  +--rw rsvp-ext:retransmits?   uint32
|  |  |  +--rw rsvp-ext:key-chain?     key-chain:key-chain-ref
|  |  +--ro state
|  |     +--ro enabled?                boolean
|  |     +--ro authentication-key?     string
|  |     +--ro crypto-algorithm        identityref
|  |     +--ro rsvp-ext:lifetime?      uint32
|  |     +--ro rsvp-ext:window-size?   uint32
|  |     +--ro rsvp-ext:challenge?     empty
|  |     +--ro rsvp-ext:retransmits?   uint32
|  |     +--ro rsvp-ext:key-chain?     key-chain:key-chain-ref
|  +--rw statistics
|  |  +--ro state
|  |     +--ro messages
```

```
|  |     |  +--ro ack-sent?                          yang:counter64
|  |     |  +--ro ack-received?                      yang:counter64
|  |     |  +--ro bundle-sent?                       yang:counter64
|  |     |  +--ro bundle-received?                   yang:counter64
|  |     |  +--ro hello-sent?                        yang:counter64
|  |     |  +--ro hello-received?                    yang:counter64
|  |     |  +--ro integrity-challenge-sent?          yang:counter64
|  |     |  +--ro integrity-challenge-received?      yang:counter64
|  |     |  +--ro integrity-response-sent?           yang:counter64
|  |     |  +--ro integrity-response-received?       yang:counter64
|  |     |  +--ro notify-sent?                       yang:counter64
|  |     |  +--ro notify-received?                   yang:counter64
|  |     |  +--ro path-sent?                         yang:counter64
|  |     |  +--ro path-received?                     yang:counter64
|  |     |  +--ro path-err-sent?                     yang:counter64
|  |     |  +--ro path-err-received?                 yang:counter64
|  |     |  +--ro path-tear-sent?                    yang:counter64
|  |     |  +--ro path-tear-received?                yang:counter64
|  |     |  +--ro resv-sent?                         yang:counter64
|  |     |  +--ro resv-received?                     yang:counter64
|  |     |  +--ro resv-confirm-sent?                 yang:counter64
|  |     |  +--ro resv-confirm-received?             yang:counter64
|  |     |  +--ro resv-err-sent?                     yang:counter64
|  |     |  +--ro resv-err-received?                 yang:counter64
|  |     |  +--ro resv-tear-sent?                    yang:counter64
|  |     |  +--ro resv-tear-received?                yang:counter64
|  |     |  +--ro summary-refresh-sent?              yang:counter64
|  |     |  +--ro summary-refresh-received?          yang:counter64
|  |     |  +--ro unknown-messages-received?         yang:counter64
|  |     +--ro packets
|  |     |  +--ro sent?       yang:counter64
|  |     |  +--ro received?   yang:counter64
|  |     +--ro errors
|  |        +--ro authenticate?   yang:counter64
|  |        +--ro checksum?       yang:counter64
|  |        +--ro packet-len?     yang:counter64
|  +--rw interface* [interface]
|     +--rw interface            if:interface-ref
|     +--rw config
|     |  +--rw rsvp-ext:refresh-interval?        uint32
|     |  +--rw rsvp-ext:refresh-misses?          uint32
|     |  +--rw rsvp-ext:checksum?                boolean
|     |  +--rw rsvp-ext:patherr-state-removal?   empty
|     +--ro state
|     |  +--ro rsvp-ext:refresh-interval?        uint32
|     |  +--ro rsvp-ext:refresh-misses?          uint32
|     |  +--ro rsvp-ext:checksum?                boolean
|     |  +--ro rsvp-ext:patherr-state-removal?   empty
```

```
|     +--rw refresh-reduction
|     |  +--rw config
|     |  |  +--rw enabled?                              boolean
|     |  |  +--rw rsvp-ext:bundle-message-max-size?     uint32
|     |  |  +--rw rsvp-ext:reliable-ack-hold-time?      uint32
|     |  |  +--rw rsvp-ext:reliable-ack-max-size?       uint32
|     |  |  +--rw rsvp-ext:reliable-retransmit-time?    uint32
|     |  |  +--rw rsvp-ext:reliable-srefresh?           empty
|     |  |  +--rw rsvp-ext:summary-max-size?            uint32
|     |  +--ro state
|     |     +--ro enabled?                              boolean
|     |     +--ro rsvp-ext:bundle-message-max-size?     uint32
|     |     +--ro rsvp-ext:reliable-ack-hold-time?      uint32
|     |     +--ro rsvp-ext:reliable-ack-max-size?       uint32
|     |     +--ro rsvp-ext:reliable-retransmit-time?    uint32
|     |     +--ro rsvp-ext:reliable-srefresh?           empty
|     |     +--ro rsvp-ext:summary-max-size?            uint32
|     +--rw hellos
|     |  +--rw config
|     |  |  +--rw enabled?                    boolean
|     |  |  +--rw rsvp-ext:interface-based?   empty
|     |  |  +--rw rsvp-ext:hello-interval?    uint32
|     |  |  +--rw rsvp-ext:hello-misses?      uint32
|     |  +--ro state
|     |     +--ro enabled?                    boolean
|     |     +--ro rsvp-ext:interface-based?   empty
|     |     +--ro rsvp-ext:hello-interval?    uint32
|     |     +--ro rsvp-ext:hello-misses?      uint32
|     +--rw authentication
|     |  +--rw config
|     |  |  +--rw enabled?                boolean
|     |  |  +--rw authentication-key?     string
|     |  |  +--rw crypto-algorithm        identityref
|     |  |  +--rw rsvp-ext:lifetime?      uint32
|     |  |  +--rw rsvp-ext:window-size?   uint32
|     |  |  +--rw rsvp-ext:challenge?     empty
|     |  |  +--rw rsvp-ext:retransmits?   uint32
|     |  |  +--rw rsvp-ext:key-chain?     key-chain:key-chain-ref
|     |  +--ro state
|     |     +--ro enabled?                boolean
|     |     +--ro authentication-key?     string
|     |     +--ro crypto-algorithm        identityref
|     |     +--ro rsvp-ext:lifetime?      uint32
|     |     +--ro rsvp-ext:window-size?   uint32
|     |     +--ro rsvp-ext:challenge?     empty
|     |     +--ro rsvp-ext:retransmits?   uint32
|     |     +--ro rsvp-ext:key-chain?     key-chain:key-chain-ref
|     +--rw statistics
```

```
|        +--ro state
|           +--ro messages
|           |  +--ro ack-sent?                     yang:counter64
|           |  +--ro ack-received?                 yang:counter64
|           |  +--ro bundle-sent?                  yang:counter64
|           |  +--ro bundle-received?              yang:counter64
|           |  +--ro hello-sent?                   yang:counter64
|           |  +--ro hello-received?               yang:counter64
|           |  +--ro integrity-challenge-sent?     yang:counter64
|           |  +--ro integrity-challenge-received? yang:counter64
|           |  +--ro integrity-response-sent?      yang:counter64
|           |  +--ro integrity-response-received?  yang:counter64
|           |  +--ro notify-sent?                  yang:counter64
|           |  +--ro notify-received?              yang:counter64
|           |  +--ro path-sent?                    yang:counter64
|           |  +--ro path-received?                yang:counter64
|           |  +--ro path-err-sent?                yang:counter64
|           |  +--ro path-err-received?            yang:counter64
|           |  +--ro path-tear-sent?               yang:counter64
|           |  +--ro path-tear-received?           yang:counter64
|           |  +--ro resv-sent?                    yang:counter64
|           |  +--ro resv-received?                yang:counter64
|           |  +--ro resv-confirm-sent?            yang:counter64
|           |  +--ro resv-confirm-received?        yang:counter64
|           |  +--ro resv-err-sent?                yang:counter64
|           |  +--ro resv-err-received?            yang:counter64
|           |  +--ro resv-tear-sent?               yang:counter64
|           |  +--ro resv-tear-received?           yang:counter64
|           |  +--ro summary-refresh-sent?         yang:counter64
|           |  +--ro summary-refresh-received?     yang:counter64
|           |  +--ro unknown-messages-received?    yang:counter64
|           +--ro packets
|           |  +--ro sent?       yang:counter64
|           |  +--ro received?   yang:counter64
|           +--ro errors
|              +--ro authenticate?   yang:counter64
|              +--ro checksum?       yang:counter64
|              +--ro packet-len?     yang:counter64
|
+--rw neighbors
   +--rw neighbor* [address]
      +--rw address    -> ../config/address
      +--rw config
      |  +--rw address?   inet:ip-address
      +--ro state
         +--ro address?                    inet:ip-address
         +--ro epoch?                      uint32
         +--ro expiry-time?                uint32
         +--ro graceful-restart
```

```
|  +--ro enabled?                   boolean
|  +--ro local-restart-time?        uint32
|  +--ro local-recovery-time?       uint32
|  +--ro neighbor-restart-time?     uint32
|  +--ro neighbor-recovery-time?    uint32
|  +--ro helper-mode
|     +--ro enabled?                                boolean
|     +--ro max-helper-restart-time?                uint32
|     +--ro max-helper-recovery-time?               uint32
|     +--ro neighbor-restart-time-remaining?        uint32
|     +--ro neighbor-recovery-time-remaining?       uint32
+--ro hello-status?                 enumeration
+--ro interface?                    if:interface-ref
+--ro neighbor-state?               enumeration
+--ro refresh-reduction-capable?    boolean
+--ro restart-count?                yang:counter32
+--ro restart-time?                 yang:date-and-time
```

Figure 4: RSVP extended model tree diagram

3.2.2. YANG Module

Figure 5 shows the RSVP extended YANG module:

<CODE BEGINS> file "ietf-rsvp-extended@2017-03-10.yang"

```
module ietf-rsvp-extended {

  namespace "urn:ietf:params:xml:ns:yang:ietf-rsvp-extended";

  prefix "rsvp-ext";

  import ietf-rsvp {
    prefix "rsvp";
  }

  import ietf-routing {
    prefix "rt";
  }

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

  import ietf-key-chain {
    prefix "key-chain";
  }

  organization
```

```
       "IETF Traffic Engineering Architecture and Signaling (TEAS)
        Working Group";

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

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

        WG Chair: Lou Berger
                  <mailto:lberger@labn.net>

        WG Chair: Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Tarek Saad
                  <mailto:tsaad@cisco.com>

        Editor:   Rakesh Gandhi
                  <mailto:rgandhi@cisco.com>

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

        Editor:   Xufeng Liu
                  <mailto:Xufeng_Liu@jabil.com>

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

        Editor:   Raqib Jones
                  <mailto:raqib@Brocade.com>

        Editor:   Bin Wen
                  <mailto:Bin_Wen@cable.comcast.com>";

     description
       "This module contains the Extended RSVP YANG data model.";

     revision "2017-03-10" {
       description "Latest revision of RSVP extended yang module.";
       reference "RFC2205";
     }

     /* RSVP features */
     feature authentication {
```

```
      description
        "Indicates support for RSVP authentication";
    }

    feature error-statistics {
      description
        "Indicates support for error statistics";
    }

    feature global-statistics {
      description
        "Indicates support for global statistics";
    }

    feature graceful-restart {
      description
        "Indicates support for RSVP graceful restart";
    }

    feature hellos {
      description
        "Indicates support for RSVP hellos (RFC3209).";
    }

    feature notify {
      description
        "Indicates support for RSVP notify message (RFC3473).";
    }

    feature refresh-reduction {
      description
        "Indicates support for RSVP refresh reduction
         (RFC2961).";
    }

    feature refresh-reduction-extended {
      description
        "Indicates support for RSVP refresh reduction
         (RFC2961).";
    }

    feature per-interface-statistics {
      description
        "Indicates support for per interface statistics";
    }

    grouping graceful-restart-extended_config {
      description
```

```
        "Configuration parameters relating to RSVP
         Graceful-Restart";
      leaf restart-time {
        type uint32;
        units seconds;
        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;
        units seconds;
        description
          "RSVP state recovery time";
      }
    }

    grouping authentication-extended_config {
      description
        "Configuration parameters relating to RSVP
        authentication";
      leaf lifetime {
        type uint32 {
          range "30..86400";
        }
                          units seconds;
        description
          "Life time for each security association";
        reference
          "RFC 2747: RSVP Cryptographic
          Authentication";
      }
      leaf window-size {
        type uint32 {
          range "1..64";
        }
        description
          "Window-size to limit number of out-of-order
           messages.";
        reference
          "RFC 2747: RSVP Cryptographic
           Authentication";
      }
      leaf challenge {
        type empty;
```

```
        description
          "Enable challenge messages.";
        reference
          "RFC 2747: RSVP Cryptographic
           Authentication";
      }
      leaf retransmits {
        type uint32 {
          range "1..10000";
        }
        description
          "Number of retransmits when messages are
           dropped.";
        reference
          "RFC 2747: RSVP Cryptographic
           Authentication";
      }
      leaf key-chain {
                          type key-chain:key-chain-ref;
        description
          "Key chain name to authenticate RSVP
           signaling messages.";
        reference
          "RFC 2747: RSVP Cryptographic
           Authentication";
      }
    }

    grouping hellos-extended_config {
      description
        "Configuration parameters relating to RSVP
         hellos";
      leaf interface-based {
        type empty;
        description
          "Enable interface-based Hello adjacency if present.";
      }
      leaf hello-interval {
        type uint32;
        units milliseconds;
        description
          "Configure interval between successive Hello
           messages in milliseconds.";
        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 hello-misses {
        type uint32 {
          range "1..10";
        }
        description
          "Configure max number of consecutive missed
           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";
      }
    }

    grouping signaling-parameters-extended_config {
      description
        "Configuration parameters relating to RSVP
         signaling";
      leaf refresh-interval {
        type uint32;
        description
          "Set interval between successive refreshes";
      }
      leaf refresh-misses {
        type uint32;
        description
          "Set max number of consecutive missed
          messages for state expiry";
      }
      leaf checksum {
        type boolean;
        description
          "Enable RSVP message checksum computation";
      }
      leaf patherr-state-removal {
        type empty;
        description
          "State-Removal flag in Path Error message
          if present.";
      }
    }

    grouping refresh-reduction-extended_config {
      description
        "Configuration parameters relating to RSVP
```

```
          refresh reduction";

        leaf bundle-message-max-size {
          type uint32 {
            range "512..65000";
          }
          description
            "Configure maximum size (bytes) of a
            single RSVP Bundle message.";
        }
        leaf reliable-ack-hold-time {
          type uint32;
          units milliseconds;
          description
            "Configure hold time in milliseconds for
            sending RSVP ACK message(s).";
        }
        leaf reliable-ack-max-size {
          type uint32;
          description
            "Configure max size of a single RSVP ACK
            message.";
        }
        leaf reliable-retransmit-time {
          type uint32;
          units milliseconds;
          description
            "Configure min delay in milliseconds to
            wait for an ACK before a retransmit.";
        }
        leaf reliable-srefresh {
          type empty;
          description
            "Configure use of reliable messaging for
            summary refresh if present.";
        }
        leaf summary-max-size {
          type uint32 {
            range "20..65000";
          }
          description
            "Configure max size (bytes) of a single
            RSVP summary refresh message.";
        }
      }

      grouping packets-extended_state {
        description
```

```
        "Packet statistics.";
      leaf discontinuity-time {
        type yang:date-and-time;
        description
          "The time on the most recent occasion at which any one
          or more of the statistic counters suffered a
          discontinuity. If no such discontinuities have occurred
          since the last re-initialization of the local
          management subsystem, then this node contains the time
          the local management subsystem re-initialized itself.";
      }
      leaf out-dropped {
        type yang:counter64;
        description
          "Out packet drop count";
      }

      leaf in-dropped {
        type yang:counter64;
        description
          "In packet drop count";
      }

      leaf out-error {
        type yang:counter64;
        description
          "Out packet error count";
      }

      leaf in-error {
        type yang:counter64;
        description
          "In packet rx error count";
      }
    }

    grouping protocol-extended_state {
      description
        "RSVP protocol statistics.";
    }

    grouping errors-extended_state {
      description
        "Error statistics.";
    }

    grouping extended_state {
      description "RSVP statistic attributes.";
```

```
      uses packets-extended_state;
      uses protocol-extended_state;
      uses errors-extended_state;
    }

    /**
     * RSVP extensions augmentations
     */

    /* RSVP globals graceful restart*/

    augment "/rt:routing/rt:control-plane-protocols/" +
            "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/" +
            "rsvp:graceful-restart/rsvp:config" {
      description
        "RSVP globals configuration extensions";
      uses graceful-restart-extended_config;
    }

    augment "/rt:routing/rt:control-plane-protocols/" +
            "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/" +
            "rsvp:graceful-restart/rsvp:state" {
      description
        "RSVP globals configuration extensions";
      uses graceful-restart-extended_config;
    }

    /* RSVP statistics augmentation */
    augment "/rt:routing/rt:control-plane-protocols/" +
            "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/" +
            "rsvp:statistics/rsvp:state/rsvp:packets" {
      description
        "RSVP packet stats extensions";
      uses packets-extended_state;
    }
    augment "/rt:routing/rt:control-plane-protocols/" +
            "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/" +
            "rsvp:statistics/rsvp:state/rsvp:messages" {
      description
        "RSVP protocol message stats extensions";
      uses protocol-extended_state;
    }
    augment "/rt:routing/rt:control-plane-protocols/" +
            "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/" +
            "rsvp:statistics/rsvp:state/rsvp:errors" {
      description
        "RSVP errors stats extensions";
      uses errors-extended_state;
```

```
    }

    /**
     * RSVP all interfaces extensions
     */

    /* RSVP interface signaling extensions */
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
        + "rsvp:config" {
      description
        "RSVP signaling all interfaces configuration extensions";
      uses signaling-parameters-extended_config;
    }
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
        + "rsvp:state" {
      description
        "RSVP signaling all interfaces state extensions";
      uses signaling-parameters-extended_config;
    }

    /* RSVP refresh reduction extension */
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
        + "rsvp:refresh-reduction/rsvp:config" {
      description
        "RSVP refresh-reduction all interface configuration
         extensions";
      uses refresh-reduction-extended_config;
    }
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
        + "rsvp:refresh-reduction/rsvp:state" {
      description
        "RSVP refresh-reduction all interfaces state extensions";
      uses refresh-reduction-extended_config;
    }

    /* RSVP hellos extension */
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
        + "rsvp:hellos/rsvp:config" {
      description
        "RSVP hello all interfaces configuration extensions";
      uses hellos-extended_config;
    }
    augment "/rt:routing/rt:control-plane-protocols/"
```

```
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
        + "rsvp:hellos/rsvp:state" {
      description
        "RSVP hello all interfaces state extensions";
      uses hellos-extended_config;
    }

    /* RSVP authentication extension */
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
        + "rsvp:authentication/rsvp:config" {
      description
        "RSVP authentication all interfaces configuration extensions";
      uses authentication-extended_config;
    }
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
        + "rsvp:authentication/rsvp:state" {
      description
        "RSVP authentication all interfaces state extensions";
      uses authentication-extended_config;
    }

    /**
     * RSVP interface extensions
     */

    /* RSVP interface signaling extensions */
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
        "rsvp:interface/rsvp:config" {
      description
        "RSVP signaling interface configuration extensions";
      uses signaling-parameters-extended_config;
    }
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
        "rsvp:interface/rsvp:state" {
      description
        "RSVP signaling interface state extensions";
      uses signaling-parameters-extended_config;
    }

    /* RSVP refresh reduction extension */
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
        "rsvp:interface/rsvp:refresh-reduction/rsvp:config" {
      description
```

```
        "RSVP refresh-reduction interface configuration extensions";
      uses refresh-reduction-extended_config;
    }
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
        "rsvp:interface/rsvp:refresh-reduction/rsvp:state" {
      description
        "RSVP refresh-reduction interface state extensions";
      uses refresh-reduction-extended_config;
    }

    /* RSVP hellos extension */
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
        "rsvp:interface/rsvp:hellos/rsvp:config" {
      description
        "RSVP hello interface configuration extensions";
      uses hellos-extended_config;
    }
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
        "rsvp:interface/rsvp:hellos/rsvp:state" {
      description
        "RSVP hello interface state extensions";
      uses hellos-extended_config;
    }

    /* RSVP authentication extension */
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
        "rsvp:interface/rsvp:authentication/rsvp:config" {
      description
        "RSVP authentication interface configuration extensions";
      uses authentication-extended_config;
    }
    augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
        "rsvp:interface/rsvp:authentication/rsvp:state" {
      description
        "RSVP authentication interface state extensions";
      uses authentication-extended_config;
    }
  }
  <CODE ENDS>
```

Figure 5: RSVP extended YANG module

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

URI: urn:ietf:params:xml:ns:yang:ietf-rsvp-extended 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-rsvp namespace: urn:ietf:params:xml:ns:yang:ietf-rsvp prefix: ietf-rsvp reference: RFC3209

name: ietf-rsvp-extended namespace: urn:ietf:params:xml:ns:yang:ietf-rsvp-extended prefix: ietf-rsvp-extendeed reference: RFC3209

5. 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.

6. Acknowledgement

The authors would like to thank Lou Berger, for reviewing and providing valuable feedback on this document.

7. Contributors

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

8. References

8.1. Normative References

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

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

[RFC2747] Baker, F., Lindell, B., and M. Talwar, "RSVP Cryptographic Authentication", RFC 2747, DOI 10.17487/RFC2747, January 2000, <http://www.rfc-editor.org/info/rfc2747>.

[RFC2961] Berger, L., Gan, D., Swallow, G., Pan, P., Tommasi, F., and S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, DOI 10.17487/RFC2961, April 2001, <http://www.rfc-editor.org/info/rfc2961>.

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

[RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, DOI 10.17487/RFC3473, January 2003, <http://www.rfc-editor.org/info/rfc3473>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC5063] Satyanarayana, A., Ed. and R. Rahman, Ed., "Extensions to GMPLS Resource Reservation Protocol (RSVP) Graceful Restart", RFC 5063, DOI 10.17487/RFC5063, October 2007, <http://www.rfc-editor.org/info/rfc5063>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <http://www.rfc-editor.org/info/rfc8022>.

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-02 (work in progress), October 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.

Authors' Addresses

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Tarek Saad (editor)
Cisco Systems, Inc.

Email: tsaad@cisco.com

Rakesh Gandhi
Cisco Systems, Inc.

Email: rgandhi@cisco.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Igor Bryskin
Huawei Technologies

Email: Igor.Bryskin@huawei.com

Himanshu Shah
Ciena

Email: hshah@ciena.com

TEAS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2017

V. Beeram
Juniper Networks
T. Saad, Ed.
R. Gandhi
Cisco Systems, Inc.
X. Liu
Jabil
I. Bryskin
Huawei Technologies
H. Shah
Ciena

March 10, 2017

A YANG Data Model for RSVP-TE
draft-ietf-teas-yang-rsvp-te-00

Abstract

This document defines a YANG data model for the configuration and management of RSVP (Resource Reservation Protocol) to establish Traffic-Engineered (TE) Label-Switched Paths (LSPs) for MPLS (Multi-Protocol Label Switching) and other technologies.

The model defines a generic RSVP-TE module for signaling LSPs that is technology agnostic. The generic RSVP-TE module is to be augmented by technology specific RSVP-TE modules that define technology specific data. This document defines the augmentation for RSVP-TE MPLS LSPs model.

This model covers data for the configuration, operational state, remote procedural calls, and event notifications.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2017.

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 interfaces, such as CLI and programmatic APIs.

This document defines a generic YANG data model for configuring and managing RSVP-TE LSP(s) [RFC3209]. The RSVP-TE generic model augments the RSVP base and extended models defined in [I-D.ietf-teas-yang-rsvp], and adds TE extensions to the RSVP protocol [RFC2205] model configuration and state data. The technology specific RSVP-TE models augment the generic RSVP-TE model with additional technology specific parameters. For example, this document also defines the MPLS RSVP-TE model for configuring and managing MPLS RSVP TE LSP(s).

In addition to augmenting the RSVP YANG module, the modules defined in this document augment the TE Interfaces, Tunnels and LSP(s) YANG module defined in [I-D.ietf-teas-yang-te] to define additional parameters to enable signaling for RSVP-TE.

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]
te	ietf-te	this document
te-types	ietf-te-types	this document
te-mpls-types	ietf-te-mpls-types	this document
te-dev	ietf-te-device	this document
te-mpls	ietf-te-mpls	this document
te-sr-mpls	ietf-te-sr-mpls	this document

Table 1: Prefixes and corresponding YANG modules

2. Design Considerations

2.1. Module Hierarchy

The data pertaining to RSVP-TE in this document is divided into two modules: a technology agnostic RSVP-TE module that holds generic parameters for RSVP-TE applicable to all technologies, and a technology specific RSVP-TE module (e.g. for MPLS RSVP-TE) that holds parameters specific to the technology.

This document defines YANG data models for RSVP-TE, and RSVP-TE MPLS configuration, state, notification and RPCs. The relationship between the different modules is depicted in Figure 1.

2.2. Data Organization

The approach described in [I-D.openconfig-netmod-opstate] is adopted to represent data configuration for intended and applied configuration, and derived state data. Each container in the model holds a "config" and "state" sub-container.

The "config" sub-container holds the intended configurable parameters, while the state sub-container holds both applied configuration parameters as well as any derived state such as counters or statistics information. The pure state data (for example, protocol derived data) is also placed under the "state" sub-container.

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.

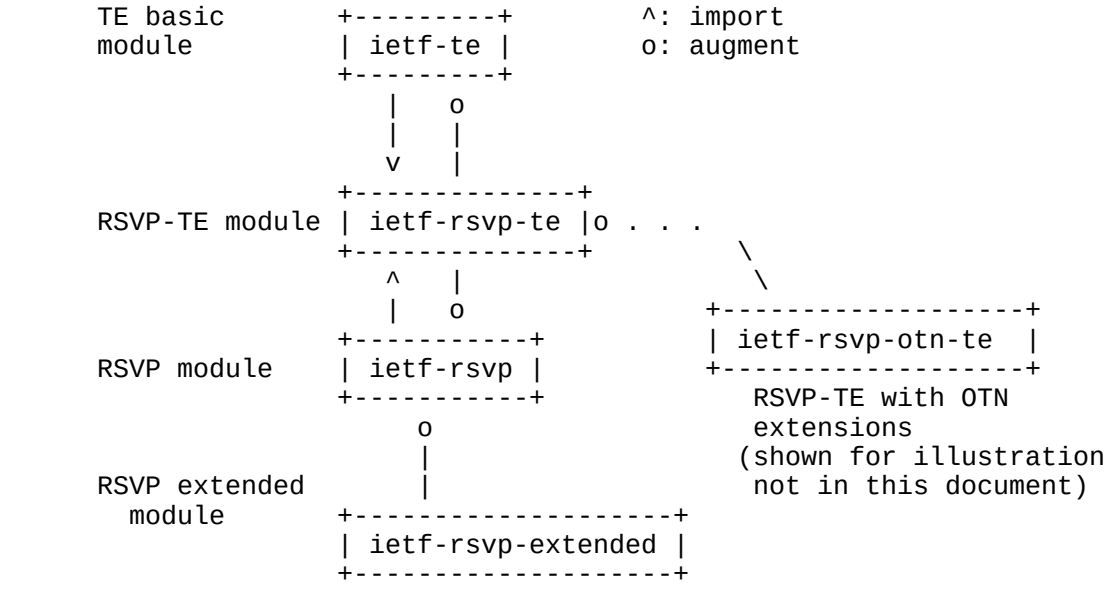


Figure 1: Relationship of RSVP and RSVP-TE modules with other protocol modules

2.3. RSVP-TE Generic Model

The RSVP-TE generic module augments the RSVP base and extended YANG modules defined in [I-D.ietf-teas-yang-rsvp] as well as the TE tunnels and interfaces module [I-D.ietf-teas-yang-te] to cover parameters specific to the configuration and management of RSVP-TE interfaces, tunnels and LSP(s).

2.3.1. Tree Diagram

There are three types of configuration and state data nodes in this module:

- those augmenting or extending the base RSVP module
- those augmenting or extending the base TE module
- those that are specific to the RSVP-TE module

Below is a YANG tree representation for data items defined in the RSVP-TE generic module:

```
module: ietf-rsvp-te
  augment /rt:routing/rt:control-plane-protocols/ +
```

```
           rt:control-plane-protocol/rsvp:rsvp/rsvp:globals:
      +--rw global-soft-preemption!
         +--rw config
         |  +--rw soft-preemption-timeout?   uint16
         +--rw state
            +--rw soft-preemption-timeout?   uint16
    augment /rt:routing/rt:control-plane-protocols/
           rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces:
      +--rw rsvp-te-interface-attributes
         +--rw config
         +--ro state
    augment /rt:routing/rt:control-plane-protocols/
           rt:control-plane-protocol/rsvp:rsvp/
           rsvp:interfaces/rsvp:interface:
      +--rw rsvp-te-interface-attributes
         +--rw config
         +--ro state
    augment /rt:routing/rt:control-plane-protocols/
           rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/
           rsvp:sessions/rsvp:session/rsvp:state/rsvp:psbs/rsvp:psb:
      +--ro tspec-average-rate?   rt-types:bandwidth-ieee-float32
      +--ro tspec-size?           rt-types:bandwidth-ieee-float32
      +--ro tspec-peak-rate?      rt-types:bandwidth-ieee-float32
      +--ro min-policed-unit?     uint32
      +--ro max-packet-size?      uint32
    augment /rt:routing/rt:control-plane-protocols/
           rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/
           rsvp:sessions/rsvp:session/rsvp:state/rsvp:rsbs/rsvp:rsb:
      +--ro fspec-average-rate?   rt-types:bandwidth-ieee-float32
      +--ro fspec-size?           rt-types:bandwidth-ieee-float32
      +--ro fspec-peak-rate?      rt-types:bandwidth-ieee-float32
      +--ro min-policed-unit?     uint32
      +--ro max-packet-size?      uint32
    augment /rt:routing/rt:control-plane-protocols/
           rt:control-plane-protocol/rsvp:rsvp/rsvp:neighbors:
    augment /te:te/te:tunnels/te:tunnel/te:config:
      +--rw lsp-signaled-name?           string
      +--rw local-recording-desired?     boolean
      +--rw se-style-desired?            boolean
      +--rw path-reevaluation-request?   boolean
      +--rw soft-preemption-desired?     boolean
      +--rw lsp-rerouting?               enumeration
      +--rw lsp-integrity-required?      boolean
      +--rw lsp-contiguous?              boolean
      +--rw lsp-stitching-desired?       boolean
      +--rw lsp-preplanned?              boolean
      +--rw lsp-oob-mapping?             boolean
      +--rw retry-timer?                 uint16
```

```
augment /te:te/te:tunnels/te:tunnel/te:state:
  +--ro lsp-signaled-name?           string
  +--ro local-recording-desired?     boolean
  +--ro se-style-desired?            boolean
  +--ro path-reevaluation-request?   boolean
  +--ro soft-preemption-desired?     boolean
  +--ro lsp-rerouting?               enumeration
  +--ro lsp-integrity-required?      boolean
  +--ro lsp-contiguous?              boolean
  +--ro lsp-stitching-desired?       boolean
  +--ro lsp-preplanned?              boolean
  +--ro lsp-oob-mapping?             boolean
  +--ro retry-timer?                 uint16
augment /te:te/te:lsps-state/te:lsp:
  +--ro associated-rsvp-session?
  -> /rt:routing/control-plane-protocols/control-plane-protocol/
     rsvp:rsvp/globals/sessions/session/local-index
  +--ro lsp-signaled-name?                  string
  +--ro local-recording-desired?            boolean
  +--ro se-style-desired?                   boolean
  +--ro path-reevaluation-request?          boolean
  +--ro soft-preemption-desired?            boolean
  +--ro lsp-rerouting?                      enumeration
  +--ro lsp-integrity-required?             boolean
  +--ro lsp-contiguous?                     boolean
  +--ro lsp-stitching-desired?              boolean
  +--ro lsp-preplanned?                     boolean
  +--ro lsp-oob-mapping?                    boolean
  +--ro explicit-route-objects
  |  +--ro incoming-explicit-route-hop* [index]
  |  |  +--ro index     -> ../config/index
  |  |  +--ro config
  |  |  |  +--ro index?            uint32
  |  |  |  +--ro (type)?
  |  |  |     +--:(ip-address)
  |  |  |     |  +--ro ip-address-hop
  |  |  |     |     +--ro address?    inet:ip-address
  |  |  |     |     +--ro hop-type?   te-hop-type
  |  |  |     +--:(as-number)
  |  |  |     |  +--ro as-number-hop
  |  |  |     |     +--ro as-number?   binary
  |  |  |     |     +--ro hop-type?    te-hop-type
  |  |  |     +--:(unnumbered-link)
  |  |  |     |  +--ro unnumbered-hop
  |  |  |     |     +--ro router-id?      inet:ip-address
  |  |  |     |     +--ro interface-id?   uint32
  |  |  |     |     +--ro hop-type?       te-hop-type
  |  |  |     +--:(label)
```

```
|  |  |     |  +--ro label-hop
|  |  |     |     +--ro value?   rt-types:generalized-label
|  |  |     +--:(sid)
|  |  |        +--ro sid-hop
|  |  |           +--ro sid?   rt-types:generalized-label
|  |  +--ro state
|  |     +--ro index?            uint32
|  |     +--ro (type)?
|  |        +--:(ip-address)
|  |        |  +--ro ip-address-hop
|  |        |     +--ro address?    inet:ip-address
|  |        |     +--ro hop-type?   te-hop-type
|  |        +--:(as-number)
|  |        |  +--ro as-number-hop
|  |        |     +--ro as-number?   binary
|  |        |     +--ro hop-type?    te-hop-type
|  |        +--:(unnumbered-link)
|  |        |  +--ro unnumbered-hop
|  |        |     +--ro router-id?      inet:ip-address
|  |        |     +--ro interface-id?   uint32
|  |        |     +--ro hop-type?       te-hop-type
|  |        +--:(label)
|  |        |  +--ro label-hop
|  |        |     +--ro value?   rt-types:generalized-label
|  |        +--:(sid)
|  |           +--ro sid-hop
|  |              +--ro sid?   rt-types:generalized-label
|  +--ro outgoing-explicit-route-hop* [index]
|     +--ro index     uint32
|     +--ro config
|     |  +--ro index?            uint32
|     |  +--ro (type)?
|     |     +--:(ip-address)
|     |     |  +--ro ip-address-hop
|     |     |     +--ro address?    inet:ip-address
|     |     |     +--ro hop-type?   te-hop-type
|     |     +--:(as-number)
|     |     |  +--ro as-number-hop
|     |     |     +--ro as-number?   binary
|     |     |     +--ro hop-type?    te-hop-type
|     |     +--:(unnumbered-link)
|     |     |  +--ro unnumbered-hop
|     |     |     +--ro router-id?      inet:ip-address
|     |     |     +--ro interface-id?   uint32
|     |     |     +--ro hop-type?       te-hop-type
|     |     +--:(label)
|     |     |  +--ro label-hop
|     |     |     +--ro value?   rt-types:generalized-label
```

```
|     |     +--:(sid)
|     |        +--ro sid-hop
|     |           +--ro sid?   rt-types:generalized-label
|     +--ro state
|        +--ro index?            uint32
|        +--ro (type)?
|           +--:(ip-address)
|           |  +--ro ip-address-hop
|           |     +--ro address?    inet:ip-address
|           |     +--ro hop-type?   te-hop-type
|           +--:(as-number)
|           |  +--ro as-number-hop
|           |     +--ro as-number?   binary
|           |     +--ro hop-type?    te-hop-type
|           +--:(unnumbered-link)
|           |  +--ro unnumbered-hop
|           |     +--ro router-id?      inet:ip-address
|           |     +--ro interface-id?   uint32
|           |     +--ro hop-type?       te-hop-type
|           +--:(label)
|           |  +--ro label-hop
|           |     +--ro value?   rt-types:generalized-label
|           +--:(sid)
|              +--ro sid-hop
|                 +--ro sid?   rt-types:generalized-label
|  +--ro incoming-record-route-subobjects
|  |  +--ro incoming-record-route-subobject* [index]
|  |     +--ro index           uint32
|  |     +--ro (type)?
|  |        +--:(ip-address)
|  |        |  +--ro ip-address?     inet:ip-address
|  |        |  +--ro ip-flags?       binary
|  |        +--:(unnumbered-link)
|  |        |  +--ro router-id?      inet:ip-address
|  |        |  +--ro interface-id?   uint32
|  |        +--:(label)
|  |           +--ro value?          rt-types:generalized-label
|  |           +--ro label-flags?    binary
|  +--ro outgoing-record-route-subobjects
|     +--ro outgoing-record-route-subobject* [index]
|        +--ro index           uint32
|        +--ro (type)?
|           +--:(ip-address)
|           |  +--ro ip-address?     inet:ip-address
|           |  +--ro ip-flags?       binary
|           +--:(unnumbered-link)
|           |  +--ro router-id?      inet:ip-address
|           |  +--ro interface-id?   uint32
```

```
               +--:(label)
                  +--ro value?          rt-types:generalized-label
                  +--ro label-flags?    binary
     augment /te:te/te-dev:interfaces/te-dev:interface:
```

Figure 2: RSVP-TE model Tree diagram

2.3.2. YANG Module

```
   <CODE BEGINS> file "ietf-rsvp-te@2017-03-10.yang"
   module ietf-rsvp-te {

     namespace "urn:ietf:params:xml:ns:yang:ietf-rsvp-te";

     prefix "rsvp-te";

     import ietf-rsvp {
       prefix rsvp;
     }

     import ietf-routing {
       prefix "rt";
     }

     import ietf-routing-types {
       prefix rt-types;
     }

     import ietf-te {
       prefix te;
     }

     import ietf-te-device {
       prefix te-dev;
     }

     /* Import TE generic types */
     import ietf-te-types {
       prefix te-types;
     }

     organization
       "IETF Traffic Engineering Architecture and Signaling (TEAS)
        Working Group";

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

```
        WG Chair: Lou Berger
                  <mailto:lberger@labn.net>

        WG Chair: Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Tarek Saad
                  <mailto:tsaad@cisco.com>

        Editor:   Rakesh Gandhi
                  <mailto:rgandhi@cisco.com>

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

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

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

        Editor:   Raqib Jones
                  <mailto:raqib@Brocade.com>

        Editor:   Bin Wen
                  <mailto:Bin_Wen@cable.comcast.com>";

     description
       "This module contains the RSVP-TE YANG generic data model.";

     revision "2017-03-10" {
       description "Latest revision to RSVP-TE generic YANG module";
       reference "RFC2205, RFC3209, etc.";
     }


     /**
      * RSVP-TE LSPs groupings.
      */

     grouping lsp-record-route-information_state {
       description "recorded route information grouping";
       container incoming-record-route-subobjects {
         description "RSVP recorded route object incoming information";
         list incoming-record-route-subobject {
```

```
          when "../../te:origin-type != 'ingress'" {
            description "Applicable on non-ingress LSPs only";
          }
          key "index";
          description
            "List of RSVP Path record-route objects";
          leaf index {
            type uint32;
            description "RRO subobject index";
          }
          uses te-types:record-route-subobject;
        }
      }
      container outgoing-record-route-subobjects {
        description "RSVP recorded route object outgoing information";
        list outgoing-record-route-subobject {
          when "../../te:origin-type != 'egress'" {
            description "Applicable on non-egress LSPs only";
          }
          key "index";
          description
            "List of RSVP Resv record-route objects";
          leaf index {
            type uint32;
            description "RRO subobject index";
          }
          uses te-types:record-route-subobject;
        }
      }
    }

    grouping lsp-explicit-route-information_state {
      description "RSVP-TE LSP explicit-route information";
      container explicit-route-objects {
        description "Explicit route object information";
        list incoming-explicit-route-hop {
          when "../../te:origin-type != 'ingress'" {
            description "Applicable on non-ingress LSPs only";
          }
          key "index";
          description
            "List of incoming RSVP Path explicit-route objects";
          leaf index {
            type leafref {
              path "../config/index";
            }
            description "ERO subobject index";
          }
```

```
          uses te-types:explicit-route-hop;
        }
        list outgoing-explicit-route-hop {
          when "../../te:origin-type != 'egress'" {
            description "Applicable on non-egress LSPs only";
          }
          key "index";
          description
            "List of outgoing RSVP Path explicit-route objects";
          leaf index {
            type uint32;
            description "ERO subobject index";
          }
          uses te-types:explicit-route-hop;
        }
      }
    }

    grouping lsp-attributes-flags_config {
      description
        "Configuration parameters relating to RSVP-TE LSP
        attribute flags";
      leaf lsp-rerouting {
        type enumeration {
          enum end-to-end-routing {
            description
              "End-to-end routing desired";
            reference "RFC4920, RFC5420";
          }
          enum boundary-rerouting {
            description
              "Boundary rerouting desired";
            reference "RFC4920, RFC5420";
          }
          enum segment-based-rerouting {
            description
              "Segment-based rerouting desired";
            reference "RFC4920, RFC5420";
          }
        }
        description "LSP rerouting types";
      }
      leaf lsp-integrity-required {
        type boolean;
        description "LSP integrity desired";
        reference "RFC4875";
      }
      leaf lsp-contiguous {
```

```
        type boolean;
        description "Contiguous LSP";
        reference "RFC5151";
      }
      leaf lsp-stitching-desired {
        type boolean;
        description "Stitched LSP";
        reference "RFC5150";
      }
      leaf lsp-preplanned {
        type boolean;
        description "Preplanned LSP";
        reference "RFC6001";
      }
      leaf lsp-oob-mapping {
        type boolean;
        description
          "Mapping is done out-of-band";
        reference "RFC6511";
      }
    }

    grouping lsp-session-attributes-obj-flags_config {
      description
        "Configuration parameters relating to RSVP-TE LSP
        session attribute flags";
      reference
        "RFC4859: Registry for RSVP-TE Session Flags";
      leaf local-recording-desired {
        type boolean;
        description "Path recording is desired.";
        reference "RFC3209";
      }
      leaf se-style-desired {
        type boolean;
        description "SE Style desired";
        reference "RFC3209";
      }
      leaf path-reevaluation-request {
        type boolean;
        description "Path re-evaluation request";
        reference "RFC4736";
      }
      leaf soft-preemption-desired {
        type boolean;
        description "Soft-preemption is desired";
        reference "RFC5712";
      }
```

```
    }

    grouping lsp-properties_config {
      description
        "Configuration parameters relating to RSVP-TE LSP
        session attribute flags";
      leaf lsp-signaled-name {
        type string;
        description
          "Sets the session name to use in the session
          attribute object.";
      }
      uses lsp-session-attributes-obj-flags_config;
      uses lsp-attributes-flags_config;
    }

    grouping tunnel-properties_config {
      description "RSVP-TE Tunnel properties grouping";
      leaf retry-timer {
        type uint16 {
          range 1..600;
        }
        units seconds;
        description
          "sets the time between attempts to establish the
           LSP";
      }
    }

    /*** End of RSVP-TE LSP groupings ***/

    /**
     * RSVP-TE generic global properties.
     */

    grouping global-soft-preemption_config {
      description
        "Configuration for global RSVP-TE soft preemption";
      leaf soft-preemption-timeout {
        type uint16 {
          range 0..300;
        }
        default 0;
        description
          "Timeout value for soft preemption to revert
          to hard preemption";
      }
    }
```

```
    grouping global-soft-preemption {
      description
        "Top level group for RSVP-TE soft-preemption";
      container global-soft-preemption {
        presence "Enables soft preemption on a node.";
        description
          "Top level container for RSVP-TE soft-preemption";
        container config {
          description
            "Configuration parameters relating to RSVP
            soft preemption support";
          uses global-soft-preemption_config;
        }
        container state {
          description "State parameters relating to RSVP
            soft preemption support";
          uses global-soft-preemption_config;
        }
      }
    }
    /*** End of RSVP-TE generic global properties. ***/

    /**
     * RSVP-TE interface generic groupings.
     */

    grouping rsvp-te-interface-attributes {
      description
        "Top level grouping for RSVP-TE interface properties.";
      container rsvp-te-interface-attributes {
        description
          "Top level container for RSVP-TE interface
          properties";
        container config {
          description
            "Configuration parameters relating to RSVP-TE
            bandwidth";
        }
        container state {
          config false;
          description
            "State information associated with RSVP-TE
            bandwidth";
        }
      }
    }
    /*** End of RSVP-TE generic groupings ***/
```

```
     /* RSVP-TE global properties */
     augment "/rt:routing/rt:control-plane-protocols/"
     + "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals" {
       description
         "RSVP-TE augmentation to RSVP globals";
         uses global-soft-preemption;
     }

     /* Linkage to the base RSVP all links */
     augment "/rt:routing/rt:control-plane-protocols/"
     + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces" {
       description
         "RSVP-TE generic data augmentation pertaining to interfaces";
       uses rsvp-te-interface-attributes;
     }

     /* Linkage to per RSVP interface */
     augment "/rt:routing/rt:control-plane-protocols/"
       + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
       "rsvp:interface" {
       description
         "RSVP-TE generic data augmentation pertaining to specific
          interface";
       uses rsvp-te-interface-attributes;
     }

     /* add augmentation for sessions and neighbors */
     augment "/rt:routing/rt:control-plane-protocols/"
     + "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/"
     + "rsvp:sessions/rsvp:session/rsvp:state/rsvp:psbs/rsvp:psb" {
       description
         "RSVP-TE generic data augmentation pertaining to session";
       /* To be added */
       leaf tspec-average-rate {
         type rt-types:bandwidth-ieee-float32;
         units "Bytes per second";
         description "Tspec Token Bucket Average Rate";
         reference "RFC2210: RSVP with INTSERV";
       }
       leaf tspec-size {
         type rt-types:bandwidth-ieee-float32;
         units "Bytes per second";
         description "Tspec Token Bucket Burst Rate";
         reference "RFC2210";
       }
       leaf tspec-peak-rate {
         type rt-types:bandwidth-ieee-float32;
         units "Bytes per second";
```

```
        description "Tspec Token Bucket Peak Data Rate";
        reference "RFC2210";
      }
      leaf min-policed-unit {
        type uint32;
        description "Tspec Minimum Policed Unit";
        reference "RFC2210";
      }
      leaf max-packet-size {
        type uint32;
        description "Tspec Maximum Packet Size";
        reference "RFC2210";
      }
    }
    augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/"
    + "rsvp:sessions/rsvp:session/rsvp:state/rsvp:rsbs/rsvp:rsb" {
      description
        "RSVP-TE generic data augmentation pertaining to session";
      leaf fspec-average-rate {
        type rt-types:bandwidth-ieee-float32;
        units "Bytes per second";
        description "Fspec Token Bucket Average Rate";
        reference "RFC2210";
      }
      leaf fspec-size {
        type rt-types:bandwidth-ieee-float32;
        units "Bytes per second";
        description "Fspec Token Bucket Burst Rate";
        reference "RFC2210";
      }
      leaf fspec-peak-rate {
        type rt-types:bandwidth-ieee-float32;
        units "Bytes per second";
        description "Fspec Token Bucket Peak Data Rate";
        reference "RFC2210";
      }
      leaf min-policed-unit {
        type uint32;
        description "Fspec Minimum Policed Unit";
        reference "RFC2210";
      }
      leaf max-packet-size {
        type uint32;
        description "Fspec Maximum Packet Size";
        reference "RFC2210";
      }
    }
```

```
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/rsvp:rsvp/rsvp:neighbors" {
  description
    "RSVP-TE generic data augmentation pertaining to neighbors";
  /* To be added */
}

/**
 * RSVP-TE generic augmentations of generic TE model.
 */

/* TE tunnel augmentation */
augment "/te:te/te:tunnels/te:tunnel/te:config" {
  when "/te:te/te:tunnels/te:tunnel" +
    "/te:p2p-primary-paths/te:p2p-primary-path/te:config" +
    "/te:path-setup-protocol = 'te-types:te-path-setup-rsvp'" {
    description
    "When the path signaling protocol is RSVP-TE ";
  }
  description
    "RSVP-TE generic data augmentation pertaining to TE tunnels";
  uses lsp-properties_config;
  uses tunnel-properties_config;
}

augment "/te:te/te:tunnels/te:tunnel/te:state" {
  when "/te:te/te:tunnels/te:tunnel" +
    "/te:p2p-primary-paths/te:p2p-primary-path/te:config" +
    "/te:path-setup-protocol = 'te-types:te-path-setup-rsvp'" {
    description
    "When the path signaling protocol is RSVP-TE ";
  }
  description
    "RSVP-TE generic data augmentation pertaining to TE tunnels";
  uses lsp-properties_config;
  uses tunnel-properties_config;
}

/* TE LSP augmentation */
augment "/te:te/te:lsps-state/te:lsp" {
  when "/te:te/te:lsps-state/te:lsp" +
    "/te:path-setup-protocol = 'te-types:te-path-setup-rsvp'" {
    description
    "When the signaling protocol is RSVP-TE ";
  }
  description
    "RSVP-TE generic data augmentation pertaining to specific TE
     LSP";
```

```
      leaf associated-rsvp-session {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
             + "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/"
             + "rsvp:sessions/rsvp:session/rsvp:local-index";
        }
        description
          "If the signalling protocol specified for this path is
          RSVP-TE, this leaf provides a reference to the associated
          session within the RSVP-TE protocol sessions list, such
          that details of the signaling can be retrieved.";
      }

      uses lsp-properties_config;
      uses lsp-explicit-route-information_state;
      uses lsp-record-route-information_state;
    }

    /* TE interface augmentation */
    augment "/te:te/te-dev:interfaces/te-dev:interface" {
      description
        "RSVP-TE generic data augmentation pertaining to specific TE
         interface";
    }
  }
  <CODE ENDS>
```

Figure 3: RSVP TE generic YANG module

2.4. RSVP-TE MPLS Model

The MPLS RSVP-TE YANG module augments the RSVP-TE generic module with parameters to configure and manage signaling of MPLS RSVP-TE LSPs. RSVP-TE YANG modules for other dataplane technologies (e.g. OTN or WDM) are outside the scope of this document and are defined in other documents.

2.4.1. Tree Diagram

The following are possible types of configuration and state data nodes in this module:

- those augmenting or extending the generic RSVP-TE module
- those augmenting or extending the TE module
- those that are specific to the RSVP-TE MPLS module

Below is a YANG tree representation for data items defined in the RSVP-TE MPLS module:

```
module: ietf-rsvp-te-mpls
  augment
  /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
  rsvp:rsvp:
    +--rw fast-reroute-local-revertive
       +--rw config
       |  +--rw rsvp-frr-local-revert-delay?   uint32
       +--ro state
          +--ro rsvp-frr-local-revert-delay?   uint32
  augment
  /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
  rsvp:rsvp/rsvp:interfaces:
  augment
  /rt:routing/rt:control-plane-protocols/
  rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/rsvp:interface:
  augment
  /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
  rsvp:rsvp/rsvp:globals/rsvp:sessions/rsvp:session/rsvp:state:
  augment
  /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
  rsvp:rsvp/rsvp:neighbors:
  augment /te:te/te:tunnels/te:tunnel/te:config:
    +--rw local-protection-desired?       empty
    +--rw bandwidth-protection-desired?   empty
    +--rw node-protection-desired?        empty
    +--rw non-php-desired?                empty
    +--rw entropy-label-cap?              empty
    +--rw oam-mep-entities-desired?       empty
    +--rw oam-mip-entities-desired?       empty
  augment /te:te/te:tunnels/te:tunnel/te:state:
    +--ro local-protection-desired?       empty
    +--ro bandwidth-protection-desired?   empty
    +--ro node-protection-desired?        empty
    +--ro non-php-desired?                empty
    +--ro entropy-label-cap?              empty
    +--ro oam-mep-entities-desired?       empty
    +--ro oam-mip-entities-desired?       empty
  augment /te:te/te:lsps-state/te:lsp:
    +--ro local-protection-desired?       empty
    +--ro bandwidth-protection-desired?   empty
    +--ro node-protection-desired?        empty
    +--ro non-php-desired?                empty
    +--ro entropy-label-cap?              empty
    +--ro oam-mep-entities-desired?       empty
    +--ro oam-mip-entities-desired?       empty
```

```
      +--ro backup-info
         +--ro backup-tunnel-name?         string
         +--ro backup-frr-on?              uint8
         +--ro backup-protected-lsp-num?   uint32
    augment /te:te/te-dev:interfaces/te-dev:interface:
      +--rw bandwidth-mpls-reservable
         +--rw config
         |  +--rw (bandwidth-value)?
         |  |  +--:(absolute)
         |  |  |  +--rw absolute-value?   uint32
         |  |  +--:(percentage)
         |  |     +--rw percent-value?    uint32
         |  +--rw (bc-model-type)?
         |     +--:(bc-model-rdm)
         |     |  +--rw bc-model-rdm
         |     |     +--rw bandwidth-mpls-constraints
         |     |        +--rw maximum-reservable?   uint32
         |     |        +--rw bc-value*             uint32
         |     +--:(bc-model-mam)
         |     |  +--rw bc-model-mam
         |     |     +--rw bandwidth-mpls-constraints
         |     |        +--rw maximum-reservable?   uint32
         |     |        +--rw bc-value*             uint32
         |     +--:(bc-model-mar)
         |        +--rw bc-model-mar
         |           +--rw bandwidth-mpls-constraints
         |              +--rw maximum-reservable?   uint32
         |              +--rw bc-value*             uint32
         +--ro state
            +--ro (bandwidth-value)?
            |  +--:(absolute)
            |  |  +--ro absolute-value?   uint32
            |  +--:(percentage)
            |     +--ro percent-value?    uint32
            +--ro (bc-model-type)?
               +--:(bc-model-rdm)
               |  +--ro bc-model-rdm
               |     +--ro bandwidth-mpls-constraints
               |        +--ro maximum-reservable?   uint32
               |        +--ro bc-value*             uint32
               +--:(bc-model-mam)
               |  +--ro bc-model-mam
               |     +--ro bandwidth-mpls-constraints
               |        +--ro maximum-reservable?   uint32
               |        +--ro bc-value*             uint32
               +--:(bc-model-mar)
                  +--ro bc-model-mar
                     +--ro bandwidth-mpls-constraints
```

```
                        +--ro maximum-reservable?   uint32
                        +--ro bc-value*             uint32
     augment /te:te/te-dev:interfaces/te-dev:interface:
       +--rw rsvp-te-frr-backups
          +--rw config
          |  +--rw (type)?
          |     +--:(static-tunnel)
          |     |  +--rw static-backups
          |     |     +--rw static-backup* [backup-tunnel-name]
          |     |        +--rw backup-tunnel-name
          -> ../config/backup-tunnel-name
          |     |        +--rw config
          |     |        |  +--rw backup-tunnel-name?
          -> /te:te/tunnels/tunnel/name
          |     |        +--ro state
          |     |           +--ro backup-tunnel-name?
          -> /te:te/tunnels/tunnel/name
          |     +--:(auto-tunnel)
          |        +--rw auto-backup-protection?         identityref
          |        +--rw auto-backup-path-computation?   identityref
          +--ro state
             +--ro (type)?
                +--:(static-tunnel)
                |  +--ro static-backups
                |     +--ro static-backup* [backup-tunnel-name]
                |        +--ro backup-tunnel-name
                -> ../config/backup-tunnel-name
                |        +--ro config
                |        |  +--ro backup-tunnel-name?
                -> /te:te/tunnels/tunnel/name
                |        +--ro state
                |           +--ro backup-tunnel-name?
                -> /te:te/tunnels/tunnel/name
                +--:(auto-tunnel)
                   +--ro auto-backup-protection?         identityref
                   +--ro auto-backup-path-computation?   identityref
```

Figure 4: RSVP-TE MPLS Tree diagram

2.4.2. YANG Module

```
<CODE BEGINS> file "ietf-rsvp-te@2017-03-10.yang"
module ietf-rsvp-te-mpls {

  namespace "urn:ietf:params:xml:ns:yang:ietf-rsvp-te-mpls";

  prefix "rsvp-te-mpls";
```

```
     import ietf-rsvp {
       prefix "rsvp";
     }

     import ietf-routing {
       prefix "rt";
     }

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

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

     import ietf-te {
       prefix "te";
     }

     import ietf-te-device {
       prefix "te-dev";
     }

     organization
       "IETF Traffic Engineering Architecture and Signaling (TEAS)
        Working Group";

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

        WG Chair: Lou Berger
                  <mailto:lberger@labn.net>

        WG Chair: Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Tarek Saad
                  <mailto:tsaad@cisco.com>

        Editor:   Rakesh Gandhi
                  <mailto:rgandhi@cisco.com>

        Editor:   Himanshu Shah
```

```
                <mailto:hshah@ciena.com>

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

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

        Editor:   Raqib Jones
                  <mailto:raqib@Brocade.com>

        Editor:   Bin Wen
                  <mailto:Bin_Wen@cable.comcast.com>";

    description
      "Latest update to MPLS RSVP-TE YANG data model.";

    revision "2017-03-10" {
      description "Update to MPLS RSVP-TE YANG initial revision.";
      reference "RFC3209, RFC6511, RFC6790, RFC7260, RFC4859, RFC4090";
    }

    /* RSVP-TE MPLS LSPs groupings */
    grouping lsp-attributes-flags-mpls_config {
      description
        "Configuration parameters relating to RSVP-TE MPLS LSP
        attribute flags";
      leaf non-php-desired {
        type empty;
        description
          "Non-PHP is desired";
        reference "RFC6511";
      }
      leaf entropy-label-cap {
        type empty;
        description "Entropy label capability";
        reference "RFC6790";
      }
      leaf oam-mep-entities-desired {
        type empty;
        description "OAM MEP entities desired";
        reference "RFC7260";
      }
      leaf oam-mip-entities-desired {
        type empty;
        description "OAM MIP entities desired";
        reference "RFC7260";
      }
```

```
    }

    grouping lsp-session-attributes-obj-flags-mpls_config {
      description
        "Configuration parameters relating to RSVP-TE MPLS LSP
         session attribute flags";
      reference
        "RFC4859: Registry for RSVP-TE Session Flags";
      leaf local-protection-desired {
        type empty;
        description "Fastreroute local protection is desired.";
        reference
          "RFC4859: Registry for RSVP-TE Session Flags";
      }
      leaf bandwidth-protection-desired {
        type empty;
        description
          "Request FRR bandwidth protection on LSRs if
          present.";
        reference "RFC4090";
      }
      leaf node-protection-desired {
        type empty;
        description
          "Request FRR node protection on LSRs if
          present.";
        reference "RFC4090";
      }
    }

    grouping tunnel-properties-mpls {
      description
        "Top level grouping for LSP properties.";
          uses lsp-session-attributes-obj-flags-mpls_config;
          uses lsp-attributes-flags-mpls_config;
    }

    grouping lsp-properties-mpls {
      description
        "Top level grouping for LSP properties.";
        uses lsp-session-attributes-obj-flags-mpls_config;
        uses lsp-attributes-flags-mpls_config;
    }
    /* End of RSVP-TE MPLS LSPs groupings */

    /* MPLS RSVP-TE interface groupings */
    grouping rsvp-te-interface_state {
      description
```

```
        "The RSVP-TE interface state grouping";
      leaf over-subscribed-bandwidth {
        type uint32;
        description
          "The amount of over-subscribed bandwidth on
          the interface";
      }
    }

    grouping rsvp-te-interface-softpreemption_state {
      description
        "The RSVP-TE interface preeemptions state grouping";
      container interface-softpreemption-state {
        description
          "The RSVP-TE interface preeemptions state grouping";
        leaf soft-preempted-bandwidth {
          type uint32;
          description
            "The amount of soft-preempted bandwidth on
            this interface";
        }
        list lsps {
          key
            "source destination tunnel-id lsp-id "+
            "extended-tunnel-id";
          description
            "List of LSPs that are soft-preempted";
          leaf source {
            type leafref {
              path "/te:te/te:lsps-state/te:lsp/"+
                   "te:source";
            }
            description
              "Tunnel sender address extracted from
              SENDER_TEMPLATE  object";
            reference "RFC3209";
          }
          leaf destination {
            type leafref {
              path "/te:te/te:lsps-state/te:lsp/"+
                   "te:destination";
            }
            description
              "Tunnel endpoint address extracted from
              SESSION object";
            reference "RFC3209";
          }
          leaf tunnel-id {
```

```
            type leafref {
              path "/te:te/te:lsps-state/te:lsp/"+
                "te:tunnel-id";
            }
            description
              "Tunnel identifier used in the SESSION
              that remains constant over the life
              of the tunnel.";
            reference "RFC3209";
          }
          leaf lsp-id {
            type leafref {
              path "/te:te/te:lsps-state/te:lsp/"+
                   "te:lsp-id";
            }
            description
              "Identifier used in the SENDER_TEMPLATE
              and the FILTER_SPEC that can be changed
              to allow a sender to share resources with
              itself.";
            reference "RFC3209";
          }
          leaf extended-tunnel-id {
            type leafref {
              path "/te:te/te:lsps-state/te:lsp/"+
                "te:extended-tunnel-id";
            }
            description
              "Extended Tunnel ID of the LSP.";
            reference "RFC3209";
          }
          leaf type {
            type leafref {
              path "/te:te/te:lsps-state/te:lsp/"+
                "te:type";
            }
            description "LSP type P2P or P2MP";
          }
        }
      }
    }

    grouping bandwidth-mpls-constraints {
      description "Bandwidth constraints.";
      container bandwidth-mpls-constraints {
        description
          "Holds the bandwidth constraints 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 constraint type";
        }
      }
    }

    grouping bandwidth-constraint-values {
      description
        "Packet bandwidth contraints values";
      choice value-type {
        description
          "Value representation";
        case percentages {
          container perc-values {
            uses bandwidth-mpls-constraints;
            description
              "Percentage values";
          }
        }
        case absolutes {
          container abs-values {
            uses bandwidth-mpls-constraints;
            description
              "Absolute values";
          }
        }
      }
    }

    grouping bandwidth-mpls-reservable_config {
      description
        "Interface bandwidth reservable configuration grouping";
      choice bandwidth-value {
        description "Reservable bandwidth configuration choice";
        case absolute {
          leaf absolute-value {
```

```
            type uint32;
            description "Absolute value of the bandwidth";
          }
        }
        case percentage {
          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-mpls-constraints;
          }
        }
        case bc-model-mam {
          container bc-model-mam {
            uses bandwidth-mpls-constraints;
            description
              "Maximum Allocation Model Bandwidth
              Constraints.";
          }
        }
        case bc-model-mar {
          container bc-model-mar {
            uses bandwidth-mpls-constraints;
            description
              "Maximum Allocation with Reservation Model
              Bandwidth Constraints.";
          }
        }
      }
    }

    grouping bandwidth-mpls-reservable {
      description
```

```
        "Packet reservable bandwidth";
      container bandwidth-mpls-reservable {
        description
          "Interface bandwidth reservable container";
        container config {
          description
            "Configuration parameters relating to
             interface bandwidth reservable properties";
          uses bandwidth-mpls-reservable_config;
        }
        container state {
          config false;
          description
            "State parameters relating to
             interface bandwidth reservable properties";
          uses bandwidth-mpls-reservable_config;
        }
      }
    }
    /* End of RSVP-TE interface groupings */

    /* RSVP-TE FRR groupings */
    grouping rsvp-te-frr-backups_config {
      description
        "Top level container for RSVP-TE FRR backup parameters";
      choice type {
        description
          "FRR backup tunnel type";
        case static-tunnel {
          container static-backups {
            description "List of static backups";
            list static-backup {
              key "backup-tunnel-name";
              description
                "List of static backup tunnels that
                protect the RSVP-TE interface.";
              leaf backup-tunnel-name {
                type leafref {
                  path "../config/backup-tunnel-name";
                }
                description "Backup tunnel name";
              }
              container config {
                description "Configuration for backup tunnels";
                leaf backup-tunnel-name {
                  type leafref {
                    path "/te:te/te:tunnels/te:tunnel/te:name";
                  }
```

```
                  description "FRR Backup tunnel name";
                }
              }
              container state {
                config false;
                description "State for backup tunnels";
                leaf backup-tunnel-name {
                  type leafref {
                    path "/te:te/te:tunnels/te:tunnel/te:name";
                  }
                  description "FRR Backup tunnel name";
                }
              }
            }
          }
        }
        case auto-tunnel {
          leaf auto-backup-protection {
            type identityref {
              base te-mpls-types:backup-protection-type;
            }
            default
              te-mpls-types:backup-protection-node-link;
            description
              "Describes whether the backup should offer
              protection against link, node, or either";
          }
          leaf auto-backup-path-computation {
            type identityref {
              base
                te-types:path-computation-srlg-type;
            }
            description
              "FRR backup computation type";
          }
        }
      }
    }

    grouping rsvp-te-frr-backups {
        description
          "RSVP-TE facility backup grouping";
      container rsvp-te-frr-backups {
        description
          "RSVP-TE facility backup properties";
        container config {
          description
            "Configuration parameters relating to
```

```
            RSVP-TE facility backups properties";
          uses rsvp-te-frr-backups_config;
        }
        container state {
          config false;
          description
            "State parameters relating to
             RSVP-TE facility backups properties";
          uses rsvp-te-frr-backups_config;
        }
      }
    }

    grouping lps-backup-info_state {
      description "Backup/bypass LSP related information";
      container backup-info {
        description
          "backup information";

        leaf backup-tunnel-name {
          type string;
          description
            "If an LSP has an FRR backup LSP that can protect it,
            this field identifies the tunnel name of the backup LSP.
            Otherwise, this field is empty.";
        }

        leaf backup-frr-on {
          type uint8;
          description
            "Whether currently this backup is carrying traffic";
        }

        leaf backup-protected-lsp-num {
          type uint32;
          description
            "Number of LSPs protected by this backup";
        }
      }
    }

    grouping fast-reroute-local-revertive_config {
      description "RSVP-TE FRR local revertive grouping";
      leaf rsvp-frr-local-revert-delay {
        type uint32;
        description
          "Time to wait after primary link is restored
          before node attempts local revertive
```

```
          procedures.";
      }
    }

    /*** End of RSVP-TE FRR backup information ***/

    grouping fast-reroute-local-revertive {
      description
        "Top level grouping for globals properties";
      container fast-reroute-local-revertive {
        description "RSVP-TE FRR local revertive container";
        container config {
          description
            "Configuration parameters relating to
             global MPLS RSVP-TE properties";
          uses fast-reroute-local-revertive_config;
        }
        container state {
          config false;
          description
            "State parameters relating to
             global MPLS RSVP-TE properties";
          uses fast-reroute-local-revertive_config;
        }
      }
    }

    /* RSVP-TE global properties */
    augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp" {
      description
        "RSVP-TE augmentation to RSVP globals";
      uses fast-reroute-local-revertive;
    }

    /* Linkage to the base RSVP all interfaces */
    augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces" {
      description
        "Augmentations for RSVP-TE MPLS all interfaces properties";
      /* To be added */
    }

    /* Linkage to per RSVP interface */
    augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
    "rsvp:interface" {
      description
```

```
        "Augmentations for RSVP-TE MPLS per interface properties";
      /* To be added */
    }

    /* add augmentation for sessions neighbors */
    augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/"
    + "rsvp:sessions/rsvp:session/rsvp:state" {
      description
        "Augmentations for RSVP-TE MPLS sessions";
      /* To be added */
    }

    augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:neighbors" {
      description
        "Augmentations for RSVP-TE MPLS neighbors properties";
      /* To be added */
    }

    /**
     * Augmentation to TE generic module
     */
    augment "/te:te/te:tunnels/te:tunnel/te:config" {
      description
        "Augmentations for RSVP-TE MPLS TE tunnel properties";
      uses tunnel-properties-mpls;
    }

    augment "/te:te/te:tunnels/te:tunnel/te:state" {
      description
        "Augmentations for RSVP-TE MPLS TE tunnel properties";
      uses tunnel-properties-mpls;
    }

    augment "/te:te/te:lsps-state/te:lsp" {
      when "/te:te/te:lsps-state/te:lsp" +
        "/te:path-setup-protocol = 'te-types:te-path-setup-rsvp'" {
        description
        "When the signaling protocol is RSVP-TE ";
      }
      description
        "RSVP-TE MPLS LSP state properties";
      uses lsp-properties-mpls;
      uses lps-backup-info_state;
    }

    augment "/te:te/te-dev:interfaces/te-dev:interface" {
```

```
      description
        "RSVP reservable bandwidth configuration properties";
      uses bandwidth-mpls-reservable;
    }

    augment "/te:te/te-dev:interfaces/te-dev:interface" {
      description
        "RSVP reservable bandwidth configuration properties";
      uses rsvp-te-frr-backups;
    }
  }
  <CODE ENDS>
```

Figure 5: RSVP TE MPLS YANG module

Figure 5 shows the YANG tree representation of the RSVP TE MPLS module that augments RSVP-TE module as well as RSVP and TE YANG modules.

3. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-rsvp-te XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-rsvp-te-mpls 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-rsvp namespace: urn:ietf:params:xml:ns:yang:ietf-rsvp-te prefix: ietf-rsvp reference: RFC3209

name: ietf-rsvp-te namespace: urn:ietf:params:xml:ns:yang:ietf-rsvp-te-mpls prefix: ietf-rsvp-te reference: RFC3209

4. 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.

5. Acknowledgement

The authors would like to thank Lou Berger for reviewing and providing valuable feedback on this document.

6. Contributors

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

7. References

7.1. Normative References

[I-D.ietf-teas-yang-rsvp]
Beeram, V., Saad, T., Gandhi, R., Liu, X., Shah, H., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Resource Reservation Protocol (RSVP)", draft-ietf-teas-yang-rsvp-06 (work in progress), October 2016.

[I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., Bryskin, I., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-05 (work in progress), October 2016.

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

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

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

7.2. 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-02 (work in progress), October 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.

Authors' Addresses

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Tarek Saad (editor)
Cisco Systems, Inc.

Email: tsaad@cisco.com

Rakesh Gandhi
Cisco Systems, Inc.

Email: rgandhi@cisco.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Igor Bryskin
Huawei Technologies

Email: Igor.Bryskin@huawei.com

Himanshu Shah
Ciena

Email: hshah@ciena.com

TEAS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 11, 2017

T. Saad, Ed.
R. Gandhi
Cisco Systems Inc
X. Liu
Jabil
V. Beeram
Juniper Networks
H. Shah
Ciena
I. Bryskin
Huawei Technologies
March 10, 2017

A YANG Data Model for Traffic Engineering Tunnels and Interfaces
draft-ietf-teas-yang-te-06

Abstract

This document defines a YANG data model for the configuration and management of Traffic Engineering (TE) interfaces, tunnels and Label Switched Paths (LSPs). The model is divided into YANG modules that classify data into generic, device-specific, technology agnostic, and technology-specific elements. The model also includes module(s) that contain reusable TE data types and data groupings.

This model covers data for configuration, operational state, remote procedural calls, and event notifications.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2017.

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. RESTCONF [RFC8040]) 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 interfaces, such as CLI and programmatic APIs.

This document describes the YANG data models for TE Tunnels, Label Switched Paths (LSPs) and TE interfaces that cover data applicable to generic or device-independent, device-specific, Multiprotocol Label Switching (MPLS) technology specific, and Segment Routing (SR) TE technology. It also describes helper modules that define TE grouping(s) and data types that can be imported by other modules.

The document defines the high-level relationship between the modules defined in this document, as well as other external protocol modules. It is expected other data plane technology model(s) will augment the TE generic model. Also, the TE generic model does not include any data specific to a signaling protocol. It is expected YANG models for TE signaling protocols, such as RSVP-TE ([RFC3209], [RFC3473]), or Segment-Routing TE (SR-TE) will augment the TE generic module.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in 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]
te	ietf-te	this document
te-types	ietf-te-types	this document
te-mpls-types	ietf-te-mpls-types	this document
te-dev	ietf-te-device	this document
te-mpls	ietf-te-mpls	this document
te-sr-mpls	ietf-te-sr-mpls	this document

Table 1: Prefixes and corresponding YANG modules

1.4. Open Issues and Next Steps

This section describes the number of open issues that are under consideration. As issues are resolved, this section will be updated to reflect this and be left there for reference. It is expected that all the issues in this section will be addressed before the document will be ready for final publication.

1.4.1. TE Technology Models

This document describes the generic TE YANG data model that is independent of any dataplane technology. One of the design objectives is to allow specific data plane technologies models to reuse the generic TE data model and possibly augment it with technology specific data model(s). There are multiple options being considered to achieve this:

- The generic TE model, including the lists of TE tunnels, LSPs, and interfaces can be defined and rooted at the top of the YANG tree. Specific leaf(s) under the TE tunnel, LSP, or interface, in this case, can identify the specific technology layer that it belongs to. This approach implies a single list for each of TE tunnel(s), LSP(s), and interface(s) in the model carries elements of different technology layers.

- An instance of the generic TE YANG model can be mounted in the YANG tree once for each TE technology layer(s). This approach provides separation of elements belonging to different technology layers into separate lists per layer in the data model. For example, the proposal in [I-D.clemm-netmod-mount] allows for this capability by "mounting" the YANG data model at a specific target.

- The generic TE data node(s) and TE list(s) for tunnels, LSPs, and interfaces are defined as grouping(s) in a separate module. The specific technology layer imports the generic TE groupings and uses them their respective technology specific module.

This revision of the model leverages the LSP encoding type of a tunnel (and interfaces) to identify the specific technology associated with the a TE interfaces, tunnel(s) and the LSP(s). For example, for an MPLS TE LSP, the LSP encoding type is assumed to be "lsp-encoding-packet".

Finally, the TE generic model does not include any signaling protocol data. It is expected that TE signaling protocol module(s) will be defined in other document(s) that will cover the RSVP-TE ([RFC3209], [RFC3473]), and Segment-Routing TE (SR-TE) model and that augment the TE generic model.

1.4.2. State Data Organization

Pure state data (for example, ephemeral or protocol derived state objects) can be modeled using one of the options below:

- Contained inside a read-write container, in a "state" sub-container, as shown in Figure 3

- Contained inside a separate read-only container, for example a lsps-state container

The first option allows for placing configuration data in the read-write "config" sub-container, and by placing state data under the read-only "state" sub-container of the parent container. However, when using approach for ephemeral or purely derived state (e.g. auto tunnels), and since in this case the state sub-container hangs off a read-write parent container, it will be possible to delete or modify 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 for ephemeral or state derived tunnels. Further discussions on this topic are expected to close on the best choice to adopt.

2. Model Overview

The data model defined in this document covers the core TE features that are commonly supported across different vendor implementations. The support of extended or vendor specific TE feature(s) are expected to be in augmentations to the data models defined in this document.

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 statistics 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. Module(s) Relationship

The TE generic model defined in "ietf-te.yang" covers the building blocks that are device independent and agnostic of any specific technology or control plane instances. The TE device model defined in "ietf-te-device.yang" augments the TE generic model and covers data that is specific to a device - for example, attributes of TE interfaces, or TE timers that are local to a TE node.

The TE data relevant to a specific instantiations of data plane technology exists in a separate YANG module(s) that augment the TE generic model. For example, the MPLS-TE module "ietf-te-mpls.yang" is defined in Figure 10 and augments the TE generic model as shown in Figure 1. Similarly, the module "ietf-te-sr-mpls.yang" models the Segment Routing (SR) TE specific data and augments the TE generic and MPLS-TE model(s).

The TE data relevant to a TE specific signaling protocol instantiation is outside the scope and is covered in other documents. For example, the RSVP-TE [RFC3209] YANG model augmentation of the TE model is covered in [I-D.ietf-teas-yang-rsvp], and other signaling protocol model(s) (e.g. for Segment-Routing TE) are expected to also augment the TE generic model.

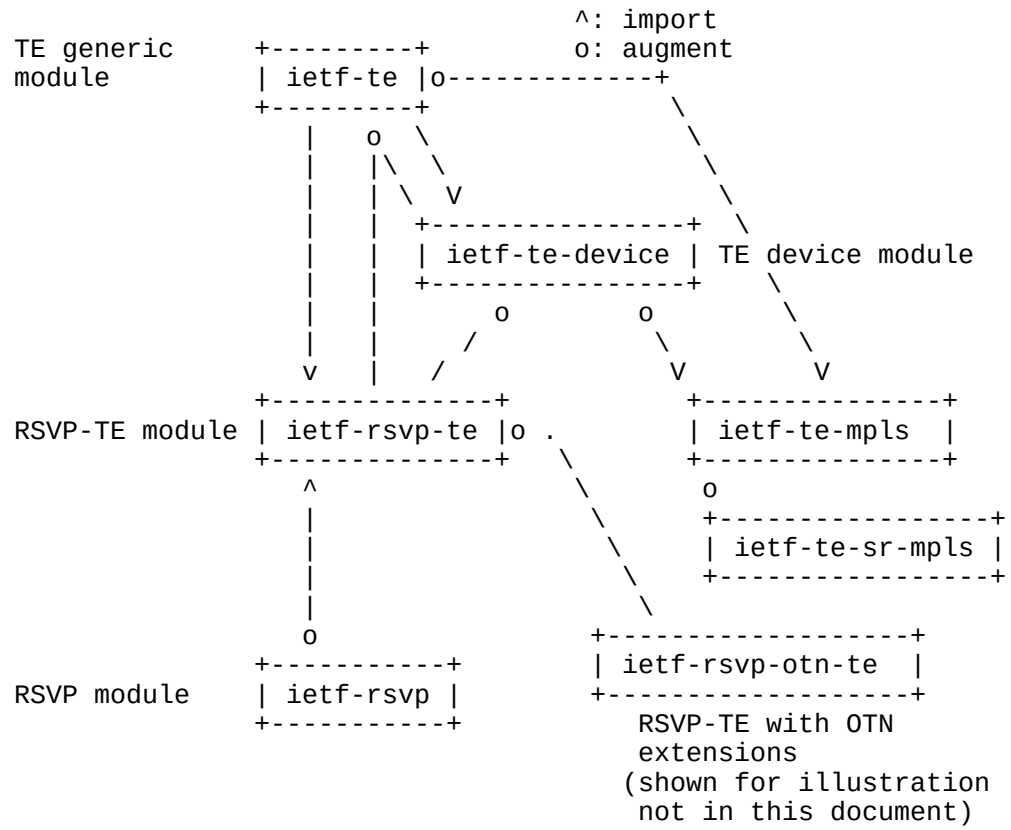


Figure 1: Relationship of TE module(s) with other signaling protocol modules

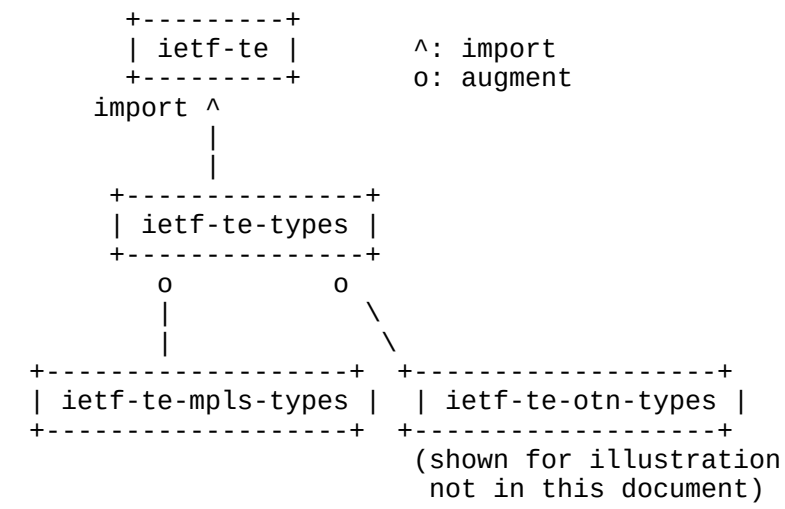


Figure 2: Relationship between generic and technology specific TE types modules

2.2. Design Considerations

The following considerations with respect data organization are taken into account:

- reusable data elements are grouped into separate TE types module(s) that can be readily imported by other modules whenever needed
- reusable TE data types that are data plane independent are grouped in the TE generic types module "ietf-te-types.yang"
- reusable TE data elements that are data plane specific (e.g. packet MPLS or switching technologies as defined in [RFC3473]) are expected to be grouped in a technology- specific types module, e.g. "ietf-te-mpls-types.yang". It is expected that technology specific types will augment TE generic types as shown in Figure 2
- The TE generic model contains device independent data and can be used to model data off a device (e.g. on a controller). The TE data that is device-specific are grouped in a separate module as shown in Figure 1.
- 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.

2.3. Optional Features

Optional features that are beyond the base TE model are left to the specific vendor to decide support using vendor model augmentation and/or using feature checks.

This model declares a number of TE functions as features (such as P2MP-TE, soft-preemption etc.).

2.4. 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 of a list (e.g. a tunnel, interface or path).

3. TE Generic Model Organization

The TE generic model covers configuration, state, RPCs, and notifications data pertaining to TE global parameters, interfaces, tunnels and LSPs parameters that are device independent.

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:

- State corresponding to applied configuration
- State corresponding to derived state, counters, stats, etc.
- State corresponding to ephemeral data (e.g. LSPs, etc.)

Data for the first two categories are contained under the respective "state" sub-container of the intended (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 tunnels
            +-- rw config
               <<intended configuration>>
            .
            +-- ro state
               <<applied configuration>>
               <<derived state associated with the tunnel>>
         .
         .
rpcs:
   +---x globals-rpc
   +---x tunnels-rpc
notifications:
   +---n globals-notif
   +---n tunnels-notif
```

Figure 3: TE generic 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:

- Table of named SRLG mappings
- Table of named (extended) administrative groups mappings
- Table of named explicit paths to be referenced by TE tunnels
- Table of named path-constraints sets
- Auto-bandwidth global parameters
- TE diff-serve TE-class maps

- o System-wide capabilities for LSP reoptimization (included in the TE device model)
  - * Reoptimization timers (periodic interval, LSP installation and cleanup)
- o System-wide capabilities for TE state flooding (included in the TE device model)
  - * Periodic flooding interval
- o Global capabilities that affect the originating, traversing and terminating LSPs. For example:
  - * Path selection parameters (e.g. metric to optimize, etc.)
  - * Path or segment protection parameters

The approach described in [I-D.openconfig-netmod-opstate] is utilized 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)

3.2. Interfaces Configuration and State Data

This branch of the model covers configuration and state data items corresponding to TE interfaces that are present on a specific device. A new module is introduced that holds the TE device specific properties.

Examples of TE interface properties 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

The state corresponding to the TE interfaces applied configuration, protocol derived state, and stats and counters all fall under the interface "state" sub-container as shown in Figure 4 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 4: 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

o Statistics: state counters, flooding counters, admission counters (accepted/rejected), preemption counters

o 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 utilized for the inclusion of operational and statistical data as shown in Figure 5.

```
module: ietf-te
   +--rw te!
      +--rw tunnels
         .
         +-- rw config
            <<intended configuration>>
         .
         +-- ro state
            <<applied configuration>>
            <<derived state associated with the tunnel>>
```

Figure 5: TE interface state tree

Examples of tunnel configuration date for TE tunnels:

- Name and type (e.g. P2P, P2MP) of the TE tunnel
- Admin-state
- Set of primary and corresponding secondary paths
- Routing usage (auto-route announce, forwarding adjacency)
- Policy based routing (PBR) parameters

3.3.1. Tunnel Compute-Only Mode

By default, a configured TE tunnel is provisioned so it can carry traffic as soon as a valid path is computed and an LSP instantiated in the network. In other cases, a TE tunnel may be provisioned for computed path reporting purposes without the need to instantiate an LSP or commit resources in the network. In such a case, a tunnel configuration in "compute-only" mode to distinguish it from default tunnel behavior.

A "compute-only" TE tunnel is configured as a usual TE tunnel with associated path constraint(s) and properties on a device or controller. The device or controller is expected to compute the feasible path(s) subject to configured constraints for of "compute-only" tunnel and reflect the computed path(s) in the LSP(s) Record-Route Object (RRO) list. A client may query "on-demand" the "compute-only" TE tunnel computed path(s) properties by querying the state of the tunnel. Alternatively, the client can subscribe on the

"compute-only" TE tunnel to be notified of computed path(s) and whenever it changes.

3.3.2. Tunnel Hierarchical Link Endpoint

TE LSPs can be set up in MPLS or Generalized MPLS (GMPLS) networks to be used to form links to carry traffic in in other (client) networks [RFC6107]. In this case, the model introduces the TE tunnel hierarchical link endpoint parameters to identify the specific link in the client layer that the TE tunnel is associated with.

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). In the model, the nodes holding LSPs data exist in the read-only lsps-state list as show in Figure 6.

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:

- 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:

- Clear TE statistics for all or for individual TE interfaces
- 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
- Trigger the tear and setup of existing tunnels or LSPs.

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.

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

Figure Figure 6 below shows the tree diagram of the YANG model defined in modules: ietf-te.yang, ietf-te-device.yang, ietf-te-mpls.yang, and ietf-te-sr.yang.

```
module: ietf-te
    +--rw te!
       +--rw globals
       |  +--rw named-admin-groups
       |  |  +--rw named-admin-group* [name]
       {te-types:extended-admin-groups,
       te-types:named-extended-admin-groups}?
       |  |     +--rw name      -> ../config/name
       |  |     +--rw config
       |  |     |  +--rw name?           string
```

```
|  |     |  +--rw bit-position?   uint32
|  |     +--ro state
|  |        +--ro name?           string
|  |        +--ro bit-position?   uint32
|  +--rw named-srlgs
|  |  +--rw named-srlg* [name] {te-types:named-srlg-groups}?
|  |     +--rw name      -> ../config/name
|  |     +--rw config
|  |     |  +--rw name?    string
|  |     |  +--rw group?   te-types:srlg
|  |     |  +--rw cost?    uint32
|  |     +--ro state
|  |        +--ro name?    string
|  |        +--ro group?   te-types:srlg
|  |        +--ro cost?    uint32
|  +--rw named-explicit-paths
|  |  +--rw named-explicit-path* [name]
|  |     +--rw name      -> ../config/name
|  |     +--rw config
|  |     |  +--rw name?                      string
|  |     |  +--rw explicit-route-objects
|  |     |     +--rw explicit-route-object* [index]
|  |     |        +--rw index                  -> ../config/index
|  |     |        +--rw explicit-route-usage?   identityref
|  |     |        +--rw config
|  |     |        |  +--rw index?            uint32
|  |     |        |  +--rw (type)?
|  |     |        |     +--:(ip-address)
|  |     |        |     |  +--rw ip-address-hop
|  |     |        |     |     +--rw address?    inet:ip-address
|  |     |        |     |     +--rw hop-type?   te-hop-type
|  |     |        |     +--:(as-number)
|  |     |        |     |  +--rw as-number-hop
|  |     |        |     |     +--rw as-number?   binary
|  |     |        |     |     +--rw hop-type?    te-hop-type
|  |     |        |     +--:(unnumbered-link)
|  |     |        |     |  +--rw unnumbered-hop
|  |     |        |     |     +--rw router-id?    inet:ip-address
|  |     |        |     |     +--rw interface-id?   uint32
|  |     |        |     |     +--rw hop-type?       te-hop-type
|  |     |        |     +--:(label)
|  |     |        |     |  +--rw label-hop
|  |     |        |     |     +--rw value?
|  |     |        |     |           rt-types:generalized-label
|  |     |        |     +--:(sid)
|  |     |        |        +--rw sid-hop
|  |     |        |           +--rw sid?
|  |     |        |                 rt-types:generalized-label
```

```
        |  |     |        +--ro state
        |  |     |           +--ro index?           uint32
        |  |     |           +--ro (type)?
        |  |     |              +--:(ip-address)
        |  |     |              |  +--ro ip-address-hop
        |  |     |              |     +--ro address?    inet:ip-address
        |  |     |              |     +--ro hop-type?   te-hop-type
        |  |     |              +--:(as-number)
        |  |     |              |  +--ro as-number-hop
        |  |     |              |     +--ro as-number?   binary
        |  |     |              |     +--ro hop-type?    te-hop-type
        |  |     |              +--:(unnumbered-link)
        |  |     |              |  +--ro unnumbered-hop
        |  |     |              |     +--ro router-id? inet:ip-address
        |  |     |              |     +--ro interface-id?   uint32
        |  |     |              |     +--ro hop-type?       te-hop-type
        |  |     |              +--:(label)
        |  |     |              |  +--ro label-hop
        |  |     |              |     +--ro value?
        |  |     |              |           rt-types:generalized-label
        |  |     |              +--:(sid)
        |  |     |                 +--ro sid-hop
        |  |     |                    +--ro sid?
        |  |     |                          rt-types:generalized-label
        |  |     +--ro state
        |  |        +--ro name?                    string
        |  |        +--ro explicit-route-objects
        |  |           +--ro explicit-route-object* [index]
        |  |              +--ro index               -> ../config/index
        |  |              +--ro explicit-route-usage?   identityref
        |  |              +--ro config
        |  |              |  +--ro index?            uint32
        |  |              |  +--ro (type)?
        |  |              |     +--:(ip-address)
        |  |              |     |  +--ro ip-address-hop
        |  |              |     |     +--ro address?    inet:ip-address
        |  |              |     |     +--ro hop-type?   te-hop-type
        |  |              |     +--:(as-number)
        |  |              |     |  +--ro as-number-hop
        |  |              |     |     +--ro as-number?   binary
        |  |              |     |     +--ro hop-type?    te-hop-type
        |  |              |     +--:(unnumbered-link)
        |  |              |     |  +--ro unnumbered-hop
        |  |              |     |     +--ro router-id?   inet:ip-address
        |  |              |     |     +--ro interface-id?   uint32
        |  |              |     |     +--ro hop-type?       te-hop-type
        |  |              |     +--:(label)
        |  |              |     |  +--ro label-hop
```

```
        |  |              |     |     +--ro value?
        |  |              |     |           rt-types:generalized-label
        |  |              |     +--:(sid)
        |  |              |        +--ro sid-hop
        |  |              |           +--ro sid?
        |  |              |                 rt-types:generalized-label
        |  |              +--ro state
        |  |                 +--ro index?            uint32
        |  |                 +--ro (type)?
        |  |                    +--:(ip-address)
        |  |                    |  +--ro ip-address-hop
        |  |                    |     +--ro address?    inet:ip-address
        |  |                    |     +--ro hop-type?   te-hop-type
        |  |                    +--:(as-number)
        |  |                    |  +--ro as-number-hop
        |  |                    |     +--ro as-number?   binary
        |  |                    |     +--ro hop-type?    te-hop-type
        |  |                    +--:(unnumbered-link)
        |  |                    |  +--ro unnumbered-hop
        |  |                    |     +--ro router-id?    inet:ip-address
        |  |                    |     +--ro interface-id?   uint32
        |  |                    |     +--ro hop-type?       te-hop-type
        |  |                    +--:(label)
        |  |                    |  +--ro label-hop
        |  |                    |     +--ro value?
        |  |                    |           rt-types:generalized-label
        |  |                    +--:(sid)
        |  |                       +--ro sid-hop
        |  |                          +--ro sid?
        |  |                                rt-types:generalized-label
        |  +--rw named-path-constraints
        |  |  +--rw named-path-constraint* [name]
        |  |     +--rw name                  -> ../config/name
        |  |     +--rw config
        |  |     |  +--rw name?                      string
        |  |     |  +--rw topology-id?             te-types:te-topology-id
        |  |     |  +--rw cost-limit?                uint32
        |  |     |  +--rw hop-limit?                 uint8
        |  |     |  +--rw metric-type?               identityref
        |  |     |  +--rw tiebreaker-type?           identityref
        |  |     |  +--rw ignore-overload?           boolean
        |  |     |  +--rw setup-priority?            uint8
        |  |     |  +--rw hold-priority?             uint8
        |  |     |  +--rw 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 path-srlgs
        |  |     |  |  +--rw (style)?
        |  |     |  |     +--:(values)
        |  |     |  |     |  +--rw usage?         identityref
        |  |     |  |     |  +--rw values*        te-types:srlg
        |  |     |  |     +--:(named)
        |  |     |  |        +--rw constraints* [usage]
        |  |     |  |           +--rw usage         identityref
        |  |     |  |           +--rw constraint
        |  |     |  |              +--rw srlg-names* [name]
        |  |     |  |                 +--rw name    string
        |  |     |  +--rw explicit-route-objects
        |  |     |  |  +--rw explicit-route-object* [index]
        |  |     |  |     +--rw index                -> ../config/index
        |  |     |  |     +--rw explicit-route-usage?   identityref
        |  |     |  |     +--rw config
        |  |     |  |     |  +--rw index?            uint32
        |  |     |  |     |  +--rw (type)?
        |  |     |  |     |     +--:(ip-address)
        |  |     |  |     |     |  +--rw ip-address-hop
        |  |     |  |     |     |     +--rw address?    inet:ip-address
        |  |     |  |     |     |     +--rw hop-type?   te-hop-type
        |  |     |  |     |     +--:(as-number)
        |  |     |  |     |     |  +--rw as-number-hop
        |  |     |  |     |     |     +--rw as-number?   binary
        |  |     |  |     |     |     +--rw hop-type?    te-hop-type
        |  |     |  |     |     +--:(unnumbered-link)
        |  |     |  |     |     |  +--rw unnumbered-hop
        |  |     |  |     |     |     +--rw router-id?    inet:ip-address
        |  |     |  |     |     |     +--rw interface-id?   uint32
        |  |     |  |     |     |     +--rw hop-type?       te-hop-type
        |  |     |  |     |     +--:(label)
        |  |     |  |     |     |  +--rw label-hop
        |  |     |  |     |     |     +--rw value?
        |  |     |  |     |     |           rt-types:generalized-label
        |  |     |  |     |     +--:(sid)
        |  |     |  |     |        +--rw sid-hop
        |  |     |  |     |           +--rw sid?
        |  |     |  |     |                 rt-types:generalized-label
        |  |     |  |     +--ro state
        |  |     |  |        +--ro index?            uint32
        |  |     |  |        +--ro (type)?
```

```
        |  |     |  |           +--:(ip-address)
        |  |     |  |           |  +--ro ip-address-hop
        |  |     |  |           |     +--ro address?    inet:ip-address
        |  |     |  |           |     +--ro hop-type?   te-hop-type
        |  |     |  |           +--:(as-number)
        |  |     |  |           |  +--ro as-number-hop
        |  |     |  |           |     +--ro as-number?   binary
        |  |     |  |           |     +--ro hop-type?    te-hop-type
        |  |     |  |           +--:(unnumbered-link)
        |  |     |  |           |  +--ro unnumbered-hop
        |  |     |  |           |     +--ro router-id?    inet:ip-address
        |  |     |  |           |     +--ro interface-id?   uint32
        |  |     |  |           |     +--ro hop-type?       te-hop-type
        |  |     |  |           +--:(label)
        |  |     |  |           |  +--ro label-hop
        |  |     |  |           |     +--ro value?
                                          rt-types:generalized-label
        |  |     |  |           +--:(sid)
        |  |     |  |              +--ro sid-hop
        |  |     |  |                 +--ro sid?
                                          rt-types:generalized-label
        |  |     |  +--rw bandwidth
        |  |     |     +--rw config
        |  |     |     |  +--rw specification-type?
                                 te-mpls-types:te-bandwidth-type
        |  |     |     |  +--rw set-bandwidth?
                                 te-mpls-types:bandwidth-kbps
        |  |     |     |  +--rw class-type?
                                 te-types:te-ds-class
        |  |     |     +--ro state
        |  |     |        +--ro specification-type?
                                 te-mpls-types:te-bandwidth-type
        |  |     |        +--ro set-bandwidth?
                                 te-mpls-types:bandwidth-kbps
        |  |     |        +--ro class-type?
                                 te-types:te-ds-class
        |  |     |        +--ro signaled-bandwidth?
                                 te-mpls-types:bandwidth-kbps
        |  |     +--ro state
        |  |     |  +--ro name?                      string
        |  |     |  +--ro topology-id?             te-types:te-topology-id
        |  |     |  +--ro cost-limit?                uint32
        |  |     |  +--ro hop-limit?                 uint8
        |  |     |  +--ro metric-type?               identityref
        |  |     |  +--ro tiebreaker-type?           identityref
        |  |     |  +--ro ignore-overload?           boolean
        |  |     |  +--ro setup-priority?            uint8
        |  |     |  +--ro hold-priority?             uint8
```

```
        |  |     |  +--ro 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 path-srlgs
        |  |     |  |  +--ro (style)?
        |  |     |  |     +--:(values)
        |  |     |  |     |  +--ro usage?         identityref
        |  |     |  |     |  +--ro values*        te-types:srlg
        |  |     |  |     +--:(named)
        |  |     |  |        +--ro constraints* [usage]
        |  |     |  |           +--ro usage         identityref
        |  |     |  |           +--ro constraint
        |  |     |  |              +--ro srlg-names* [name]
        |  |     |  |                 +--ro name    string
        |  |     |  +--ro explicit-route-objects
        |  |     |  |  +--ro explicit-route-object* [index]
        |  |     |  |     +--ro index               -> ../config/index
        |  |     |  |     +--ro explicit-route-usage?   identityref
        |  |     |  |     +--ro config
        |  |     |  |     |  +--ro index?            uint32
        |  |     |  |     |  +--ro (type)?
        |  |     |  |     |     +--:(ip-address)
        |  |     |  |     |     |  +--ro ip-address-hop
        |  |     |  |     |     |     +--ro address?    inet:ip-address
        |  |     |  |     |     |     +--ro hop-type?   te-hop-type
        |  |     |  |     |     +--:(as-number)
        |  |     |  |     |     |  +--ro as-number-hop
        |  |     |  |     |     |     +--ro as-number?   binary
        |  |     |  |     |     |     +--ro hop-type?    te-hop-type
        |  |     |  |     |     +--:(unnumbered-link)
        |  |     |  |     |     |  +--ro unnumbered-hop
        |  |     |  |     |     |     +--ro router-id?    inet:ip-address
        |  |     |  |     |     |     +--ro interface-id?   uint32
        |  |     |  |     |     |     +--ro hop-type?       te-hop-type
        |  |     |  |     |     +--:(label)
        |  |     |  |     |     |  +--ro label-hop
        |  |     |  |     |     |     +--ro value?
                                          rt-types:generalized-label
        |  |     |  |     |     +--:(sid)
        |  |     |  |     |        +--ro sid-hop
```

```
        |  |     |  |     |          +--ro sid?
        |  |     |  |     |                rt-types:generalized-label
        |  |     |  |     +--ro state
        |  |     |  |        +--ro index?            uint32
        |  |     |  |        +--ro (type)?
        |  |     |  |           +--:(ip-address)
        |  |     |  |           |  +--ro ip-address-hop
        |  |     |  |           |     +--ro address?    inet:ip-address
        |  |     |  |           |     +--ro hop-type?   te-hop-type
        |  |     |  |           +--:(as-number)
        |  |     |  |           |  +--ro as-number-hop
        |  |     |  |           |     +--ro as-number?   binary
        |  |     |  |           |     +--ro hop-type?    te-hop-type
        |  |     |  |           +--:(unnumbered-link)
        |  |     |  |           |  +--ro unnumbered-hop
        |  |     |  |           |     +--ro router-id?   inet:ip-address
        |  |     |  |           |     +--ro interface-id?   uint32
        |  |     |  |           |     +--ro hop-type?       te-hop-type
        |  |     |  |           +--:(label)
        |  |     |  |           |  +--ro label-hop
        |  |     |  |           |     +--ro value?
        |  |     |  |           |           rt-types:generalized-label
        |  |     |  |           +--:(sid)
        |  |     |  |              +--ro sid-hop
        |  |     |  |                 +--ro sid?
        |  |     |  |                       rt-types:generalized-label
        |  |     |  +--ro bandwidth
        |  |     |     +--ro config
        |  |     |     |  +--ro specification-type?
        |  |     |     |        te-mpls-types:te-bandwidth-type
        |  |     |     |  +--ro set-bandwidth?
        |  |     |     |        te-mpls-types:bandwidth-kbps
        |  |     |     |  +--ro class-type?
        |  |     |     |        te-types:te-ds-class
        |  |     |     +--ro state
        |  |     |        +--ro specification-type?
        |  |     |        |     te-mpls-types:te-bandwidth-type
        |  |     |        +--ro set-bandwidth?
        |  |     |        |     te-mpls-types:bandwidth-kbps
        |  |     |        +--ro class-type?
        |  |     |        |     te-types:te-ds-class
        |  |     |        +--ro signaled-bandwidth?
        |  |     |              te-mpls-types:bandwidth-kbps
        |  |     +--rw te-sr-mpls:config
        |  |     |  +--rw te-sr-mpls:sid-selection-mode?
        |  |     |        te-sid-selection-mode
        |  |     |  +--rw te-sr-mpls:sid-protection?      identityref
        |  |     +--ro te-sr-mpls:state
```

```
|  |        +--ro te-sr-mpls:sid-selection-mode?
                          te-sid-selection-mode
|  |        +--ro te-sr-mpls:sid-protection?       identityref
|  +--rw te-dev:config
|  |  +--rw te-dev:lsp-install-interval?        uint32
|  |  +--rw te-dev:lsp-cleanup-interval?        uint32
|  |  +--rw te-dev:lsp-invalidation-interval?   uint32
|  +--ro te-dev:state
|     +--ro te-dev:lsp-install-interval?        uint32
|     +--ro te-dev:lsp-cleanup-interval?        uint32
|     +--ro te-dev:lsp-invalidation-interval?   uint32
+--rw tunnels
|  +--rw tunnel* [name]
|  |  +--rw name                          -> ../config/name
|  |  +--rw identifier?                   -> ../config/identifier
|  |  +--rw config
|  |  |  +--rw name?                                 string
|  |  |  +--rw type?                                 identityref
|  |  |  +--rw identifier?                           uint16
|  |  |  +--rw description?                          string
|  |  |  +--rw encoding?                             identityref
|  |  |  +--rw protection-type?                      identityref
|  |  |  +--rw admin-status?                         identityref
|  |  |  +--rw preference?                           uint8
|  |  |  +--rw reoptimize-timer?                     uint16
|  |  |  +--rw source?                            inet:ip-address
|  |  |  +--rw destination?                       inet:ip-address
|  |  |  +--rw src-tp-id?                            binary
|  |  |  +--rw dst-tp-id?                            binary
|  |  |  +--rw setup-priority?                       uint8
|  |  |  +--rw hold-priority?                        uint8
|  |  |  +--rw signaling-type?                       identityref
|  |  |  +--rw hierarchical-link-id
|  |  |  |  +--rw local-te-node-id?      te-types:te-node-id
|  |  |  |  +--rw local-te-link-tp-id?   te-types:te-tp-id
|  |  |  |  +--rw remote-te-node-id?     te-types:te-node-id
|  |  |  |  +--rw te-topology-id?        te-types:te-topology-id
|  |  |  +--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 te-dev:lsp-install-interval?        uint32
|  |  |  +--rw te-dev:lsp-cleanup-interval?        uint32
|  |  |  +--rw te-dev:lsp-invalidation-interval?   uint32
|  |  +--ro state
```

```
        |  |  |  +--ro name?                                  string
        |  |  |  +--ro type?                                  identityref
        |  |  |  +--ro identifier?                            uint16
        |  |  |  +--ro description?                           string
        |  |  |  +--ro encoding?                              identityref
        |  |  |  +--ro protection-type?                       identityref
        |  |  |  +--ro admin-status?                          identityref
        |  |  |  +--ro preference?                            uint8
        |  |  |  +--ro reoptimize-timer?                      uint16
        |  |  |  +--ro source?                              inet:ip-address
        |  |  |  +--ro destination?                         inet:ip-address
        |  |  |  +--ro src-tp-id?                             binary
        |  |  |  +--ro dst-tp-id?                             binary
        |  |  |  +--ro setup-priority?                        uint8
        |  |  |  +--ro hold-priority?                         uint8
        |  |  |  +--ro signaling-type?                        identityref
        |  |  |  +--ro hierarchical-link-id
        |  |  |  |  +--ro local-te-node-id?      te-types:te-node-id
        |  |  |  |  +--ro local-te-link-tp-id?   te-types:te-tp-id
        |  |  |  |  +--ro remote-te-node-id?     te-types:te-node-id
        |  |  |  |  +--ro te-topology-id?        te-types:te-topology-id
        |  |  |  +--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 oper-status?                           identityref
        |  |  |  +--ro te-dev:lsp-install-interval?           uint32
        |  |  |  +--ro te-dev:lsp-cleanup-interval?           uint32
        |  |  |  +--ro te-dev:lsp-invalidation-interval?      uint32
        |  |  +--rw bandwidth
        |  |  |  +--rw config
        |  |  |  |  +--rw specification-type?
                          te-mpls-types:te-bandwidth-type
        |  |  |  |  +--rw set-bandwidth?
                          te-mpls-types:bandwidth-kbps
        |  |  |  |  +--rw class-type?           te-types:te-ds-class
        |  |  |  +--ro state
        |  |  |  |  +--ro specification-type?
                          te-mpls-types:te-bandwidth-type
        |  |  |  |  +--ro set-bandwidth?
                          te-mpls-types:bandwidth-kbps
        |  |  |  |  +--ro class-type?           te-types:te-ds-class
        |  |  |  |  +--ro signaled-bandwidth?
                          te-mpls-types:bandwidth-kbps
        |  |  |  +--rw auto-bandwidth
```

```
       |  |  |     +--rw config
       |  |  |     |  +--rw enabled?            boolean
       |  |  |     |  +--rw min-bw?
                           te-mpls-types:bandwidth-kbps
       |  |  |     |  +--rw max-bw?
                           te-mpls-types:bandwidth-kbps
       |  |  |     |  +--rw adjust-interval?    uint32
       |  |  |     |  +--rw adjust-threshold?   te-types:percentage
       |  |  |     +--ro state
       |  |  |     |  +--ro enabled?            boolean
       |  |  |     |  +--ro min-bw?
                           te-mpls-types:bandwidth-kbps
       |  |  |     |  +--ro max-bw?
                           te-mpls-types:bandwidth-kbps
       |  |  |     |  +--ro adjust-interval?    uint32
       |  |  |     |  +--ro adjust-threshold?   te-types:percentage
       |  |  |     +--rw overflow
       |  |  |     |  +--rw config
       |  |  |     |  |  +--rw enabled?               boolean
       |  |  |     |  |  +--rw overflow-threshold?  te-types:percentage
       |  |  |     |  |  +--rw trigger-event-count?   uint16
       |  |  |     |  +--ro state
       |  |  |     |     +--ro enabled?               boolean
       |  |  |     |     +--ro overflow-threshold?  te-types:percentage
       |  |  |     |     +--ro trigger-event-count?   uint16
       |  |  |     +--rw underflow
       |  |  |        +--rw config
       |  |  |        |  +--rw enabled?               boolean
       |  |  |        |  +--rw underflow-threshold? te-types:percentage
       |  |  |        |  +--rw trigger-event-count?   uint16
       |  |  |        +--ro state
       |  |  |           +--ro enabled?               boolean
       |  |  |           +--ro underflow-threshold? te-types:percentage
       |  |  |           +--ro trigger-event-count?   uint16
       |  |  +--rw p2p-primary-paths
       |  |  |  +--rw p2p-primary-path* [name]
       |  |  |     +--rw name                        -> ../config/name
       |  |  |     +--rw config
       |  |  |     |  +--rw name?                       string
       |  |  |     |  +--rw preference?                 uint8
       |  |  |     |  +--rw path-setup-protocol?        identityref
       |  |  |     |  +--rw path-computation-method?    identityref
       |  |  |     |  +--rw path-computation-server?    inet:ip-address
       |  |  |     |  +--rw compute-only?               empty
       |  |  |     |  +--rw use-cspf?                   boolean
       |  |  |     |  +--rw verbatim?                   empty
       |  |  |     |  +--rw lockdown?                   empty
       |  |  |     |  +--rw named-explicit-path?
```

```
       -> ../../../../../../globals/named-explicit-paths/
       named-explicit-path/config/name
       |  |  |     |  +--rw named-path-constraint?
       -> ../../../../../../globals/named-path-constraints/
       named-path-constraint/config/name
       {te-types:named-path-constraints}?
       |  |  |     |  +--rw te-mpls:static-lsp-name?
       |  |  |     |        mpls-static:static-lsp-ref
       |  |  |     +--ro state
       |  |  |     |  +--ro name?                      string
       |  |  |     |  +--ro preference?                uint8
       |  |  |     |  +--ro path-setup-protocol?       identityref
       |  |  |     |  +--ro path-computation-method?   identityref
       |  |  |     |  +--ro path-computation-server?   inet:ip-address
       |  |  |     |  +--ro compute-only?              empty
       |  |  |     |  +--ro use-cspf?                  boolean
       |  |  |     |  +--ro verbatim?                  empty
       |  |  |     |  +--ro lockdown?                  empty
       |  |  |     |  +--ro named-explicit-path?
       -> ../../../../../../globals/named-explicit-paths/
       named-explicit-path/config/name
       |  |  |     |  +--ro named-path-constraint?
       -> ../../../../../../globals/named-path-constraints/
       named-path-constraint/config/name
       {te-types:named-path-constraints}?
       |  |  |     |  +--ro lsps
       |  |  |     |  |  +--ro lsp*
       [source destination tunnel-id lsp-id extended-tunnel-id type]
       |  |  |     |  |     +--ro source
       -> ../../../../../../../../../lsps-state/lsp/source
       |  |  |     |  |     +--ro destination
       -> ../../../../../../../../../lsps-state/lsp/destination
       |  |  |     |  |     +--ro tunnel-id
       -> ../../../../../../../../../lsps-state/lsp/tunnel-id
       |  |  |     |  |     +--ro lsp-id
       -> ../../../../../../../../../lsps-state/lsp/lsp-id
       |  |  |     |  |     +--ro extended-tunnel-id
       -> ../../../../../../../../../lsps-state/lsp/extended-tunnel-id
       |  |  |     |  |     +--ro type
       -> ../../../../../../../../../lsps-state/lsp/type
       |  |  |     |  |     +--ro signaling-type?       identityref
       |  |  |     |  +--ro te-mpls:static-lsp-name?
       mpls-static:static-lsp-ref
       |  |  |     +--rw candidate-p2p-secondary-paths
       |  |  |        +--rw candidate-p2p-secondary-path*
       [secondary-path]
       |  |  |           +--rw secondary-path
       -> ../config/secondary-path
```

```
|  |  |           +--rw config
|  |  |           |  +--rw secondary-path?
-> ../../../../../../p2p-secondary-paths/p2p-secondary-path/
config/name
|  |  |           |  +--rw priority?              uint16
|  |  |           |  +--rw path-setup-protocol?   identityref
|  |  |           +--ro state
|  |  |              +--ro secondary-path?
-> ../../../../../../p2p-secondary-paths/p2p-secondary-path/
config/name
|  |  |              +--ro priority?              uint16
|  |  |              +--ro path-setup-protocol?   identityref
|  |  |              +--ro active?                boolean
|  |  +--rw p2p-secondary-paths
|  |  |  +--rw p2p-secondary-path* [name]
|  |  |     +--rw name      -> ../config/name
|  |  |     +--rw config
|  |  |     |  +--rw name?                        string
|  |  |     |  +--rw preference?                  uint8
|  |  |     |  +--rw path-setup-protocol?         identityref
|  |  |     |  +--rw path-computation-method?     identityref
|  |  |     |  +--rw path-computation-server?     inet:ip-address
|  |  |     |  +--rw compute-only?                empty
|  |  |     |  +--rw use-cspf?                    boolean
|  |  |     |  +--rw verbatim?                    empty
|  |  |     |  +--rw lockdown?                    empty
|  |  |     |  +--rw named-explicit-path?
-> ../../../../../../globals/named-explicit-paths/
named-explicit-path/config/name
|  |  |     |  +--rw named-path-constraint?
-> ../../../../../../globals/named-path-constraints/
named-path-constraint/config/name
{te-types:named-path-constraints}?
|  |  |     |  +--rw te-mpls:static-lsp-name?
mpls-static:static-lsp-ref
|  |  |     +--ro state
|  |  |        +--ro name?                        string
|  |  |        +--ro preference?                  uint8
|  |  |        +--ro path-setup-protocol?         identityref
|  |  |        +--ro path-computation-method?     identityref
|  |  |        +--ro path-computation-server?     inet:ip-address
|  |  |        +--ro compute-only?                empty
|  |  |        +--ro use-cspf?                    boolean
|  |  |        +--ro verbatim?                    empty
|  |  |        +--ro lockdown?                    empty
|  |  |        +--ro named-explicit-path?
-> ../../../../../../globals/named-explicit-paths/
named-explicit-path/config/name
```

```
|  |  |        +--ro named-path-constraint?
-> ../../../../../../globals/named-path-constraints/
named-path-constraint/config/name
{te-types:named-path-constraints}?
|  |  |        +--ro lsps
|  |  |        |  +--ro lsp*
[source destination tunnel-id lsp-id extended-tunnel-id type]
|  |  |        |     +--ro source
-> ../../../../../../../../../lsps-state/lsp/source
|  |  |        |     +--ro destination
-> ../../../../../../../../../lsps-state/lsp/destination
|  |  |        |     +--ro tunnel-id
-> ../../../../../../../../../lsps-state/lsp/tunnel-id
|  |  |        |     +--ro lsp-id
-> ../../../../../../../../../lsps-state/lsp/lsp-id
|  |  |        |     +--ro extended-tunnel-id
-> ../../../../../../../../../lsps-state/lsp/extended-tunnel-id
|  |  |        |     +--ro type
-> ../../../../../../../../../lsps-state/lsp/type
|  |  |        |     +--ro signaling-type?       identityref
|  |  |        +--ro te-mpls:static-lsp-name?
mpls-static:static-lsp-ref
|  |  +---x tunnel-action
|  |  |  +---w input
|  |  |  |  +---w action-type?   identityref
|  |  |  +--ro output
|  |  |     +--ro action-result?   identityref
|  |  +--rw te-mpls:tunnel-igp-shortcut
|  |  |  +--rw te-mpls:config
|  |  |  |  +--rw te-mpls:shortcut-eligible?   boolean
|  |  |  |  +--rw te-mpls:metric-type?         identityref
|  |  |  |  +--rw te-mpls:metric?              int32
|  |  |  |  +--rw te-mpls:routing-afs*         inet:ip-version
|  |  |  +--rw te-mpls:state
|  |  |     +--rw te-mpls:shortcut-eligible?   boolean
|  |  |     +--rw te-mpls:metric-type?         identityref
|  |  |     +--rw te-mpls:metric?              int32
|  |  |     +--rw te-mpls:routing-afs*         inet:ip-version
|  |  +--rw te-mpls:forwarding
|  |     +--rw te-mpls:config
|  |     |  +--rw te-mpls:binding-label?   rt-types:mpls-label
|  |     |  +--rw te-mpls:load-share?      uint32
|  |     |  +--rw te-mpls:policy-class?    uint8
|  |     +--rw te-mpls:state
|  |        +--rw te-mpls:binding-label?   rt-types:mpls-label
|  |        +--rw te-mpls:load-share?      uint32
|  |        +--rw te-mpls:policy-class?    uint8
|  +--rw tunnel-p2mp* [name]
```

```
|     +--rw name          -> ../config/name
|     +--rw identifier?   -> ../config/identifier
|     +--rw config
|     |  +--rw name?                  string
|     |  +--rw type?                  identityref
|     |  +--rw identifier?            uint16
|     |  +--rw description?           string
|     |  +--rw setup-priority?        uint8
|     |  +--rw hold-priority?         uint8
|     |  +--rw lsp-protection-type?   identityref
|     |  +--rw admin-status?          identityref
|     |  +--rw reoptimize-timer?      uint16
|     +--ro state
|        +--ro name?                  string
|        +--ro type?                  identityref
|        +--ro identifier?            uint16
|        +--ro description?           string
|        +--ro setup-priority?        uint8
|        +--ro hold-priority?         uint8
|        +--ro lsp-protection-type?   identityref
|        +--ro admin-status?          identityref
|        +--ro reoptimize-timer?      uint16
+--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 path-setup-protocol?            identityref
|     +--ro origin-type?                    enumeration
|     +--ro lsp-resource-status?            enumeration
|     +--ro lsp-protection-role?            enumeration
|     +--ro lsp-carry-normal-traffic?       empty
|     +--ro lsp-record-route-subobjects
|     |  +--ro record-route-subobject* [index]
|     |     +--ro index           uint32
|     |     +--ro (type)?
|     |        +--:(ip-address)
|     |        |  +--ro ip-address?     inet:ip-address
|     |        |  +--ro ip-flags?       binary
|     |        +--:(unnumbered-link)
|     |        |  +--ro router-id?      inet:ip-address
|     |        |  +--ro interface-id?   uint32
|     |        +--:(label)
```

```
|     |           +--ro value?        rt-types:generalized-label
|     |           +--ro label-flags?    binary
|     +--ro te-dev:lsp-timers
|     |  +--ro te-dev:life-time?         uint32
|     |  +--ro te-dev:time-to-install?   uint32
|     |  +--ro te-dev:time-to-destroy?   uint32
|     +--ro te-dev:downstream-info
|     |  +--ro te-dev:nhop?                  inet:ip-address
|     |  +--ro te-dev:outgoing-interface?   if:interface-ref
|     |  +--ro te-dev:neighbor?              inet:ip-address
|     |  +--ro te-dev:label?         rt-types:generalized-label
|     +--ro te-dev:upstream-info
|        +--ro te-dev:phop?       inet:ip-address
|        +--ro te-dev:neighbor?   inet:ip-address
|        +--ro te-dev:label?      rt-types:generalized-label
+--rw te-dev:interfaces
   +--rw te-dev:config
   |  +--rw te-dev:threshold-type?            enumeration
   |  +--rw te-dev:delta-percentage?          te-types:percentage
   |  +--rw te-dev:threshold-specification?   enumeration
   |  +--rw te-dev:up-thresholds*             te-types:percentage
   |  +--rw te-dev:down-thresholds*           te-types:percentage
   |  +--rw te-dev:up-down-thresholds*        te-types:percentage
   +--ro te-dev:state
   |  +--ro te-dev:threshold-type?            enumeration
   |  +--ro te-dev:delta-percentage?          te-types:percentage
   |  +--ro te-dev:threshold-specification?   enumeration
   |  +--ro te-dev:up-thresholds*             te-types:percentage
   |  +--ro te-dev:down-thresholds*           te-types:percentage
   |  +--ro te-dev:up-down-thresholds*        te-types:percentage
   +--rw te-dev:interface* [interface]
      +--rw te-dev:interface    if:interface-ref
      +--rw te-dev:config
      |  +--rw te-dev:te-metric?              te-types:te-metric
      |  +--rw (te-dev:admin-group-type)?
      |  |  +--:(te-dev:value-admin-groups)
      |  |  |  +--rw (te-dev:value-admin-group-type)?
      |  |  |     +--:(te-dev:admin-groups)
      |  |  |     | +--rw te-dev:admin-group?te-types:admin-group
      |  |  |     +--:(te-dev:extended-admin-groups)
                              {te-types:extended-admin-groups}?
      |  |  |       +--rw te-dev:extended-admin-group?
                              te-types:extended-admin-group
      |  |  +--:(te-dev:named-admin-groups)
      |  |     +--rw te-dev:named-admin-groups[named-admin-group]
      {te-types:extended-admin-groups,
      te-types:named-extended-admin-groups}?
      |  |        +--rw te-dev:named-admin-group
```

```
            -> ../../../../../te:globals/named-admin-groups/
            named-admin-group/config/name
            |  +--rw (te-dev:srlg-type)?
            |  |  +--:(te-dev:value-srlgs)
            |  |  |  +--rw te-dev:values* [value]
            |  |  |     +--rw te-dev:value    uint32
            |  |  +--:(te-dev:named-srlgs)
            |  |     +--rw te-dev:named-srlgs* [named-srlg]
            {te-types:named-srlg-groups}?
            |  |        +--rw te-dev:named-srlg
            -> ../../../../../te:globals/named-srlgs/
            named-srlg/config/name
            |  +--rw te-dev:threshold-type?            enumeration
            |  +--rw te-dev:delta-percentage?       te-types:percentage
            |  +--rw te-dev:threshold-specification?   enumeration
            |  +--rw te-dev:up-thresholds*          te-types:percentage
            |  +--rw te-dev:down-thresholds*        te-types:percentage
            |  +--rw te-dev:up-down-thresholds*     te-types:percentage
            |  +--rw te-dev:switching-capabilities*
            |     +--rw te-dev:switching-capability    identityref
            |     +--rw te-dev:encoding?               identityref
            +--ro te-dev:state
               +--ro te-dev:te-metric?           te-types:te-metric
               +--ro (te-dev:admin-group-type)?
               |  +--:(te-dev:value-admin-groups)
               |  |  +--ro (te-dev:value-admin-group-type)?
               |  |     +--:(te-dev:admin-groups)
               |  |     |  +--ro te-dev:admin-group?
               te-types:admin-group
               |  |     +--:(te-dev:extended-admin-groups)
               {te-types:extended-admin-groups}?
               |  |        +--ro te-dev:extended-admin-group?
               |  +--:(te-dev:named-admin-groups)
               |    +--ro te-dev:named-admin-groups*[named-admin-group]
               |       +--ro te-dev:named-admin-group
               -> ../../../../../te:globals/named-admin-groups/
               named-admin-group/config/name
               +--ro (te-dev:srlg-type)?
               |  +--:(te-dev:value-srlgs)
               |  |  +--ro te-dev:values* [value]
               |  |     +--ro te-dev:value    uint32
               |  +--:(te-dev:named-srlgs)
               |     +--ro te-dev:named-srlgs* [named-srlg]
               |        +--ro te-dev:named-srlg
               -> ../../../../../te:globals/named-srlgs/
               named-srlg/config/name
               +--ro te-dev:switching-capabilities*
               |  +--ro te-dev:switching-capability    identityref
```

```
               |  +--ro te-dev:encoding?               identityref
               +--ro te-dev:threshold-type?            enumeration
               +--ro te-dev:delta-percentage?       te-types:percentage
               +--ro te-dev:threshold-specification?   enumeration
               +--ro te-dev:up-thresholds*          te-types:percentage
               +--ro te-dev:down-thresholds*        te-types:percentage
               +--ro te-dev:up-down-thresholds*     te-types:percentage
               +--ro te-dev:te-advertisements_state
                  +--ro te-dev:flood-interval?           uint32
                  +--ro te-dev:last-flooded-time?        uint32
                  +--ro te-dev:next-flooded-time?        uint32
                  +--ro te-dev:last-flooded-trigger?     enumeration
                  +--ro te-dev:advertized-level-areas* [level-area]
                     +--ro te-dev:level-area    uint32

  rpcs:
    +---x globals-rpc
    +---x interfaces-rpc
    +---x tunnels-rpc
       +---w input
       |  +---w tunnel-info
       |     +---w (type)?
       |        +--:(tunnel-p2p)
       |        |  +---w p2p-id?    te:tunnel-ref
       |        +--:(tunnel-p2mp)
       |           +---w p2mp-id?   te:tunnel-p2mp-ref
       +--ro output
          +--ro result
             +--ro result?   enumeration

  notifications:
    +---n globals-notif
    +---n tunnels-notif
module: ietf-te-device

  rpcs:
    +---x interfaces-rpc

  notifications:
    +---n interfaces-notif
```

Figure 6: TE generic model configuration and state tree

4. TE Generic and Helper YANG Modules

```
 <CODE BEGINS>file "ietf-te-types@2017-03-10.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;
   }

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

   import ietf-routing-types {
     prefix "rt-types";
   }

   organization
     "IETF Traffic Engineering Architecture and Signaling (TEAS)
      Working Group";

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

      WG Chair: Lou Berger
                <mailto:lberger@labn.net>

      WG Chair: Vishnu Pavan Beeram
                <mailto:vbeeram@juniper.net>

      Editor:   Tarek Saad
                <mailto:tsaad@cisco.com>

      Editor:   Rakesh Gandhi
                <mailto:rgandhi@cisco.com>

      Editor:   Vishnu Pavan Beeram
                <mailto:vbeeram@juniper.net>

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

      Editor:   Xufeng Liu
                <mailto:xufeng.liu@ericsson.com>

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

```
      Editor:   Raqib Jones
                <mailto:raqib@Brocade.com>

      Editor:   Bin Wen
                <mailto:Bin_Wen@cable.comcast.com>";

   description
     "This module contains a collection of generally
     useful TE specific YANG data type defintions.";

   revision "2017-03-10" {
     description "Latest revision of TE types";
     reference "RFC3209";
   }

   /*
    * Identities
    */

   identity path-computation-method {
     description
      "base identity for supported path computation
       mechanisms";
   }

   identity path-locally-computed {
     base path-computation-method;
     description
       "indicates a constrained-path LSP in which the
       path is computed by the local LER";
   }

   identity path-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.
       In the case that an LSP is defined to be externally queried, it
       may also have associated explicit definitions (which are provided
       to the external source to aid computation); and the path that is
       returned by the external source is not required to provide a
       wholly resolved path back to the originating system - that is to
       say, some local computation may also be required";
   }

   identity path-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 te-ds-class {
      type uint8 {
        range "0..7";
      }
      description
        "The Differentiatied Class-Type of traffic.";
      reference "RFC4124: section-4.3.1";
    }

    typedef te-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";
    }

    identity LSP_METRIC_TYPE {
      description
        "Base identity for types of LSP metric specification";
    }

    identity LSP_METRIC_RELATIVE {
      base LSP_METRIC_TYPE;
      description
        "The metric specified for the LSPs to which this identity refers
        is specified as a relative value to the IGP metric cost to the
        LSP's tail-end.";
    }

    identity LSP_METRIC_ABSOLUTE {
      base LSP_METRIC_TYPE;
      description
        "The metric specified for the LSPs to which this identity refers
        is specified as an absolute value";
```

```
   }

   identity LSP_METRIC_INHERITED {
     base LSP_METRIC_TYPE;
     description
       "The metric for for the LSPs to which this identity refers is
       not specified explicitly - but rather inherited from the IGP
       cost directly";
   }

   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 tunnel-action-type {
     description
       "Base identity from which specific tunnel action types
        are derived.";
   }

   identity tunnel-action-resetup {
     base tunnel-action-type;
     description
       "TE tunnel action resetup. Tears the
       tunnel's current LSP (if any) and
       attempts to re-establish a new LSP";
   }

   identity tunnel-action-reoptimize {
     base tunnel-action-type;
     description
       "TE tunnel action reoptimize.
        Reoptimizes placement of the tunnel LSP(s)";
   }
```

```
   identity tunnel-action-switchpath {
     base tunnel-action-type;
     description
       "TE tunnel action reoptimize
        Switches the tunnel's LSP to use the specified path";
   }

   identity te-action-result {
     description
       "Base identity from which specific TE action results
        are derived.";
   }

   identity te-action-success {
     base te-action-result;
     description "TE action successul.";
   }

   identity te-action-fail {
     base te-action-result;
     description "TE action failed.";
   }

   identity tunnel-action-inprogress {
     base te-action-result;
     description "TE action inprogress.";
   }

   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 path-invalidation-action-type {
     description
       "Base identity for TE path invalidation action types";
```

```
   }

   identity path-invalidation-action-drop-type {
     base path-invalidation-action-type;
     description
       "TE path invalidation action drop";
   }

   identity path-invalidation-action-drop-tear {
     base path-invalidation-action-type;
     description
       "TE path invalidation action tear";
   }

   identity lsp-prot-type {
     description
       "Base identity from which LSP protection types are
       derived.";
   }

   identity lsp-prot-unprotected {
     base lsp-prot-type;
     description
       "LSP protection 'Unprotected'";
     reference "RFC4872";
   }

   identity lsp-prot-reroute-extra {
     base lsp-prot-type;
     description
       "LSP protection '(Full) Rerouting'";
     reference "RFC4872";
   }

   identity lsp-prot-reroute {
     base lsp-prot-type;
     description
       "LSP protection 'Rerouting without Extra-Traffic'";
     reference "RFC4872";
   }

   identity lsp-prot-1-for-n {
     base lsp-prot-type;
     description
       "LSP protection '1:N Protection with Extra-Traffic'";
     reference "RFC4872";
   }
```

```
   identity lsp-prot-unidir-1-to-1 {
     base lsp-prot-type;
     description
       "LSP protection '1+1 Unidirectional Protection'";
     reference "RFC4872";
   }

   identity lsp-prot-bidir-1-to-1 {
     base lsp-prot-type;
     description
       "LSP protection '1+1 Bidirectional Protection'";
     reference "RFC4872";
   }

   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";
  }

  identity path-signaling-type {
    description
      "Base identity from which specific path signaling
       types are derived.";
  }

  identity path-signaling-rsvpte {
```

```
    base tunnel-type;
    description
      "RSVP-TE path signaling type";
  }

  identity path-signaling-sr {
    base tunnel-type;
    description
      "Segment-routing path signaling type";
  }

  identity te-path-setup-protocol {
    description
      "base identity for supported TE LSPs signaling
       protocols";
  }

  identity te-path-setup-static {
    base te-path-setup-protocol;
    description
      "Static LSP provisioning";
  }

  identity te-path-setup-rsvp {
    base te-path-setup-protocol;
    description
      "RSVP-TE signaling protocol";
  }

  identity te-path-setup-sr {
    base te-path-setup-protocol;
    description
      "Segment routing";
  }

  /* 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-hop_config {
     description
       "The explicit route subobject grouping";
     leaf index {
       type uint32;
       description "ERO subobject index";
     }
     choice type {
       description
         "The explicit route subobject type";
       case ip-address {
         description
           "IP address explicit route subobject";
         container ip-address-hop {
           description "IP address hop type";
           leaf address {
             type inet:ip-address;
             description
               "ERO IP address.";
           }
           leaf hop-type {
             type te-hop-type;
               description
                 "strict or loose hop";
```

```
          }
        }
      }
      case as-number {
        container as-number-hop {
          leaf as-number {
            type binary {
              length 16;
            }
            description "AS number";
          }
          leaf hop-type {
            type te-hop-type;
              description
                "strict or loose hop";
          }
          description
            "Autonomous System explicit route subobject";
        }
      }
      case unnumbered-link {
        container unnumbered-hop {
          leaf router-id {
            type inet:ip-address;
            description
              "A router-id address";
          }
          leaf interface-id {
            type uint32;
            description "The interface identifier";
          }
          leaf hop-type {
            type te-hop-type;
              description
                "strict or loose hop";
          }
          description
            "Unnumbered link explicit route subobject";
          reference
            "RFC3477: Signalling Unnumbered Links in
            RSVP-TE";
        }
      }
      case label {
        container label-hop {
          description "Label hop type";
          leaf value {
            type rt-types:generalized-label;
```

```
            description "the label value";
          }
        }
        description
          "The Label ERO subobject";
      }
      case sid {
        container sid-hop {
          description "foo";
          leaf sid {
            type rt-types:generalized-label;
            description "Segment-routing identifier";
          }
        }
        description "Segment-routing identifier";
      }
    }
  }

  grouping explicit-route-hop {
    description "Explicit route hop grouping";
    container config {
      description
        "Configuration parameters for the explicit route hop";
      uses explicit-route-hop_config;
    }
    container state {
      config false;
      description
        "State parameters for the explicit route hop";
      uses explicit-route-hop_config;
    }
  }

  grouping record-route-subobject {
    description
      "The record route subobject grouping";
    choice type {
      description
        "The record route subobject type";
      case ip-address {
        leaf ip-address {
          type inet:ip-address;
          description
            "RRO IP address subobject.";
        }
        leaf ip-flags {
          type binary {
```

```
            length 8;
          }
          description
            "RRO IP address sub-object flags";
          reference "RFC3209";
        }
      }
      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 record route subobject";
        reference
          "RFC3477: Signalling Unnumbered Links in
           RSVP-TE";
      }
      case label {
        leaf value {
          type rt-types:generalized-label;
          description "the label value";
        }
        leaf label-flags {
          type binary {
            length 8;
          }
          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-metric-hop {
      base path-metric-type;
      description
        "Hop 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";
   }

   identity te-optimization-criterion {
     description
       "Base identity for TE optimization criterion.";
     reference
       "RFC3272: Overview and Principles of Internet Traffic
        Engineering.";
   }

   identity not-optimized {
     base te-optimization-criterion;
     description "Optimization is not applied.";
   }

   identity cost {
     base te-optimization-criterion;
     description "Optimized on cost.";
   }
```

```
   identity delay {
     base te-optimization-criterion;
     description "Optimized on delay.";
   }

   /*
    * Typedefs
    */

   typedef percentage {
     type uint8 {
       range "0..100";
     }
     description
       "Integer indicating a percentage value";
   }

   typedef performance-metric-normality {
     type enumeration {
       enum "unknown" {
         value 0;
         description
           "Unknown.";
       }
       enum "normal" {
         value 1;
         description
           "Normal.";
       }
       enum "abnormal" {
         value 2;
         description
           "Abnormal. The anomalous bit is set.";
       }
     }
     description
       "Indicates whether a performance metric is normal, abnormal, or
        unknown.";
     reference
       "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions.
        RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
        RFC7823: Performance-Based Path Selection for Explicitly
        Routed Label Switched Paths (LSPs) Using TE Metric
        Extensions";
   }

   typedef te-admin-status {
     type enumeration {
```

```
        enum up {
          description
            "Enabled.";
        }
        enum down {
          description
            "Disabled.";
        }
        enum testing {
          description
            "In some test mode.";
        }
        enum preparing-maintenance {
          description
            "Resource is disabled in the control plane to prepare for
             graceful shutdown for maintenance purposes.";
          reference
            "RFC5817: Graceful Shutdown in MPLS and Generalized MPLS
             Traffic Engineering Networks";
        }
        enum maintenance {
          description
            "Resource is disabled in the data plane for maintenance
             purposes.";
        }
      }
      description
        "Defines a type representing the administrative status of
         a TE resource.";
    }

    typedef te-global-id {
      type uint32;
      description
        "An identifier to uniquely identify an operator, which can be
         either a provider or a client.
         The definition of this type is taken from RFC6370 and RFC5003.
         This attribute type is used solely to provide a globally
         unique context for TE topologies.";
    }

    typedef te-link-access-type {
      type enumeration {
        enum point-to-point {
          description
            "The link is point-to-point.";
        }
        enum multi-access {
```

```
        description
          "The link is multi-access, including broacast and NBMA.";
      }
    }
    description
      "Defines a type representing the access type of a TE link.";
    reference
      "RFC3630: Traffic Engineering (TE) Extensions to OSPF
       Version 2.";
  }

  typedef te-node-id {
    type yang:dotted-quad;
    description
      "An identifier for a node in a topology.
       The identifier is represented as 32-bit unsigned integer in
       the dotted-quad notation.
       This attribute is mapped to Router ID in
       RFC3630, RFC5329, RFC5305, and RFC6119.";
  }

  typedef te-oper-status {
    type enumeration {
      enum up {
        description
        "Operational up.";
      }
      enum down {
        description
        "Operational down.";
      }
      enum testing {
        description
        "In some test mode.";
      }
      enum unknown {
        description
        "Status cannot be determined for some reason.";
      }
      enum preparing-maintenance {
        description
          "Resource is disabled in the control plane to prepare for
           graceful shutdown for maintenance purposes.";
        reference
          "RFC5817: Graceful Shutdown in MPLS and Generalized MPLS
           Traffic Engineering Networks";
      }
      enum maintenance {
```

```
        description
          "Resource is disabled in the data plane for maintenance
           purposes.";
      }
    }
    description
      "Defines a type representing the operational status of
       a TE resource.";
  }

  typedef te-recovery-status {
    type enumeration {
      enum normal {
        description
          "Both the recovery and working spans are fully
           allocated and active, data traffic is being
           transported over (or selected from) the working
           span, and no trigger events are reported.";
      }
      enum recovery-started {
        description
          "The recovery action has been started, but not completed.";
      }
      enum recovery-succeeded {
        description
          "The recovery action has succeeded. The working span has
           reported a failure/degrade condition and the user traffic
           is being transported (or selected) on the recovery span.";
      }
      enum recovery-failed {
        description
          "The recovery action has failed.";
      }
      enum reversion-started {
        description
          "The reversion has started.";
      }
      enum reversion-failed {
        description
          "The reversion has failed.";
      }
      enum recovery-unavailable {
        description
          "The recovery is unavailable -- either as a result of an
           operator Lockout command or a failure condition detected
           on the recovery span.";
      }
      enum recovery-admin {
```

```
          description
            "The operator has issued a command switching the user
             traffic to the recovery span.";
        }
        enum wait-to-restore {
          description
            "The recovery domain is recovering from a failuer/degrade
             condition on the working span that is being controlled by
             the Wait-to-Restore (WTR) timer.";
        }
      }
      description
        "Defines the status of a recovery action.";
      reference
        "RFC4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS).
         RFC6378: MPLS Transport Profile (MPLS-TP) Linear Protection";
    }

    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-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
```

```
   typedef te-topology-id {
     type string {
       pattern
         '([a-zA-Z0-9\-_.]+:)*'
       + '/?([a-zA-Z0-9\-_.]+)(/[a-zA-Z0-9\-_.]+)*';
     }
     description
       "An identifier for a topology.
        It is optional to have one or more prefixes at the begining,
        separated by colons. The prefixes can be the network-types,
        defined in ietf-network.yang, to help user to understand the
        topology better before further inquiry.";
   }

   typedef te-tp-id {
     type union {
       type uint32;          // Unnumbered
       type inet:ip-address; // IPv4 or IPv6 address
     }
     description
       "An identifier for a TE link endpoint on a node.
        This attribute is mapped to local or remote link identifier in
        RFC3630 and RFC5305.";
   }

   typedef admin-group {
     type binary {
       length 4;
     }
     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";
  }

  /**
   * TE performance metric groupings
   **/
  grouping performance-metric-container {
    description
      "A container containing performance metric attributes.";
    container performance-metric {
      description
        "Link performance information in real time.";
```

```
      reference
        "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions.
         RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
         RFC7823: Performance-Based Path Selection for Explicitly
         Routed Label Switched Paths (LSPs) Using TE Metric
         Extensions";
      container measurement {
        description
          "Measured performance metric values. Static configuration
           and manual overrides of these measurements are also
           allowed.";
        uses performance-metric-attributes;
      }
      container normality
      {
        description
          "Performance metric normality values.";
        uses performance-metric-normality-attributes;
      }
      uses performance-metric-throttle-container;
    }
  } // performance-metric-container

  grouping performance-metric-attributes {
    description
      "Link performance information in real time.";
    reference
      "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions.
       RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
       RFC7823: Performance-Based Path Selection for Explicitly
       Routed Label Switched Paths (LSPs) Using TE Metric
       Extensions";
    leaf unidirectional-delay {
      type uint32 {
        range 0..16777215;
      }
      description "Delay or latency in micro seconds.";
    }
    leaf unidirectional-min-delay {
      type uint32 {
        range 0..16777215;
      }
      description "Minimum delay or latency in micro seconds.";
    }
    leaf unidirectional-max-delay {
      type uint32 {
        range 0..16777215;
      }
```

```
        description "Maximum delay or latency in micro seconds.";
      }
      leaf unidirectional-delay-variation {
        type uint32 {
          range 0..16777215;
        }
        description "Delay variation in micro seconds.";
      }
      leaf unidirectional-packet-loss {
        type decimal64 {
          fraction-digits 6;
          range "0 .. 50.331642";
        }
        description
          "Packet loss as a percentage of the total traffic sent
           over a configurable interval. The finest precision is
           0.000003%.";
      }
      leaf unidirectional-residual-bandwidth {
        type rt-types:bandwidth-ieee-float32;
        description
          "Residual bandwidth that subtracts tunnel
           reservations from Maximum Bandwidth (or link capacity)
           [RFC3630] and provides an aggregated remainder across QoS
           classes.";
      }
      leaf unidirectional-available-bandwidth {
        type rt-types:bandwidth-ieee-float32;
        description
          "Available bandwidth that is defined to be residual
           bandwidth minus the measured bandwidth used for the
           actual forwarding of non-RSVP-TE LSP packets.  For a
           bundled link, available bandwidth is defined to be the
           sum of the component link available bandwidths.";
      }
      leaf unidirectional-utilized-bandwidth {
        type rt-types:bandwidth-ieee-float32;
        description
          "Bandwidth utilization that represents the actual
           utilization of the link (i.e. as measured in the router).
           For a bundled link, bandwidth utilization is defined to
           be the sum of the component link bandwidth
           utilizations.";
      }
    } // performance-metric-attributes

    grouping performance-metric-normality-attributes {
      description
```

```
        "Link performance metric normality attributes.";
      reference
        "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions.
         RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
         RFC7823: Performance-Based Path Selection for Explicitly
         Routed Label Switched Paths (LSPs) Using TE Metric
         Extensions";
      leaf unidirectional-delay {
        type te-types:performance-metric-normality;
        description "Delay normality.";
      }
      leaf unidirectional-min-delay {
        type te-types:performance-metric-normality;
        description "Minimum delay or latency normality.";
      }
      leaf unidirectional-max-delay {
        type te-types:performance-metric-normality;
        description "Maximum delay or latency normality.";
      }
      leaf unidirectional-delay-variation {
        type te-types:performance-metric-normality;
        description "Delay variation normality.";
      }
      leaf unidirectional-packet-loss {
        type te-types:performance-metric-normality;
        description "Packet loss normality.";
      }
      leaf unidirectional-residual-bandwidth {
        type te-types:performance-metric-normality;
        description "Residual bandwidth normality.";
      }
      leaf unidirectional-available-bandwidth {
        type te-types:performance-metric-normality;
        description "Available bandwidth normality.";
      }
      leaf unidirectional-utilized-bandwidth {
        type te-types:performance-metric-normality;
        description "Bandwidth utilization normality.";
      }
    } // performance-metric-normality-attributes

    grouping performance-metric-throttle-container {
      description
        "A container controlling performance metric throttle.";
      container throttle {
        must "suppression-interval >= measure-interval" {
          error-message
            "suppression-interval cannot be less then
```

```
           measure-interval.";
        description
          "Constraint on suppression-interval and
           measure-interval.";
      }
      description
        "Link performance information in real time.";
      reference
        "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions.
         RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
         RFC7823: Performance-Based Path Selection for Explicitly
         Routed Label Switched Paths (LSPs) Using TE Metric
         Extensions";
      leaf unidirectional-delay-offset {
        type uint32 {
          range 0..16777215;
        }
        description
          "Offset value to be added to the measured delay value.";
      }
      leaf measure-interval {
        type uint32;
        default 30;
        description
          "Interval in seconds to measure the extended metric
           values.";
      }
      leaf advertisement-interval {
        type uint32;
        description
          "Interval in seconds to advertise the extended metric
           values.";
      }
      leaf suppression-interval {
        type uint32 {
          range "1 .. max";
        }
        default 120;
        description
          "Interval in seconds to suppress advertising the extended
           metric values.";
      }
      container threshold-out {
        uses performance-metric-attributes;
        description
          "If the measured parameter falls outside an upper bound
           for all but the min delay metric (or lower bound for
           min-delay metric only) and the advertised value is not
```

```
              already outside that bound, anomalous announcement will be
              triggered.";
        }
        container threshold-in {
          uses performance-metric-attributes;
          description
            "If the measured parameter falls inside an upper bound
             for all but the min delay metric (or lower bound for
             min-delay metric only) and the advertised value is not
             already inside that bound, normal (anomalous-flag cleared)
             announcement will be triggered.";
        }
        container threshold-accelerated-advertisement {
          description
            "When the difference between the last advertised value and
             current measured value exceed this threshold, anomalous
             announcement will be triggered.";
          uses performance-metric-attributes;
        }
      }
    } // performance-metric-throttle-container

    /**
     * TE tunnel generic groupings
     **/

    /* Tunnel path selection parameters */
    /*** End of TE tunnel groupings ***/

    /**
     * TE interface generic groupings
     **/
 }
 <CODE ENDS>
```

Figure 7: TE basic types YANG module

```
<CODE BEGINS> file "ietf-te@2017-03-10.yang"
module ietf-te {
  yang-version 1.1;

  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 te-types;
  }

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

  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
     Working Group";

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

     WG Chair: Lou Berger
               <mailto:lberger@labn.net>

     WG Chair: Vishnu Pavan Beeram
               <mailto:vbeeram@juniper.net>

     Editor:   Tarek Saad
               <mailto:tsaad@cisco.com>

     Editor:   Rakesh Gandhi
               <mailto:rgandhi@cisco.com>

     Editor:   Vishnu Pavan Beeram
               <mailto:vbeeram@juniper.net>

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

     Editor:   Xufeng Liu
               <mailto:Xufeng_Liu@jabil.com>

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

     Editor:   Raqib Jones
               <mailto:raqib@Brocade.com>

     Editor:   Bin Wen
               <mailto:Bin_Wen@cable.comcast.com>";
```

```
  description
    "YANG data module for TE configuration,
    state, RPC and notifications.";

  revision "2017-03-10" {
    description "Latest update to TE generic YANG module.";
    reference "TBD";
  }

  typedef tunnel-ref {
    type leafref {
      path "/te:te/te:tunnels/te:tunnel/te:name";
    }
    description
      "This type is used by data models that need to reference
       configured TE tunnel.";
  }

  typedef tunnel-p2mp-ref {
    type leafref {
      path "/te:te/te:tunnels/te:tunnel-p2mp/te:name";
    }
    description
      "This type is used by data models that need to reference
       configured P2MP TE tunnel.";
  }

  /**
   * TE tunnel generic groupings
   */
  grouping path-route-objects {
    description
      "List of EROs to be included or excluded when performing
       the path computation.";
    container explicit-route-objects {
      description
        "Container for the include or exclude route object list";
      list explicit-route-object {
        key index;
        description
          "List of explicit route objects to include or
           exclude in path computation";
        leaf index {
          type leafref {
            path "../config/index";
          }
          description
            "Index of this explicit route object";
```

```
        }
        leaf explicit-route-usage {
          type identityref {
            base te-types:route-usage-type;
          }
          description "An explicit-route hop action.";
        }
        uses te-types:explicit-route-hop;
      }
    }
  }

  grouping path-affinities {
    description
      "Path affinities grouping";
    container 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 te-types: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 path-srlgs {
    description
      "Path SRLG properties grouping";
    container path-srlgs {
      description
        "Path SRLG properties container";
      choice style {
        description
          "Type of SRLG representation";
        case values {
          leaf usage {
            type identityref {
              base te-types:route-exclude-srlg;
            }
            description "SRLG usage";
          }
          leaf-list values {
            type te-types:srlg;
            description "SRLG value";
          }
        }
        case named {
          list constraints {
            key "usage";
            leaf usage {
              type identityref {
                base te-types: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 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 te-types:bidir-association-type;
          }
          default te-types:bidir-assoc-non-corouted;
          description
            "The TE tunnel association type.";
        }
        leaf provisioing {
          type identityref {
            base te-types: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";
        }
      }
    }
  }

  grouping p2p-secondary-path-properties {
    description
      "tunnel path properties.";
    container config {
      description
        "Configuration parameters relating to
        tunnel properties";
      uses p2p-path-properties_config;
    }
    container state {
      config false;
      description
        "State information associated with tunnel
        properties";
      uses p2p-path-properties_config;
      uses p2p-secondary-path-properties_state;
    }
  }

  grouping p2p-primary-path-properties {
    description
      "TE tunnel primary path properties grouping";
    container config {
```

```
      description
        "Configuration parameters relating to
        tunnel properties";
      uses p2p-path-properties_config;
    }
    container state {
      config false;
      description
        "State information associated with tunnel
        properties";
      uses p2p-path-properties_config;
      uses p2p-primary-path-properties_state;
    }
  }

  grouping p2p-primary-path-properties_state {
    description "TE path state parameters";
    container lsps {
      description "TE LSPs container";
      list lsp {
        key
          "source destination tunnel-id lsp-id "+
          "extended-tunnel-id type";
        description "List of LSPs associated with the tunnel.";

        leaf source {
          type leafref {
            path "../../../../../../../../lsps-state/lsp/source";
          }
          description
            "Tunnel sender address extracted from
            SENDER_TEMPLATE  object";
          reference "RFC3209";
        }
        leaf destination {
          type leafref {
            path "../../../../../../../../lsps-state/lsp/destination";
          }
          description
            "Tunnel endpoint address extracted from
            SESSION object";
          reference "RFC3209";
        }
        leaf tunnel-id {
          type leafref {
            path "../../../../../../../../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 "../../../../../../../../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 "../../../../../../../../lsps-state/lsp/" +
                 "extended-tunnel-id";
          }
          description
            "Extended Tunnel ID of the LSP.";
          reference "RFC3209";
        }
        leaf type {
          type leafref {
            path "../../../../../../../../lsps-state/lsp/type";
          }
          description "LSP type P2P or P2MP";
        }
        leaf signaling-type {
          type identityref {
            base te-types:path-signaling-type;
          }
          description "TE tunnel path signaling type";
        }
      }
    }
  }

  grouping p2p-secondary-path-properties_state {
    description "TE secondary path state parameters";
    list lsp {
      key "source";
      description "List of LSPs associated with the tunnel.";

      leaf source {
```

```
        type leafref {
          path "../../../../../../../../lsps-state/lsp/source";
        }
        description
          "Tunnel sender address extracted from
          SENDER_TEMPLATE  object";
        reference "RFC3209";
      }
      leaf destination {
        type leafref {
          path "../../../../../../../../lsps-state/lsp/destination";
        }
        description
          "Tunnel endpoint address extracted from
          SESSION object";
        reference "RFC3209";
      }
      leaf tunnel-id {
        type leafref {
          path "../../../../../../../../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 "../../../../../../../../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 "../../../../../../../../lsps-state/lsp" +
               "/extended-tunnel-id";
        }
        description
          "Extended Tunnel ID of the LSP.";
        reference "RFC3209";
      }
      leaf type {
```

```
        type leafref {
          path "../../../../../../../../lsps-state/lsp/type";
        }
        description "LSP type P2P or P2MP";
      }
      leaf active {
        type boolean;
        description
          "Indicates the current active path option that has
          been selected of the candidate secondary paths";
      }
    }
  }

  grouping p2p-path-properties_config {
    description
      "TE tunnel path properties configuration grouping";
    leaf name {
      type string;
      description "TE path name";
    }
    leaf preference {
      type uint8 {
        range "1..255";
      }
      description
        "Specifies a preference for this path. The lower the
        number higher the preference";
    }
    leaf path-setup-protocol {
      type identityref {
        base te-types:te-path-setup-protocol;
      }
      description
        "Signaling protocol used to set up this tunnel";
    }
    leaf path-computation-method {
      type identityref {
        base te-types:path-computation-method;
      }
      default te-types:path-locally-computed;
      description
        "The method used for computing the path, either
        locally computed, queried from a server or not
        computed at all (explicitly configured).";
    }
    leaf path-computation-server {
      when "../path-computation-method = 'path-externally-queried'" {
```

```
        description
          "The path-computation server when the path is
           externally queried";
      }
      type inet:ip-address;
      description
        "Address of the external path computation
         server";
    }
    leaf compute-only {
      type empty;
      description
        "When set, the path is computed and updated whenever
         the topology is updated. No resources are committed
         or reserved in the network.";
    }
    leaf use-cspf {
      when "../path-computation-method = 'path-locally-computed'";
      type boolean;
      description "A CSPF dynamically computed path";
    }
    leaf verbatim {
      type empty;
      description
        "Indicates no topology or CSPF is attempted on the
         specified path.";
    }
    leaf lockdown {
      type empty;
      description
        "Indicates no reoptimization to be attempted for
         this path.";
    }
    leaf named-explicit-path {
      when "../path-computation-method = 'path-explicitly-defined'";
      type leafref {
        path "../../../../../../globals/named-explicit-paths/"
        + "named-explicit-path/config/name";
      }
      description "The explicit-path name";
    }
    leaf named-path-constraint {
      if-feature te-types:named-path-constraints;
      type leafref {
        path "../../../../../../globals/"
        + "named-path-constraints/named-path-constraint/"
        + "config/name";
      }
```

```
      description
        "Reference to a globally defined named path
        constraint set";
    }
  }

  /* TE tunnel configuration data */
  grouping tunnel-p2mp-params_config {
    description
      "Configuration parameters relating to TE tunnel";
    leaf name {
      type string;
      description "TE tunnel name.";
    }
    leaf type {
      type identityref {
        base te-types:tunnel-type;
      }
      description "TE tunnel type.";
    }
    leaf identifier {
      type uint16;
      description
        "TE tunnel Identifier.";
    }
    leaf description {
      type string;
      description
        "Textual description for this TE tunnel";
    }
    leaf setup-priority {
      type uint8 {
        range "0..7";
      }
      description
        "TE LSP setup priority";
    }
    leaf hold-priority {
      type uint8 {
        range "0..7";
      }
      description
        "TE LSP hold priority";
    }
    leaf lsp-protection-type {
      type identityref {
        base te-types:lsp-prot-type;
      }
```

```
      description "LSP protection type.";
    }
    leaf admin-status {
      type identityref {
        base te-types:state-type;
      }
      default te-types:state-up;
      description "TE tunnel administrative state.";
    }
    leaf reoptimize-timer {
      type uint16;
      units seconds;
      description
       "frequency of reoptimization of
        a traffic engineered LSP";
    }
  }

  grouping te-tunnel-bandwidth_config {
    description
      "Configuration parameters related to bandwidth for a tunnel";

    leaf specification-type {
      type te-mpls-types:te-bandwidth-type;
      default SPECIFIED;
      description
        "The method used for settign the bandwidth, either explicitly
        specified or configured";
    }

    leaf set-bandwidth {
      when "../specification-type = 'te-mpls-types:SPECIFIED'" {
       description
         "The bandwidth value when bandwidth is explicitly
          specified";
      }
      type te-mpls-types:bandwidth-kbps;
      description
       "set bandwidth explicitly, e.g., using
        offline calculation";
    }
    leaf class-type {
      type te-types:te-ds-class;
      description
        "The Class-Type of traffic transported by the LSP.";
      reference "RFC4124: section-4.3.1";
    }
  }
```

```
  grouping te-tunnel-bandwidth_state {
    description
      "Operational state parameters relating to bandwidth for a tunnel";

    leaf signaled-bandwidth {
      type te-mpls-types:bandwidth-kbps;
      description
        "The currently signaled bandwidth of the LSP. In the case where
        the bandwidth is specified explicitly, then this will match the
        value of the set-bandwidth leaf; in cases where the bandwidth is
        dynamically computed by the system, the current value of the
        bandwidth should be reflected.";
    }
  }

  grouping te-lsp-auto-bandwidth_config {
    description
      "Configuration parameters related to autobandwidth";

    leaf enabled {
      type boolean;
      default false;
      description
        "enables mpls auto-bandwidth on the
         lsp";
    }

    leaf min-bw {
      type te-mpls-types:bandwidth-kbps;
      description
        "set the minimum bandwidth in Kbps for an
         auto-bandwidth LSP";
    }

    leaf max-bw {
      type te-mpls-types:bandwidth-kbps;
      description
        "set the maximum bandwidth in Kbps for an
         auto-bandwidth LSP";
    }

    leaf adjust-interval {
      type uint32;
      description
        "time in seconds between adjustments to
         LSP bandwidth";
    }
```

```
    leaf adjust-threshold {
      type te-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";
    }
  }

  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 te-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 te-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-path-bandwidth_top {
    description
      "Top level grouping for specifying bandwidth for a TE path";

    container bandwidth {
      description
        "Bandwidth configuration for TE LSPs";

      container config {
        description
          "Configuration parameters related to bandwidth on TE
          tunnels:";
        uses te-tunnel-bandwidth_config;
      }

      container state {
        config false;
        description
          "State parameters related to bandwidth
          configuration of TE tunnels";
        uses te-tunnel-bandwidth_config;
        uses te-tunnel-bandwidth_state;
      }
    }
  }

  grouping te-tunnel-bandwidth_top {
    description
```

```
      "Top level grouping for specifying bandwidth for a tunnel";

    container bandwidth {
      description
        "Bandwidth configuration for TE LSPs";

      container config {
        description
          "Configuration parameters related to bandwidth on TE
          tunnels:";
        uses te-tunnel-bandwidth_config;
      }

      container state {
        config false;
        description
          "State parameters related to bandwidth
          configuration of TE tunnels";
        uses te-tunnel-bandwidth_config;
        uses te-tunnel-bandwidth_state;
      }

      container auto-bandwidth {
        when "../config/specification-type = 'AUTO'" {
          description
            "Include this container for auto bandwidth
            specific configuration";
        }
        description
          "Parameters related to auto-bandwidth";

        container config {
          description
            "Configuration parameters relating to MPLS
            auto-bandwidth on the tunnel.";
          uses te-lsp-auto-bandwidth_config;
        }
        container state {
          config false;
          description
            "State parameters relating to MPLS
            auto-bandwidth on the tunnel.";
          uses te-lsp-auto-bandwidth_config;
        }

        container overflow {
          description
            "configuration of MPLS overflow bandwidth
```

```
            adjustement for the LSP";

          container config {
            description
              "Config information for MPLS overflow bandwidth
              adjustment";
            uses te-lsp-overflow_config;
          }

          container state {
            config false;
            description
              "Config information for MPLS overflow bandwidth
              adjustment";
            uses te-lsp-overflow_config;
          }
        }

        container underflow {
          description
            "configuration of MPLS underflow bandwidth
            adjustement for the LSP";

          container config {
            description
              "Config information for MPLS underflow bandwidth
              adjustment";
            uses te-lsp-underflow_config;
          }

          container state {
            config false;
            description
              "State information for MPLS underflo
              adjustment";
            uses te-lsp-underflow_config;
          }
        }
      }
    }
  }

  grouping tunnel-p2p-params_config {
    description
      "Configuration parameters relating to TE tunnel";
    leaf name {
      type string;
      description "TE tunnel name.";
```

```
    }
    leaf type {
      type identityref {
        base te-types:tunnel-type;
      }
      description "TE tunnel type.";
    }
    leaf identifier {
      type uint16;
      description
        "TE tunnel Identifier.";
    }
    leaf description {
      type string;
      description
        "Textual description for this TE tunnel";
    }
    leaf encoding {
      type identityref {
        base te-types:lsp-encoding-types;
      }
      description "LSP encoding type";
    }
    leaf protection-type {
      type identityref {
        base te-types:lsp-prot-type;
      }
      description "LSP protection type.";
    }
    leaf admin-status {
      type identityref {
        base te-types:state-type;
      }
      default te-types:state-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";
    }
    leaf reoptimize-timer {
      type uint16;
      units seconds;
      description
```

```
       "frequency of reoptimization of
        a traffic engineered LSP";
    }
    leaf source {
      type inet:ip-address;
      description
        "TE tunnel source address.";
    }
    leaf destination {
      /* Add when check */
      type inet:ip-address;
      description
        "P2P tunnel destination address";
    }
    leaf src-tp-id {
      type binary;
      description
        "TE tunnel source termination point identifier.";
    }
    leaf dst-tp-id {
      /* Add when check */
      type binary;
      description
        "TE tunnel destination termination point identifier.";
    }
    leaf setup-priority {
      type uint8 {
        range "0..7";
      }
      description
        "TE LSP setup priority";
    }
    leaf hold-priority {
      type uint8 {
        range "0..7";
      }
      description
        "TE LSP hold priority";
    }
    leaf signaling-type {
      type identityref {
        base te-types:path-signaling-type;
      }
      description "TE tunnel path signaling type";
    }
    container hierarchical-link-id {
        description
          "Identifies a hierarchical link (in server layer)
```

```
           that this tunnel is associated with.";
        leaf local-te-node-id {
          type te-types:te-node-id;
          description
            "Local TE node identifier";
        }
        leaf local-te-link-tp-id {
          type te-types:te-tp-id;
          description
            "Local TE link termination point identifier";
        }
        leaf remote-te-node-id {
          type te-types:te-node-id;
          description
            "Remote TE node identifier";
        }
        leaf te-topology-id {
          type te-types: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.";
        }
    }
    uses bidir-assoc-properties;
  }

  grouping tunnel-p2p-params_state {
    description
      "State parameters relating to TE tunnel";
    leaf oper-status {
      type identityref {
        base te-types:state-type;
      }
      description "TE tunnel operational state.";
    }
  }

  /* TE tunnel configuration/state grouping */
  grouping tunnel-p2mp-properties {
    description
      "Top level grouping for P2MP tunnel properties.";
    container config {
      description
        "Configuration parameters relating to
         tunnel P2MP properties";
      uses tunnel-p2mp-params_config;
    }
```

```
    container state {
      config false;
      description
        "State information associated with tunnel
         properties";
      uses tunnel-p2mp-params_config;
    }
  }

  grouping p2p-path-candidate-secondary-path-config {
    description
      "Configuration parameters relating to a secondary path which
      is a candidate for a particular primary path";

    leaf secondary-path {
      type leafref {
        path "../../../../../../p2p-secondary-paths/" +
             "p2p-secondary-path/config/name";
      }
      description
        "A reference to the secondary path that should be utilised
        when the containing primary path option is in use";
    }

    leaf priority {
      type uint16;
      description
        "The priority of the specified secondary path option. Higher
        priority options are less preferable - such that a secondary
        path reference with a priority of 0 is the most preferred";
    }
    leaf path-setup-protocol {
      type identityref {
        base te-types:te-path-setup-protocol;
      }
      description
        "Signaling protocol used to set up this tunnel";
    }
  }

  grouping p2p-path-candidate-secondary-path-state {
    description
      "Operational state parameters relating to a secondary path
      which is a candidate for a particular primary path";

    leaf active {
      type boolean;
      description
```

```
        "Indicates the current active path option that has
        been selected of the candidate secondary paths";
    }
  }

  grouping tunnel-p2p-properties {
    description
      "Top level grouping for tunnel properties.";
    container config {
      description
        "Configuration parameters relating to
         tunnel properties";
      uses tunnel-p2p-params_config;
    }
    container state {
      config false;
      description
        "State information associated with tunnel
         properties";
      uses tunnel-p2p-params_config;
      uses tunnel-p2p-params_state;
    }
                uses te-tunnel-bandwidth_top;

    container p2p-primary-paths {
      description "Set of P2P primary aths container";
      list p2p-primary-path {
        key "name";
        description
          "List of primary paths for this tunnel.";
        leaf name {
          type leafref {
            path "../config/name";
          }
          description "TE path name";
        }
        uses p2p-primary-path-properties;
        container candidate-p2p-secondary-paths {
          description
            "The set of candidate secondary paths which may be used
            for this primary path. When secondary paths are specified
            in the list the path of the secondary LSP in use must be
            restricted to those path options referenced. The
            priority of the secondary paths is specified within the
            list. Higher priority values are less preferred - that is
            to say that a path with priority 0 is the most preferred
            path. In the case that the list is empty, any secondary
            path option may be utilised when the current primary path
```

```
            is in use.";
          list candidate-p2p-secondary-path {
            key "secondary-path";
            description
              "List of secondary paths for this tunnel.";
            leaf secondary-path {
              type leafref {
                path "../config/secondary-path";
              }
              description "TE path name";
            }
            container config {
              description
                "Configuration parameters relating to the candidate
                secondary path";

              uses p2p-path-candidate-secondary-path-config;
            }

            container state {
              config false;
              description
                "Operational state parameters relating to the candidate
                secondary path";

              uses p2p-path-candidate-secondary-path-config;
              uses p2p-path-candidate-secondary-path-state;
            }
          }
        }
      }
    }
    container p2p-secondary-paths {
      description "Set of P2P secondary paths container";
      list p2p-secondary-path {
        key "name";
        description
          "List of secondary paths for this tunnel.";
        leaf name {
          type leafref {
            path "../config/name";
          }
          description "TE path name";
        }
        uses p2p-primary-path-properties;
      }
    }
  }
```

```
  grouping tunnel-actions {
    description "Tunnel actions";
    action tunnel-action {
      description "Tunnel action";
      input {
        leaf action-type {
          type identityref {
            base te-types:tunnel-action-type;
          }
          description "Tunnel action type";
        }
      }
      output {
        leaf action-result {
          type identityref {
            base te-types:te-action-result;
          }
          description "The result of the RPC operation";
        }
      }
    }
  }
  /*** End of TE tunnel groupings ***/

  /**
   * LSP related generic groupings
   */
  grouping lsp-record-route-information_state {
    description "recorded route information grouping";
    container lsp-record-route-subobjects {
      description "RSVP recorded route object information";
      list record-route-subobject {
        when "../../origin-type = 'ingress'" {
          description "Applicable on non-ingress LSPs only";
        }
        key "index";
        description "Record route sub-object list";
        leaf index {
          type uint32;
          description "RRO subobject index";
        }
        uses te-types:record-route-subobject;
      }
    }
  }

  grouping lsp-properties_state {
    description
```

```
      "State parameters relating to LSP";
    leaf oper-status {
      type identityref {
        base te-types:state-type;
      }
      description "LSP operational state.";
    }
    leaf path-setup-protocol {
      type identityref {
        base te-types:te-path-setup-protocol;
      }
      description
        "Signaling protocol used to set up this tunnel";
    }
    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-role {
      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-carry-normal-traffic {
      type empty;
      description
        "This bit is set when a protecting LSP is carrying the normal
         traffic after protection switching";
    }
  }
  /*** End of TE LSP groupings ***/

  /**
   * TE global generic groupings
   */

  /* Global named admin-groups configuration data */
  grouping named-admin-groups_config {
    description
      "Global named administrative groups configuration
      grouping";
    leaf name {
      type string;
      description
        "A string name that uniquely identifies a TE
        interface named admin-group";
```

```
    }
    leaf bit-position {
      type uint32;
      description
        "Bit position representing the administrative group";
    }
  }
  grouping named-admin-groups {
    description
      "Global named administrative groups configuration
      grouping";
    container named-admin-groups {
      description "TE named admin groups container";
      list named-admin-group {
        if-feature te-types:extended-admin-groups;
        if-feature te-types:named-extended-admin-groups;
        key "name";
        description
          "List of named TE admin-groups";
        leaf name {
          type leafref {
            path "../config/name";
          }
          description "Admin-group name";
        }
        container config {
          description
            "Configuration parameters related to admin-groups";
          uses named-admin-groups_config;
        }
        container state {
          config false;
          description
            "State parameters related to admin-groups";
          uses named-admin-groups_config;
        }
      }
    }
  }

  /* Global named admin-srlgs configuration data */
  grouping named-srlgs_config {
    description
      "Global named SRLGs configuration grouping";
    leaf name {
      type string;
      description
        "A string name that uniquely identifies a TE
```

```
        interface named srlg";
    }
    leaf group {
      type te-types:srlg;
      description "An SRLG value";
    }
    leaf cost {
      type uint32;
      description
        "SRLG associated cost. Used during path to append
         the path cost when traversing a link with this SRLG";
    }
  }

  grouping named-srlgs {
    description
      "Global named SRLGs configuration grouping";
    container named-srlgs {
      description "TE named SRLGs container";
      list named-srlg {
        if-feature te-types:named-srlg-groups;
        key "name";
        description
          "A list of named SRLG groups";
        leaf name {
          type leafref {
            path "../config/name";
          }
          description "SRLG name";
        }
        container config {
          description
            "Configuration parameters related to named SRLGs";
          uses named-srlgs_config;
        }
        container state {
          config false;
          description
            "State parameters related to named SRLGs";
          uses named-srlgs_config;
        }
      }
    }
  }

  /* Global named explicit-paths configuration data */
  grouping named-explicit-paths_config {
    description
```

```
      "Global explicit path configuration
      grouping";
    leaf name {
      type string;
      description
        "A string name that uniquely identifies an
        explicit path";
    }
    container explicit-route-objects {
      description "Explicit route objects container";
      list explicit-route-object {
        key "index";
        description
          "List of explicit route objects";
        leaf index {
          type leafref {
            path "../config/index";
          }
          description
            "Index of this explicit route object";
        }
        leaf explicit-route-usage {
          type identityref {
            base te-types:route-usage-type;
          }
          description "An explicit-route hop action.";
        }
        uses te-types:explicit-route-hop;
      }
    }
  }

  grouping named-explicit-paths {
    description
      "Global explicit path configuration
      grouping";
    container named-explicit-paths {
      description "TE named explicit path container";
      list named-explicit-path {
        key "name";
        description
          "A list of explicit paths";
        leaf name {
          type leafref {
            path "../config/name";
          }
          description "Explicit-path name";
        }
```

```
        container config {
          description
            "Configuration parameters related to named paths";
          uses named-explicit-paths_config;
        }
        container state {
          config false;
          description
            "State parameters related to named paths";
          uses named-explicit-paths_config;
        }
      }
    }
  }

  /* Global named paths constraints configuration data */
  grouping named-path-constraints_config {
    description
      "Global named path constraints configuration
      grouping";
    leaf name {
      type string;
      description
        "A string name that uniquely identifies a
        path constraint set";
    }
    leaf topology-id {
      type te-types:te-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 te-types:path-metric-type;
      }
      default te-types:path-metric-te;
      description
        "The tunnel path metric type.";
    }
    leaf tiebreaker-type {
      type identityref {
        base te-types:path-tiebreaker-type;
      }
      default te-types:path-tiebreaker-maxfill;
      description
        "The tunnel path computation tie breakers.";
    }
    leaf ignore-overload {
      type boolean;
      description
        "The tunnel path can traverse overloaded node.";
    }
    leaf setup-priority {
      type uint8 {
        range "0..7";
      }
      description
        "TE LSP setup priority";
    }
    leaf hold-priority {
      type uint8 {
        range "0..7";
      }
      description
        "TE LSP hold priority";
    }
    uses path-affinities;
    uses path-srlgs;
    uses path-route-objects;
    uses te-path-bandwidth_top;
  }

  grouping named-path-constraints {
    description
      "Global named path constraints configuration
      grouping";
    container named-path-constraints {
      description "TE named path constraints container";
      list named-path-constraint {
        if-feature te-types:named-path-constraints;
        key "name";
```

```
        description
          "A list of named path constraints";
        leaf name {
          type leafref {
            path "../config/name";
          }
          description "Path constraint name";
        }
        container config {
          description
            "Configuration parameters related to admin-groups";
          uses named-path-constraints_config;
        }
        container state {
          config false;
          description
            "State parameters related to admin-groups";
          uses named-path-constraints_config;
        }
      }
    }
  }

  /* TE globals container data */
  grouping globals-grouping {
    description
      "Globals TE system-wide configuration data grouping";
    container globals {
      description
        "Globals TE system-wide configuration data container";
      uses named-admin-groups;
      uses named-srlgs;
      uses named-explicit-paths;
      uses named-path-constraints;
    }
  }

  /* TE tunnels container data */
  grouping tunnels-grouping {
    description
      "Tunnels TE configuration data grouping";
    container tunnels {
      description
        "Tunnels TE configuration data container";

      list tunnel {
        key "name";
        unique "identifier";
```

```
        description "P2P TE tunnels list.";
        leaf name {
          type leafref {
            path "../config/name";
          }
          description "TE tunnel name.";
        }
        leaf identifier {
          type leafref {
            path "../config/identifier";
          }
          description
            "TE tunnel Identifier.";
        }
        uses tunnel-p2p-properties;
        uses tunnel-actions;
      }
      list tunnel-p2mp {
        key "name";
        unique "identifier";
        description "P2MP TE tunnels list.";
        leaf name {
          type leafref {
            path "../config/name";
          }
          description "TE tunnel name.";
        }
        leaf identifier {
          type leafref {
            path "../config/identifier";
          }
          description
            "TE tunnel Identifier.";
        }
        uses tunnel-p2mp-properties;
      }
    }
  }

  /* TE LSPs ephemeral state container data */
  grouping lsps-state-grouping {
    description
      "LSPs state operational data grouping";
    container lsps-state {
      config "false";
      description "LSPs 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 te-types:tunnel-type;
          }
          description "The LSP type P2P or P2MP";
        }
```

```
        uses lsp-properties_state;
        uses lsp-record-route-information_state;
      }
    }
  }
  /*** End of TE global groupings ***/

  /**
   * TE configurations container
   */
  container te {
    presence "Enable TE feature.";
    description
       "TE global container.";

    /* TE Global Configuration Data */
    uses globals-grouping;

    /* TE Tunnel Configuration Data */
    uses tunnels-grouping;

    /* TE LSPs State Data */
    uses lsps-state-grouping;
  }

  /* 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";
    input {
      container tunnel-info {
        description "Tunnel Identification";
        choice type {
          description "Tunnel information type";
          case tunnel-p2p {
            leaf p2p-id {
              type te:tunnel-ref;
```

```
              description "P2P TE tunnel";
            }
          }
          case tunnel-p2mp {
            leaf p2mp-id {
              type te:tunnel-p2mp-ref;
              description "P2MP TE tunnel";
            }
          }
        }
      }
    }
    output {
      container result {
        description
          "The container result of the RPC operation";
        leaf result {
          type enumeration {
            enum success {
              description "Origin ingress";
            }
            enum in-progress {
              description "Origin egress";
            }
            enum fail {
              description "transit";
            }
          }
          description "The result of the RPC operation";
        }
      }
    }
  }

  /* TE Global Notification Data */
  notification globals-notif {
    description
      "Notification messages for Global TE.";
  }

  /* TE Tunnel Notification Data */
  notification tunnels-notif {
    description
      "Notification messages for TE tunnels.";
  }
}
<CODE ENDS>
```

Figure 8: TE generic YANG module

```
<CODE BEGINS> file "ietf-te-device@2017-03-10.yang"
module ietf-te-device {

  namespace "urn:ietf:params:xml:ns:yang:ietf-te-device";

  /* Replace with IANA when assigned */
  prefix "te-dev";

  /* Import TE generic types */
  import ietf-te {
    prefix te;
  }

  /* Import TE generic types */
  import ietf-te-types {
    prefix te-types;
  }

  import ietf-interfaces {
    prefix if;
  }

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-routing-types {
    prefix "rt-types";
  }

  organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
     Working Group";

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

     WG Chair: Lou Berger
               <mailto:lberger@labn.net>

     WG Chair: Vishnu Pavan Beeram
               <mailto:vbeeram@juniper.net>

     Editor:   Tarek Saad
               <mailto:tsaad@cisco.com>
```

```
        Editor:   Rakesh Gandhi
                  <mailto:rgandhi@cisco.com>

        Editor:   Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

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

        Editor:   Xufeng Liu
                  <mailto:xufeng.liu@ericsson.com>

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

        Editor:   Raqib Jones
                  <mailto:raqib@Brocade.com>

        Editor:   Bin Wen
                  <mailto:Bin_Wen@cable.comcast.com>";

    description
      "YANG data module for TE device configurations,
      state, RPC and notifications.";

    revision "2017-03-10" {
      description "Latest update to TE device YANG module.";
      reference "TBD";
    }


    /**
     * TE LSP device state grouping
     */
    grouping lsps-device_state {
      description "TE LSP device state grouping";
      container lsp-timers {
        when "../te: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-destroy {
            type uint32;
            units seconds;
            description
              "lsp expiration delay time";
          }
        }

        container downstream-info {
          when "../te:origin-type != 'egress'" {
            description "Applicable to ingress LSPs only";
          }
          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 rt-types:generalized-label;
            description
              "downstream label.";
          }
        }

        container upstream-info {
```

```
        when "../te:origin-type != 'ingress'" {
          description "Applicable to non-ingress LSPs only";
        }
        description
          "upstream information";

        leaf phop {
          type inet:ip-address;
          description
            "upstream nexthop or previous-hop.";
        }

        leaf neighbor {
          type inet:ip-address;
          description
            "upstream neighbor.";
        }

        leaf label {
          type rt-types:generalized-label;
          description
            "upstream label.";
        }
      }
    }

    /**
     * Device general groupings.
     */
    grouping tunnel-device_config {
      description "Device TE tunnel configs";
      leaf path-invalidation-action {
        type identityref {
          base te-types:path-invalidation-action-type;
        }
        description "Tunnel path invalidtion action";
      }
    }

    grouping lsp-device-timers_config {
      description "Device TE LSP timers configs";
      leaf lsp-install-interval {
        type uint32;
        units seconds;
        description
          "lsp installation delay time";
      }
      leaf lsp-cleanup-interval {
```

```
        type uint32;
        units seconds;
        description
          "lsp cleanup delay time";
      }
      leaf lsp-invalidation-interval {
        type uint32;
        units seconds;
        description
          "lsp path invalidation before taking action delay time";
      }
    }
    grouping lsp-device-timers {
      description "TE LSP timers configuration";
      container config {
        description
          "Configuration parameters for TE LSP timers";
        uses lsp-device-timers_config;

      }
      container state {
        config false;
        description
          "State parameters for TE LSP timers";
        uses lsp-device-timers_config;
      }
    }

    /**
     * TE global device generic groupings
     */

    /* TE interface container data */
    grouping interfaces-grouping {
      description
        "Interface TE configuration data grouping";
      container interfaces {
        description
          "Configuration data model for TE interfaces.";
        uses te-all-attributes;
        list interface {
          key "interface";
          description "TE interfaces.";
          leaf interface {
            type if:interface-ref;
            description
              "TE interface name.";
          }
```

```
          /* TE interface parameters */
          uses te-attributes;
        }
      }
    }

    /**
     * TE interface device 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 admin-groups {
              description
                "Administrative group/Resource
                class/Color.";
              leaf admin-group {
                type te-types:admin-group;
                description
                  "TE interface administrative group";
              }
            }
            case extended-admin-groups {
              if-feature te-types:extended-admin-groups;
              description
                "Extended administrative group/Resource
                class/Color.";
              leaf extended-admin-group {
                type te-types:extended-admin-group;
                description
                  "TE interface extended administrativei
                  group";
              }
            }
          }
        }
        case named-admin-groups {
          list named-admin-groups {
            if-feature te-types:extended-admin-groups;
            if-feature 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/" +
                  "te:named-admin-groups/te:named-admin-group/" +
                  "te:config/te:name";
              }
              description "A named admin-group entry";
            }
          }
        }
      }
    }

    /* 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 te-types:named-srlg-groups;
            key named-srlg;
            description
              "A list of named SRLG entries";
            leaf named-srlg {
              type leafref {
                path "../../../../../te:globals/" +
                  "te:named-srlgs/te:named-srlg/te:config/te:name";
              }
              description
                "A named SRLG entry";
            }
```

```
          }
        }
      }
    }

    grouping 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 te-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 te-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 te-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 te-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";
      }
    }

    /* TE interface metric */
    grouping te-metric_config {
      description "Interface TE metric grouping";
      leaf te-metric {
        type te-types:te-metric;
        description "Interface TE metric.";
```

```
      }
    }

    /* 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 te-types:switching-capabilities;
          }
          description
            "Switching Capability for this interface";
        }
        leaf encoding {
          type identityref {
            base te-types:lsp-encoding-types;
          }
          description
            "Encoding supported by this interface";
        }
      }
    }

    grouping te-advertisements_state {
      description
        "TE interface advertisements state grouping";
      container te-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";
            }
          }
        }
      }
```

```
     /* TE interface attributes grouping */
     grouping te-attributes {
       description "TE attributes configuration grouping";
       container config {
         description
           "Configuration parameters for interface TE
            attributes";
         uses te-metric_config;
         uses te-admin-groups_config;
         uses te-srlgs_config;
                           uses te-igp-flooding-bandwidth_config;
         uses te-switching-cap_config;
       }
       container state {
         config false;
         description
           "State parameters for interface TE metric";
         uses te-metric_config;
         uses te-admin-groups_config;
         uses te-srlgs_config;
         uses te-switching-cap_config;
                           uses te-igp-flooding-bandwidth_config;
         uses te-advertisements_state;
       }
     }

     grouping te-all-attributes {
       description
         "TE attributes configuration grouping for all
          interfaces";
       container config {
         description
           "Configuration parameters for all interface TE
            attributes";
                           uses te-igp-flooding-bandwidth_config;
       }
       container state {
         config false;
         description
           "State parameters for all interface TE metric";
                           uses te-igp-flooding-bandwidth_config;
       }
     }
     /*** End of TE interfaces device groupings ***/


     /**
      * TE device augmentations
```

```
     */
    augment "/te:te" {
      description "TE global container.";
      /* TE Interface Configuration Data */
      uses interfaces-grouping;
    }

    /* TE globals device augmentation */
    augment "/te:te/te:globals" {
      description
        "Global TE device specific configuration parameters";
      uses lsp-device-timers;
    }

    /* TE tunnels device configuration augmentation */
    augment "/te:te/te:tunnels/te:tunnel/te:config" {
      description
        "Tunnel device dependent augmentation";
      uses lsp-device-timers_config;
    }
    augment "/te:te/te:tunnels/te:tunnel/te:state" {
      description
        "Tunnel device dependent augmentation";
      uses lsp-device-timers_config;
    }

    /* TE LSPs device state augmentation */
    augment "/te:te/te:lsps-state/te:lsp" {
      description
        "LSP device dependent augmentation";
      uses lsps-device_state;
    }

    /* TE interfaces RPCs/execution Data */
    rpc interfaces-rpc {
      description
        "Execution data for TE interfaces.";
    }

    /* TE Interfaces Notification Data */
    notification interfaces-notif {
      description
        "Notification messages for TE interfaces.";
    }
  }
  <CODE ENDS>
```

Figure 9: TE MPLS specific types YANG module

```
<CODE BEGINS> file "ietf-te-mpls@2017-03-10.yang"
module ietf-te-mpls {

  namespace "urn:ietf:params:xml:ns:yang:ietf-te-mpls";

  /* Replace with IANA when assigned */
  prefix "te-mpls";

  /* Import TE generic types */
  import ietf-te {
    prefix te;
  }

  /* Import TE generic types */
  import ietf-te-types {
    prefix te-types;
  }

  import ietf-routing-types {
    prefix "rt-types";
  }

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

  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
     Working Group";

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

     WG Chair: Lou Berger
               <mailto:lberger@labn.net>

     WG Chair: Vishnu Pavan Beeram
               <mailto:vbeeram@juniper.net>

     Editor:   Tarek Saad
               <mailto:tsaad@cisco.com>

     Editor:   Rakesh Gandhi
```

```
                <mailto:rgandhi@cisco.com>

        Editor:   Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

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

        Editor:   Xufeng Liu
                  <mailto:xufeng.liu@ericsson.com>

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

        Editor:   Raqib Jones
                  <mailto:raqib@Brocade.com>

        Editor:   Bin Wen
                  <mailto:Bin_Wen@cable.comcast.com>";

    description
      "YANG data module for MPLS TE configurations,
      state, RPC and notifications.";

    revision "2017-03-10" {
      description "Latest update to MPLS TE YANG module.";
      reference "TBD";
    }

    /* MPLS TE tunnel properties*/

    grouping tunnel-igp-shortcut_config {
      description "TE tunnel IGP shortcut configs";
      leaf shortcut-eligible {
        type boolean;
        default "true";
        description
          "Whether this LSP is considered to be eligible for us as a
          shortcut in the IGP. In the case that this leaf is set to
          true, the IGP SPF calculation uses the metric specified to
          determine whether traffic should be carried over this LSP";
      }
      leaf metric-type {
        type identityref {
          base te-types:LSP_METRIC_TYPE;
        }
        default te-types:LSP_METRIC_INHERITED;
        description
```

```
            "The type of metric specification that should be used to set
            the LSP(s) metric";
        }
        leaf metric {
          type int32;
          description
            "The value of the metric that should be specified. The value
            supplied in this leaf is used in conjunction with the metric
            type to determine the value of the metric used by the system.
            Where the metric-type is set to LSP_METRIC_ABSOLUTE - the
            value of this leaf is used directly; where it is set to
            LSP_METRIC_RELATIVE, the relevant (positive or negative)
            offset is used to formulate the metric; where metric-type
            is LSP_METRIC_INHERITED, the value of this leaf is not
            utilised";
        }
        leaf-list routing-afs {
          type inet:ip-version;
          description
            "Address families";
        }
      }

      grouping tunnel-igp-shortcuts {
        description
          "TE tunnel IGP shortcut grouping";
        container tunnel-igp-shortcut {
          description
            "Tunnel IGP shortcut properties";
          container config {
            description
              "Configuration parameters for tunnel IGP shortcuts";
            uses tunnel-igp-shortcut_config;
          }
          container state {
            description
              "State parameters for tunnel IGP shortcuts";
            uses tunnel-igp-shortcut_config;
          }
        }
      }

      grouping tunnel-forwarding-adjacency_configs {
        description "Tunnel forwarding adjacency grouping";
        leaf binding-label {
          type rt-types:mpls-label;
          description "MPLS tunnel binding label";
        }
```

```
      leaf load-share {
        type uint32 {
          range "1..4294967295";
        }
        description "ECMP tunnel forwarding
          load-share factor.";
      }
      leaf policy-class {
        type uint8 {
          range "1..7";
        }
        description
          "The class associated with this tunnel";
      }
    }

    grouping tunnel-forwarding-adjacency {
      description "Properties for using tunnel in forwarding.";
      container forwarding {
        description
          "Tunnel forwarding properties container";
        container config {
          description
            "Configuration parameters for tunnel forwarding adjacency";
          uses tunnel-forwarding-adjacency_configs;
        }
        container state {
          description
            "State parameters for tunnel forwarding adjacency";
          uses tunnel-forwarding-adjacency_configs;
        }
      }
    }

    /*** End of MPLS TE tunnel configuration/state */


    /**
     * MPLS TE augmentations
     */

    /* MPLS TE tunnel augmentations */
    augment "/te:te/te:tunnels/te:tunnel" {
      description "MPLS TE tunnel config augmentations";
      uses tunnel-igp-shortcuts;
      uses tunnel-forwarding-adjacency;
    }
```

```
     /* MPLS TE LSPs augmentations */
     augment "/te:te/te:tunnels/te:tunnel/" +
             "te:p2p-primary-paths/te:p2p-primary-path/" +
             "te:config" {
       when "/te:te/te:tunnels/te:tunnel" +
         "/te:p2p-primary-paths/te:p2p-primary-path/te:config" +
         "/te:path-setup-protocol = 'te-types:te-path-setup-static'" {
         description
         "When the path is statically provisioned";
       }
       description "MPLS TE LSP augmentation";
       leaf static-lsp-name {
         type mpls-static:static-lsp-ref;
         description "Static LSP name";
       }
     }
     augment "/te:te/te:tunnels/te:tunnel/" +
             "te:p2p-primary-paths/te:p2p-primary-path/" +
             "te:state" {
       description "MPLS TE LSP augmentation";
       leaf static-lsp-name {
         type mpls-static:static-lsp-ref;
         description "Static LSP name";
       }
     }
     augment "/te:te/te:tunnels/te:tunnel/" +
             "te:p2p-secondary-paths/te:p2p-secondary-path/" +
             "te:config" {
       when "/te:te/te:tunnels/te:tunnel" +
         "/te:p2p-secondary-paths/te:p2p-secondary-path/te:config" +
         "/te:path-setup-protocol = 'te-types:te-path-setup-static'" {
         description
         "When the path is statically provisioned";
       }
       description "MPLS TE LSP augmentation";
       leaf static-lsp-name {
         type mpls-static:static-lsp-ref;
         description "Static LSP name";
       }
     }
     augment "/te:te/te:tunnels/te:tunnel/" +
             "te:p2p-secondary-paths/te:p2p-secondary-path/" +
             "te:state" {
       description "MPLS TE LSP augmentation";
       leaf static-lsp-name {
         type mpls-static:static-lsp-ref;
         description "Static LSP name";
       }
```

```
    }
  }
  <CODE ENDS>
```

Figure 10: TE MPLS YANG module

```
  <CODE BEGINS> file "ietf-te-mpls-types@2017-03-10.yang"
  module ietf-te-mpls-types {

    namespace "urn:ietf:params:xml:ns:yang:ietf-te-mpls-types";

    /* Replace with IANA when assigned */
    prefix "te-mpls-types";

    organization
      "IETF TEAS Working Group";

    contact "Fill me";

    description
      "This module contains a collection of generally
      useful TE specific YANG data type defintions.";

    revision "2017-03-10" {
      description "Latest revision of TE MPLS types";
      reference "RFC3209";
    }

    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.";
    }

    typedef bandwidth-kbps {
      type uint64;
      units "Kbps";
      description
        "Bandwidth values expressed in kilobits per second";
    }

    typedef bandwidth-mbps {
      type uint64;
      units "Mbps";
      description
        "Bandwidth values expressed in megabits per second";
    }

    typedef bandwidth-gbps {
      type uint64;
      units "Gbps";
      description
        "Bandwidth values expressed in gigabits per second";
    }

    typedef te-bandwidth-type {
      type enumeration {
        enum SPECIFIED {
          description
```

```
            "Bandwidth is explicitly specified";
        }
        enum AUTO {
          description
            "Bandwidth is automatically computed";
        }
      }
      description
        "enumerated type for specifying whether bandwidth is
         explicitly specified or automatically computed";
    }

    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.";
    }
  }
  <CODE ENDS>
```

Figure 11: TE MPLS types YANG module

```
  <CODE BEGINS> file "ietf-te-sr-mpls@2017-03-10.yang"
```

```
   module ietf-te-sr-mpls {

     namespace "urn:ietf:params:xml:ns:yang:ietf-te-sr-mpls";

     /* Replace with IANA when assigned */
     prefix "te-sr-mpls";

     /* Import TE generic types */
     import ietf-te {
       prefix te;
     }

     /* Import TE generic types */
     import ietf-te-types {
       prefix te-types;
     }

     organization
       "IETF Traffic Engineering Architecture and Signaling (TEAS)
        Working Group";

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

        WG Chair: Lou Berger
                  <mailto:lberger@labn.net>

        WG Chair: Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Tarek Saad
                  <mailto:tsaad@cisco.com>

        Editor:   Rakesh Gandhi
                  <mailto:rgandhi@cisco.com>

        Editor:   Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

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

        Editor:   Xufeng Liu
                  <mailto:xufeng.liu@ericsson.com>

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

```
        Editor:   Raqib Jones
                  <mailto:raqib@Brocade.com>

        Editor:   Bin Wen
                  <mailto:Bin_Wen@cable.comcast.com>";

    description
      "YANG data module for MPLS TE configurations,
      state, RPC and notifications.";

    revision "2017-03-10" {
      description "Latest update to MPLS TE YANG module.";
      reference "TBD";
    }

    identity sr-protection-type {
      description
         "The Adj-SID base protection types";
    }

    identity sr-protection-type-protected {
      base sr-protection-type;
      description
         "The Adj-SID is eligible if protected";
    }

    identity sr-protection-type-unprotected {
      base sr-protection-type;
      description
         "The Adj-SID is eligible if unprotected";
    }

    identity sr-protection-type-any {
      base sr-protection-type;
      description
         "The Adj-SID is eligible if protected or unprotected";
    }

    typedef te-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";
        }
      }
      description "SID selection mode type";
    }

    /* MPLS SR-TE tunnel properties*/
    grouping tunnel-sr-mpls-properties_config {
      description "MPLS TE SR tunnel properties";
      leaf path-signaling-type {
        type identityref {
          base te-types:path-signaling-type;
        }
        description "TE tunnel path signaling type";
      }
    }

    grouping te-sr-named-path-constraints_config {
      description
        "Configuration parameters relating to SR-TE LSPs";

      leaf sid-selection-mode {
        type te-sid-selection-mode;
        default MIXED_MODE;
        description
          "The restrictions placed on the SIDs to be selected by the
          calculation method for the explicit path when it is
          instantiated for a SR-TE LSP";
      }

      leaf sid-protection {
        type identityref {
          base sr-protection-type;
        }
        default sr-protection-type-any;
        description
          "When set to protected only SIDs that are
          protected are to be selected by the calculating method
          when the explicit path is instantiated by a SR-TE LSP.";
      }
    }

    grouping te-sr-named-path-constraints {
      description "Named TE SR path constraints grouping";
      container config {
        description
          "Configuration parameters related to TE SR named
```

```
            path constraints";
          uses te-sr-named-path-constraints_config;
        }
        container state {
          config false;
          description
            "State parameters related to TE SR named
             path constraints";
          uses te-sr-named-path-constraints_config;
        }
      }

      /*** End of MPLS SR-TE tunnel configuration/state */


      /**
       * MPLS TE augmentations
       */

      /* MPLS TE global augmentations */
      augment "/te:te/te:globals/te:named-path-constraints" +
              "/te:named-path-constraint" {
        description
            "Augmentations for MPLS SR-TE config named constraints";
        uses te-sr-named-path-constraints;
      }

      /* MPLS TE tunnel augmentations */

      /* MPLS TE LSPs augmentations */
    }
    <CODE ENDS>
```

Figure 12: SR TE MPLS 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-device XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-te-mpls XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-te-sr-mpls 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-mpls-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-device namespace: urn:ietf:params:xml:ns:yang:ietf-te prefix: ietf-te-device reference: RFC3209

name: ietf-te-mpls namespace: urn:ietf:params:xml:ns:yang:ietf-te-mpls prefix: ietf-te-mpls reference: RFC3209

name: ietf-te-sr-mpls namespace: urn:ietf:params:xml:ns:yang:ietf-te-sr-mpls prefix: ietf-te-sr-mpls

name: ietf-te-types namespace: urn:ietf:params:xml:ns:yang:ietf-te-types prefix: ietf-te-types reference: RFC3209

name: ietf-te-mpls-types namespace: urn:ietf:params:xml:ns:yang:ietf-te-mpls-types prefix: ietf-te-mpls-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/lsps-state": This list specifies the state derived LSPs. 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 the members of the multi-vendor YANG design team who are involved in the definition of this model.

The authors would also like to thank Loa Andersson, Lou Berger, Sergio Belotti, Italo Busi, Aihua Guo, Dhruv Dhody, Anurag Sharma, and Xian Zhang for their comments and providing valuable feedback on this document.

8. Contributors

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

9. References

9.1. Normative References

[I-D.ietf-teas-yang-rsvp]
Beeram, V., Saad, T., Gandhi, R., Liu, X., Shah, H., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Resource Reservation Protocol (RSVP)", draft-ietf-teas-yang-rsvp-06 (work in progress), October 2016.

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

[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>.

[RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, DOI 10.17487/RFC3473, January 2003, <http://www.rfc-editor.org/info/rfc3473>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6107] Shiomoto, K., Ed. and A. Farrel, Ed., "Procedures for Dynamically Signaled Hierarchical Label Switched Paths", RFC 6107, DOI 10.17487/RFC6107, February 2011, <http://www.rfc-editor.org/info/rfc6107>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <http://www.rfc-editor.org/info/rfc8040>.

9.2. Informative References

[I-D.clemm-netmod-mount]
Clemm, A., Medved, J., and E. Voit, "Mounting YANG-Defined Information from Remote Datastores", draft-clemm-netmod-mount-05 (work in progress), September 2016.

[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-02 (work in progress), October 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.

Authors' Addresses

Tarek Saad (editor)
Cisco Systems Inc

Email: tsaad@cisco.com

Rakesh Gandhi
Cisco Systems Inc

Email: rgandhi@cisco.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Himanshu Shah
Ciena

Email: hshah@ciena.com

Igor Bryskin
Huawei Technologies

Email: Igor.Bryskin@huawei.com

TEAS Working Group
Internet Draft
Intended status: Standards Track

Xufeng Liu
Jabil
Igor Bryskin
Huawei Technologies
Vishnu Pavan Beeram
Juniper Networks
Tarek Saad
Cisco Systems Inc
Himanshu Shah
Ciena
Oscar Gonzalez De Dios
Telefonica

Expires: December 12, 2017 June 12, 2017

YANG Data Model for TE Topologies
draft-ietf-teas-yang-te-topo-09

Status of this Memo

Copyright Notice

Abstract

This document defines a YANG data model for representing, retrieving and manipulating TE Topologies. The model serves as a base model that other technology specific TE Topology models can augment.

Conventions used in this document

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

Table of Contents

1. Introduction

The Traffic Engineering Database (TED) is an essential component of Traffic Engineered (TE) systems that are based on MPLS-TE [RFC2702] and GMPLS [RFC3945]. The TED is a collection of all TE information about all TE nodes and TE links in the network. The TE Topology is a schematic arrangement of TE nodes and TE links present in a given TED. There could be one or more TE Topologies present in a given Traffic Engineered system. The TE Topology is the topology on which path computational algorithms are run to compute Traffic Engineered Paths (TE Paths).

This document defines a YANG [RFC6020] 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. Terminology

TED: The Traffic Engineering Database is a collection of all TE information about all TE nodes and TE links in a given network.

TE-Topology: The TE Topology is a schematic arrangement of TE nodes and TE links in a given TED. It forms the basis for a graph suitable for TE path computations.

Native TE Topology: Native TE Topology is a topology that is native to a given provider network. Native TE topology could be discovered via various routing protocols and/or subscribe/publish techniques. This is the topology on which path computational algorithms are run to compute TE Paths.

Customized TE Topology: Customized TE Topology is a custom topology that is produced by a provider for a given Client. This topology typically augments the Client's Native TE Topology. Path computational algorithms aren't typically run on the Customized TE Topology; they are run on the Client's augmented Native TE Topology.

1.2. Tree Structure - Legend

A simplified graphical representation of the data model is presented in Section 6. 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.

1.3. 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

2. Characterizing TE Topologies

The data model proposed by this document takes the following characteristics of TE Topologies into account:

- TE Topology is an abstract control-plane representation of the data-plane topology. Hence attributes specific to the data-plane must make their way into the corresponding TE Topology modeling. The TE Topology comprises of dynamic auto-discovered data (data that may change frequently - example: unreserved bandwidth available on data-plane links) as well as fairly static data (data that rarely changes- examples: layer network identification, switching and adaptation capabilities and limitations, fate sharing, administrative colors) associated with data-plane nodes and links. It is possible for a single TE Topology to encompass TE information at multiple switching layers.

- TE Topologies are protocol independent. Information about topological elements may be learnt via link-state protocols, but the topology can exist without being dependent on any particular protocol.

- TE Topology may not be congruent to the routing topology (topology constructed based on routing adjacencies) in a given TE System. There isn't always a one-to-one association between a TE-link and a routing adjacency. For example, the presence of a TE link between a pair of nodes doesn't necessarily imply the existence of a routing-adjacency between these nodes.

- Each TE Topological element has an information source associated with it. In some scenarios, there could be more than one information source associated with each topological element.

- TE Topologies can be hierarchical. Each node and link of a given TE Topology can be associated with respective underlay topology. This means that each node and link of a given TE Topology can be associated with an independent stack of supporting TE Topologies.

- TE Topologies can be customized. TE topologies of a given network presented by the network provider to its client could be customized on per-client request basis. This customization could be performed by provider, by client or by provider/client negotiation. The relationship between a customized topology (as presented to the client) and provider's native topology (as known in its entirety to the provider itself) could be captured as hierarchical (overlay-underlay), but otherwise the two topologies are decoupled from each other.

3. Modeling Abstractions and Transformations

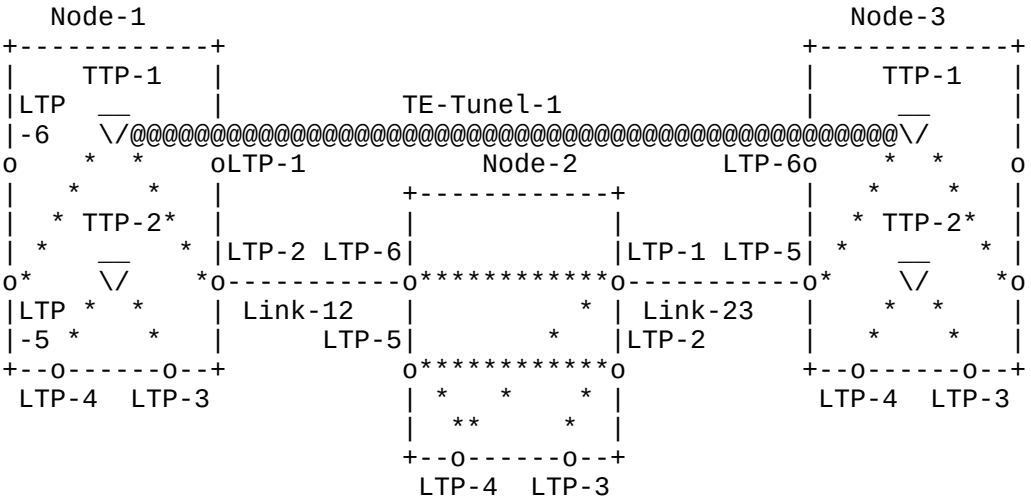


Figure 1: TE Topology Modeling Abstractions

3.1. TE Topology

TE topology is a traffic engineering representation of one or more layers of network topologies. TE topology is comprised of TE nodes (TE graph vertices) interconnected via TE links (TE graph edges). A TE topology is mapped to a TE graph.

3.2. TE Node

TE node is an element of a TE topology (presented as a vertex on TE graph). TE node represents one or several nodes (physical switches), or a fraction of a node. TE node belongs to and is fully defined in exactly one TE topology. TE node is assigned with the TE topology scope unique ID. TE node attributes include information related to the data plane aspects of the associated node(s) (e.g. connectivity matrix), as well as configuration data (such as TE node name). A given TE node can be reached on the TE graph over one of TE links terminated by the TE node.

Multi-layer TE nodes providing switching functions at multiple network layers are an example where a physical node can be decomposed into multiple logical TE nodes (fractions of a node). Some of these (logical) TE nodes may reside in the client layer TE topology while the remaining TE nodes belong to the server layer TE topology.

In Figure 1, Node-1, Node-2, and Node-3 are TE nodes.

3.3. TE Link

TE link is an element of a TE topology (presented as an edge on TE graph, arrows indicate one or both directions of the TE link). TE link represents one or several (physical) links or a fraction of a link. TE link belongs to and is fully defined in exactly one TE topology. TE link is assigned with the TE topology scope unique ID. TE link attributes include parameters related to the data plane aspects of the associated link(s) (e.g. unreserved bandwidth, resource maps/pools, etc.), as well as the configuration data (such as remote node/link IDs, SRLGs, administrative colors, etc.). TE link is connected to TE node, terminating the TE link via exactly one TE link termination point (LTP).

In Figure 1, Link-12 and Link-23 are TE links.

3.4. Transitional TE Link for Multi-Layer Topologies

Networks are typically composed of multiple network layers where one or multiple signals in the client layer network can be multiplexed and encapsulated into a server layer signal. The server layer signal can be carried in the server layer network across multiple nodes until the server layer signal is terminated and the client layer signals reappear in the node that terminates the server layer signal. Examples of multi-layer networks are: IP over MPLS over Ethernet, low order ODUk signals multiplexed into a high order ODUl (l>k) carried over an OCh signal (optical transport network).

TE links as defined in 3.3. can be used to represent links within a network layer. In case of a multi-layer network, TE nodes and TE links only allow representation of each network layer as a separate TE topology Each of these single layer TE topologies would be isolated from their client and their server layer TE topology, if present (the highest and the lowest network layer in the hierarchy only have a single adjacent layer below or above, respectively). Multiplexing of client layer signals and encapsulating them into a server layer signal requires a function that is provided inside a node (typically realized in hardware). This function is also called layer transition.

One of the key requirements for path computation is to be able to calculate a path between two endpoints across a multi-layer network based on the TE topology representing this multi-layer network. This means that an additional TE construct is needed that represents potential layer transitions in the multi-layer TE-topology that

connects the TE-topologies representing each separate network layer. The so-called transitional TE link is such a construct and it represents the layer transition function residing inside a node that is decomposed into multiple logical nodes that are represented as TE nodes. Hence, a transitional TE link connects a client layer node with a server layer node. A TE link as defined in 3.3. has LTPs of exactly the same kind on each link end whereas the transitional TE link has client layer LTPs on the client side of the transitional link and in most cases a single server layer LTP on the server side. It should be noted that transitional links are a helper construct in the multi-layer TE topology and they only exist as long as they are not in use (as they represent potential connectivity). When the server layer trail has been established between the server layer LTP of two transitional links in the server layer network, the resulting client layer link in the data plane will be represented as a normal TE link in the client layer topology. The transitional TE links will re-appear when the server layer trail has been torn down.

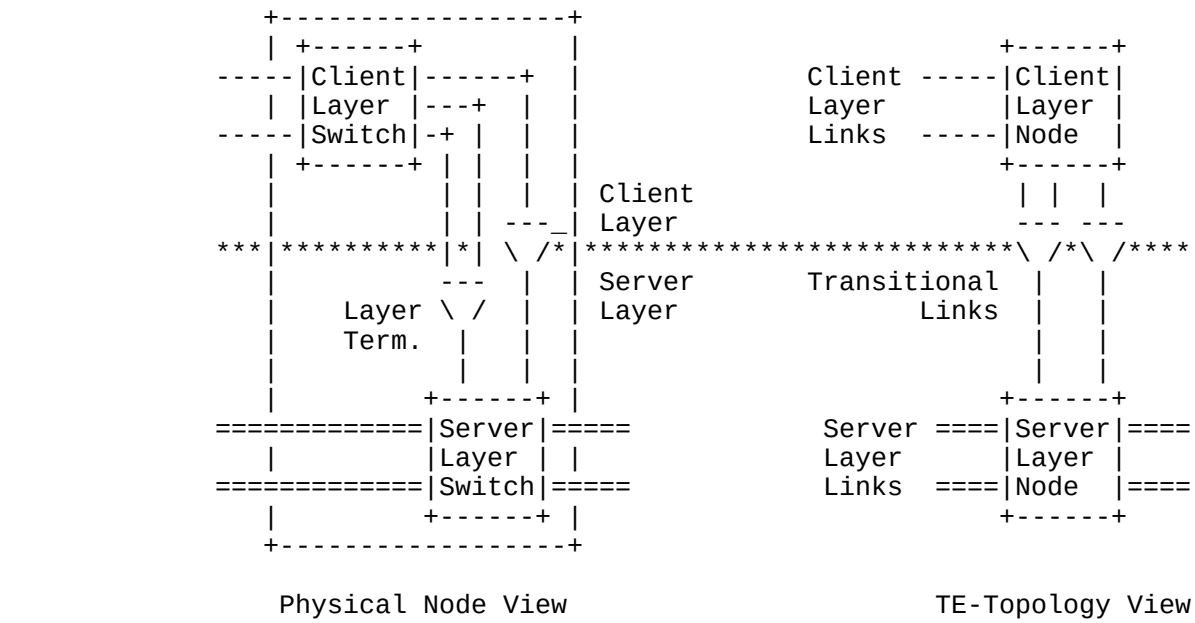


Figure 2: Modeling a Multi-Layer Node (Dual-Layer Example)

3.5. TE Link Termination Point (LTP)

TE link termination point (LTP) is a conceptual point of connection of a TE node to one of the TE links, terminated by the TE node. Cardinality between an LTP and the associated TE link is 1:0..1.

In Figure 1, Node-2 has six LTPs: LTP-1 to LTP-6.

3.6. TE Tunnel Termination Point (TTP)

TE tunnel termination point (TTP) is an element of TE topology representing one or several of potential transport service termination points (i.e. service client adaptation points such as WDM/OCh transponder). TTP is associated with (hosted by) exactly one TE node. TTP is assigned with the TE node scope unique ID. Depending on the TE node's internal constraints, a given TTP hosted by the TE node could be accessed via one, several or all TE links terminated by the TE node.

In Figure 1, Node-1 has two TTPs: TTP-1 and TTP-2.

3.7. TE Node Connectivity Matrix

TE node connectivity matrix is a TE node's attribute describing the TE node's switching limitations in a form of valid switching combinations of the TE node's LTPs (see below). From the point of view of a potential TE path arriving at the TE node at a given inbound LTP, the node's connectivity matrix describes valid (permissible) outbound LTPs for the TE path to leave the TE node from.

In Figure 1, the connectivity matrix on Node-2 is:
{<LTP-6, LTP-1>, <LTP-5, LTP-2>, <LTP-5, LTP-4>, <LTP-4, LTP-1>, <LTP-3, LTP-2>}

3.8. TTP Local Link Connectivity List (LLCL)

TTP Local Link Connectivity List (LLCL) is a List of TE links terminated by the TTP hosting TE node (i.e. list of the TE link LTPs), which the TTP could be connected to. From the point of view of a potential TE path LLCL provides a list of valid TE links the TE path needs to start/stop on for the connection, taking the TE path, to be successfully terminated on the TTP in question.

In Figure 1, the LLCL on Node-1 is:
{<TTP-1, LTP-5>, <TTP-1, LTP-2>, <TTP-2, LTP-3>, <TTP-2, LTP4>}

3.9. TE Path

TE path is an ordered list of TE links and/or TE nodes on the TE topology graph, inter-connecting a pair of TTPs to be taken by a potential connection. TE paths, for example, could be a product of successful path computation performed for a given transport service.

In Figure 1, the TE Path for TE-Tunnel-1 is:
{Node-1:TTP-1, Link-12, Node-2, Link-23, Node-3:TTP1}

3.10. TE Inter-Layer Lock

TE inter-layer lock is a modeling concept describing client-server layer adaptation relationships and hence important for the multi-layer traffic engineering. It is an association of M client layer LTPs and N server layer TTPs, within which data arriving at any of the client layer LTPs could be adopted onto any of the server layer TTPs. TE inter-layer lock is identified by inter-layer lock ID, which is unique across all TE topologies provided by the same provider. The client layer LIPs and the server layer TTPs associated within a given TE inter-layer lock are decorated with the same inter-layer lock ID attribute.

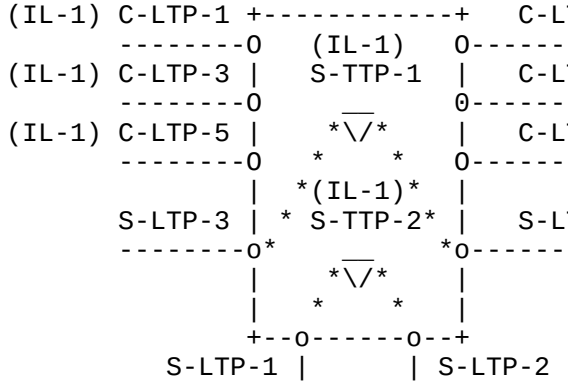


Figure 3: TE Inter-Layer Lock ID Associations

On the picture above a TE inter-layer lock with IL_1 ID associates 6 client layer LTPs (C-LTP-1 - C-LTP-6) with two server layer TTPs (S-TTP-1 and S-TTP-2). They all have the same attribute - TE inter-layer lock ID: IL-1, which is the only thing that indicates the association. A given LTP may have 0, 1 or more inter-layer lock IDs. In the latter case this means that the data arriving at the LTP may be adopted onto any of TTPs associated with all specified inter-layer locks. For example, C-LTP-1 could have two inter-layer lock IDs - IL-1 and IL-2. This would mean that C-LTP-1 for adaptation purposes could use not just TTPs associated with inter-layer lock IL-1 (i.e. S-TTP-1 and S-TTP-2 on the picture), but any of TTPs associated with inter-layer lock IL-2 as well. Likewise, a given TTP may have one or more inter-layer lock IDs, meaning that it can offer the adaptation service to any of client layer LTPs with inter-layer lock ID matching one of its own. Additionally, each TTP has an attribute - Unreserved

Adaptation Bandwidth, which announces its remaining adaptation resources sharable between all potential client LTPs.

LTPs and TTPs associated within the same TE inter-layer lock may be hosted by the same (hybrid, multi-layer) TE node or multiple TE nodes located in the same or separate TE topologies. The latter is especially important since TE topologies of different layer networks could be modeled by separate augmentations of the basic (common to all layers) TE topology model.

3.11. Underlay TE topology

Underlay TE topology is a TE topology that serves as a base for constructing of overlay TE topologies

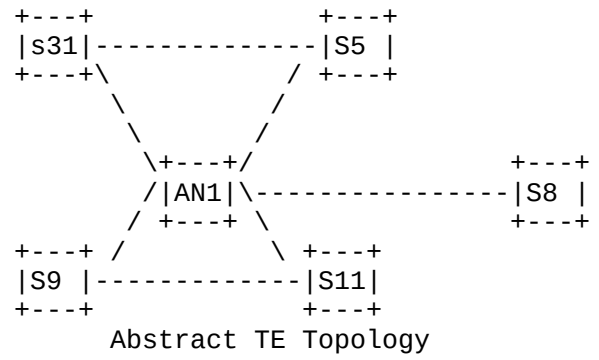
3.12. Overlay TE topology

Overlay TE topology is a TE topology constructed based on one or more underlay TE topologies. Each TE node of the overlay TE topology represents an arbitrary segment of an underlay TE topology; each TE link of the overlay TE topology represents an arbitrary TE path in one of the underlay TE topologies. The overlay TE topology and the supporting underlay TE topologies may represent distinct layer networks (e.g. OTN/ODUk and WDM/OCh respectively) or the same layer network.

3.13. Abstract TE topology

Abstract TE topology is a topology that contains abstract topological elements (nodes, links, tunnel termination points). Abstract TE topology is an overlay TE topology created by a topology provider and customized for a topology provider’s client based on one or more of the provider’s native TE topologies (underlay TE topologies), the provider’s policies and the client’s preferences. For example, a first level topology provider (such as Domain Controller) can create an abstract TE topology for its client (e.g. Multi-Domain Service Coordinator) based on the provider’s one or more native TE topologies, local policies/profiles and the client’s TE topology configuration requests

Figure 4 shows an example of abstract TE topology.



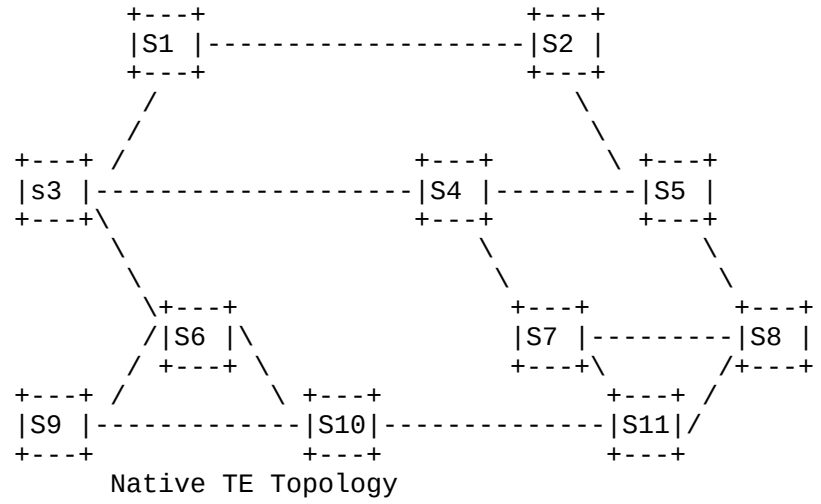


Figure 4: Abstract TE Topology

4. Model Applicability

4.1. Native TE Topologies

The model discussed in this draft can be used to represent and retrieve native TE topologies on a given TE system.

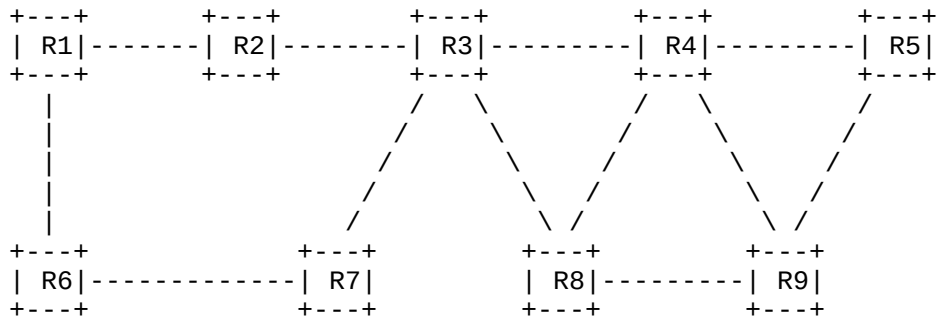


Figure 5a: Example Network Topology

Consider the network topology depicted in Figure 5a (R1 .. R9 are nodes representing routers). An implementation MAY choose to construct a native TE Topology using all nodes and links present in the given TED as depicted in Figure 5b. The data model proposed in this document can be used to retrieve/represent this TE topology.

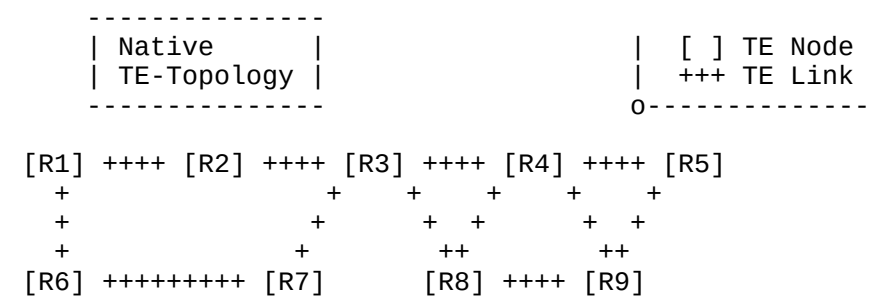


Figure 5b: Native TE Topology as seen on Node R3

Consider the case of the topology being split in a way that some nodes participate in OSPF-TE while others participate in ISIS-TE (Figure 6a). An implementation MAY choose to construct separate TE Topologies based on the information source. The native TE Topologies constructed using only nodes and links that were learnt via a specific information source are depicted in Figure 6b. The data model proposed in this document can be used to retrieve/represent these TE topologies.

Similarly, the data model can be used to represent/retrieve a TE Topology that is constructed using only nodes and links that belong to a particular technology layer. The data model is flexible enough to retrieve and represent many such native TE Topologies.

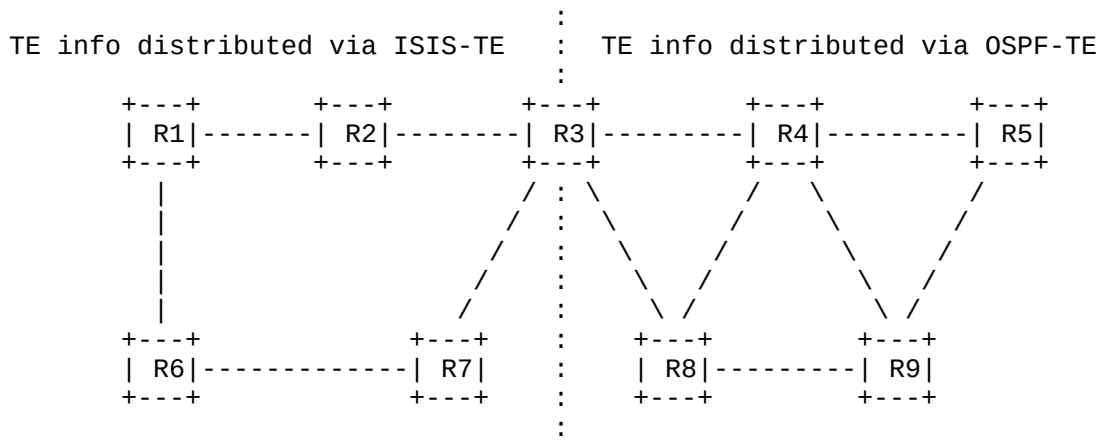


Figure 6a: Example Network Topology

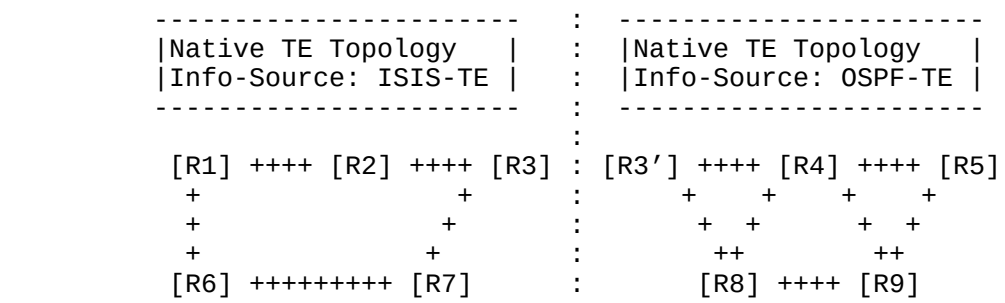


Figure 6b: Native TE Topologies as seen on Node R3

4.2. Customized TE Topologies

Customized TE topology is a topology that was modified by the provider to honor a particular client’s requirements or preferences. The model discussed in this draft can be used to represent, retrieve and manipulate customized TE Topologies. The model allows the provider to present the network in abstract TE Terms on a per client basis. These customized topologies contain sufficient information for the path computing client to select paths according to its policies.

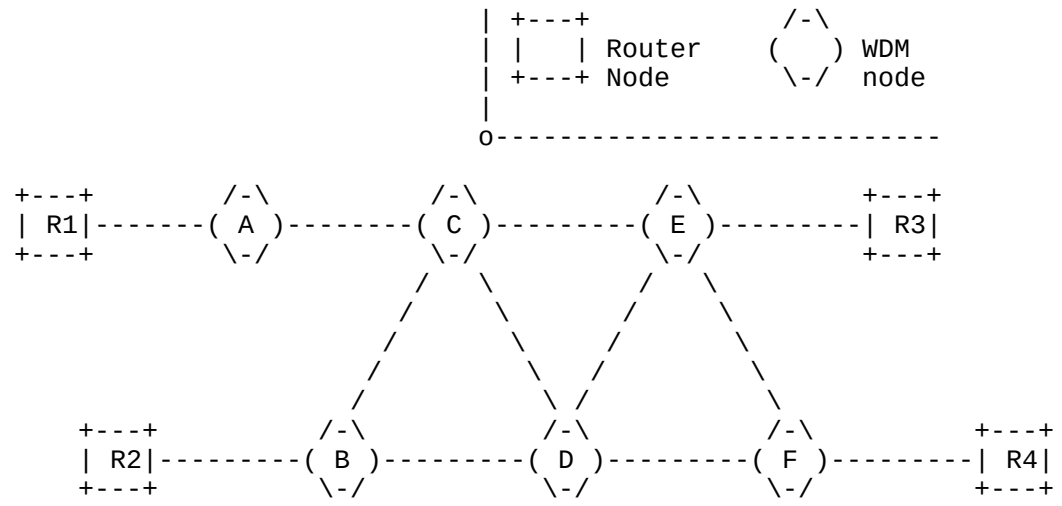


Figure 7: Example packet optical topology

Consider the network topology depicted in Figure 7. This is a typical packet optical transport deployment scenario where the WDM layer network domain serves as a Server Network Domain providing transport connectivity to the packet layer network Domain (Client Network Domain). Nodes R1, R2, R3 and R4 are IP routers that are connected to an Optical WDM transport network. A, B, C, D, E and F are WDM nodes that constitute the Server Network Domain.

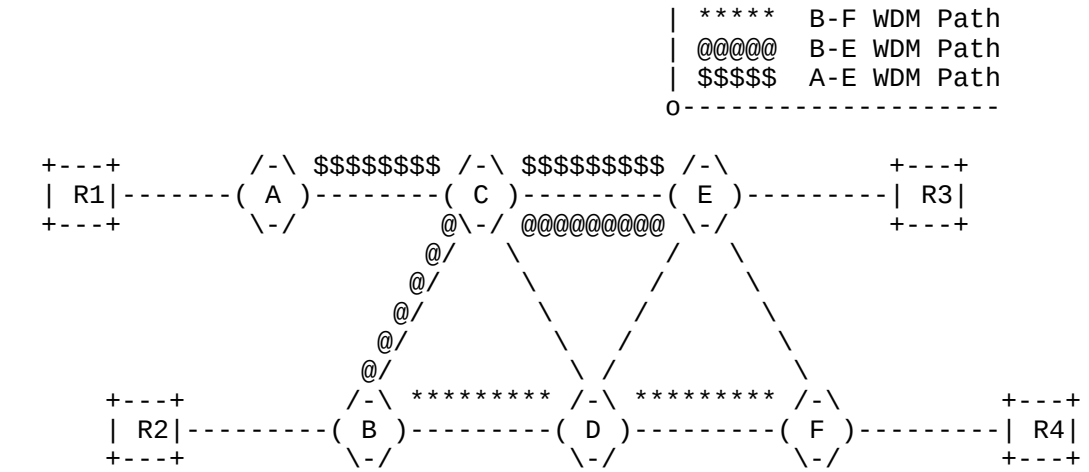


Figure 8a: Paths within the provider domain

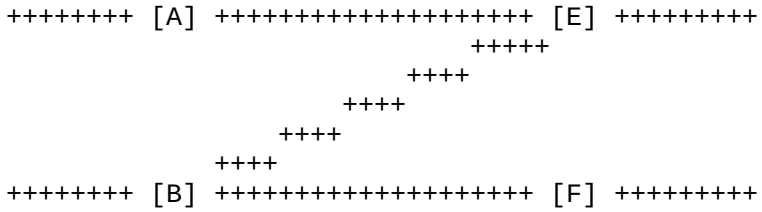


Figure 8b: Customized TE Topology provided to the Client

The goal here is to augment the Client TE Topology with a customized TE Topology provided by the WDM network. Given the availability of the paths A-E, B-F and B-E (Figure 8a), a customized TE Topology as depicted in Figure 8b is provided to the Client. This customized TE Topology is merged with the Client's Native TE Topology and the resulting topology is depicted in Figure 8c.

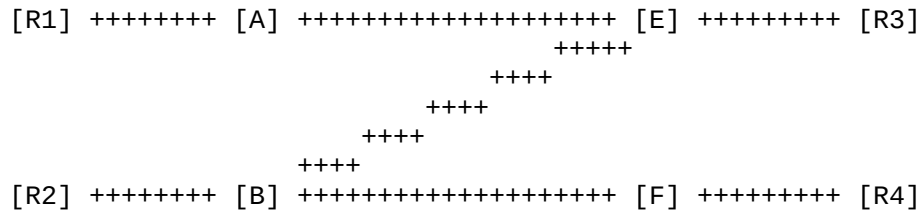


Figure 8c: Customized TE Topology merged with the Client's Native TE Topology

The data model proposed in this document can be used to retrieve/represent/manipulate the customized TE Topology depicted in Figure 8b.

A customized TE topology is not necessarily an abstract TE topology. he provider may produce, for example, an abstract TE topology of certain type (e.g. single-abstract-node-with-connectivit-matrix topology, a border_nodes_connected_via_mesh_of_abstract_links topology, etc.) and expose it to all/some clients in expectation that the clients will use it without customization.
On the other hand, a client may request a customized version of the provider's native TE topology (e.g. by requesting removal of TE links which belong to certain layers, are too slow, not protected and/or have a certain affinity). Note that the resulting TE topology will

not be abstract (because it will not contain abstract elements), but customized (modified upon client's instructions).

The client ID field in the TE topology identifier (Section 5.4. indicates which client the TE topology is customized for. Although a authorized client MAY receive a TE topology with the client ID field matching some other client, the client can customize only TE topologies with the client ID field either 0 or matching the ID of the client in question. If the client starts reconfiguration of a topology its client ID will be automatically set in the topology ID field for all future configurations and updates wrt. the topology in question.

The provider MAY tell the client that a given TE topology cannot be re-negotiated, by setting its own (provider's) ID in the client ID field of the topology ID.

4.3. Merging TE Topologies Provided by Multiple Providers

A client may receive TE topologies provided by multiple providers, each of which managing a separate domain of multi-domain network. In order to make use of said topologies, the client is expected to merge the provided TE topologies into one or more client's native TE topologies, each of which homogeneously representing the multi-domain network. This makes it possible for the client to select end-to-end TE paths for its services traversing multiple domains.

In particular, the process of merging TE topologies includes:

- Identifying neighboring domains and locking their topologies horizontally by connecting their inter-domain open-ended TE links;
- Renaming TE node, link, and SRLG IDs to ones allocated from a separate name space; this is necessary because all TE topologies are considered to be, generally speaking, independent with a possibility of clashes among TE node, link or SRLG IDs;
- Locking, vertically, TE topologies associated with different layer networks, according to provided topology inter-layer locks; this is to facilitate inter-layer path computations across multiple TE topologies provided by the same topology provider.

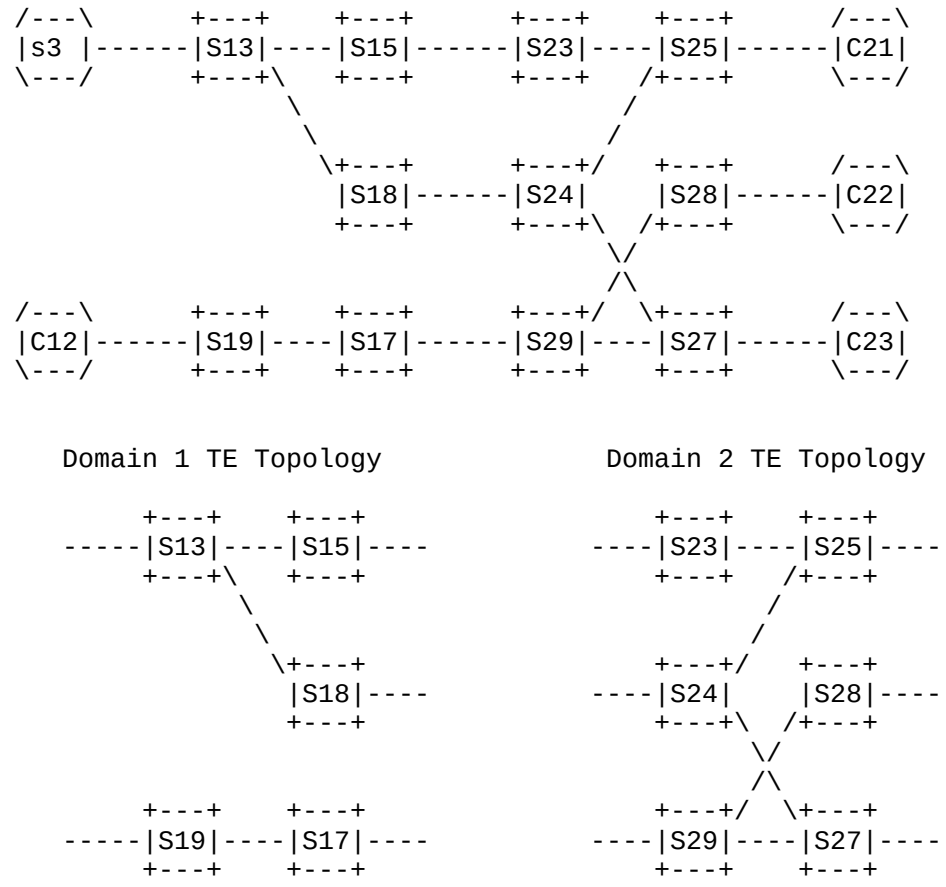


Figure 9: Merging Domain TE Topologies

Figure 9 illustrates the process of merging, by the client, of TE topologies provided by the client's providers. In the Figure, each of the two providers caters to the client (abstract or native) TE topology, describing the network domain under the respective provider's control. The client, by consulting the attributes of the inter-domain TE links - such as inter-domain plug IDs or remote TE node/link IDs (as defined by the TE Topology model) - is able to determine that:

a) the two domains are adjacent and are inter-connected via three inter-domain TE links, and;

b) each domain is connected to a separate customer site, connecting the left domain in the Figure to customer devices C-11 and C-12, and the right domain to customer devices C-21, C-22 and C-23.

Therefore, the client inter-connects the open-ended TE links, as shown on the upper part of the Figure.

As mentioned, one way to inter-connect the open-ended inter-domain TE links of neighboring domains is to mandate the providers to specify remote nodeID/linkID attribute in the provided inter-domain TE links. This, however, may prove to be not flexible. For example, the providers may not know the respective remote nodeIDs/ linkIDs. More importantly, this option does not allow for the client to mix-n-match multiple (more than one) topologies catered by the same providers (see below). Another, more flexible, option to resolve the open-ended inter-domain TE links is by decorating them with the inter-domain plug ID attribute. Inter-domain plug ID is a network-wide unique number that identifies on the network a connectivity supporting a given inter-domain TE link. Instead of specifying remote node ID/link ID, an inter-domain TE link may provide a non-zero inert-domain plug ID. It is expected that two neighboring domain TE topologies (provided by separate providers) will have each at least one open-ended inter-domain TE link with an inter-domain plug ID matching to one provided by its neighbor. For example, the inter-domain TE link originating from node S5 of the Domain 1 TE topology (Figure 1) and the inter-domain TE link coming from node S3 of Domain2 TE topology may specify matching inter-domain plug ID (e.g. 175344) This allows for the client to identify adjacent nodes in the separate neighboring TE topologies and resolve the inter-domain TE links connecting them regardless of their respective nodeIDs/linkIDs (which, as mentioned, could be allocated from independent name spaces). Inter-domain plug IDs may be assigned and managed by a central network authority. Alternatively, inter-domain plug IDs could be dynamically auto-discovered (e.g. via LMP protocol).

Furthermore, the client renames the TE nodes, links and SRLGs offered in the abstract TE topologies by assigning to them IDs allocated from a separate name space managed by the client. Such renaming is necessary, because the two abstract TE topologies may have their own name spaces, generally speaking, independent one from another; hence, ID overlaps/clashes are possible. For example, both TE topologies have TE nodes named S7, which, after renaming, appear in the merged TE topology as S17 and S27, respectively.

Once the merging process is complete, the client can use the merged TE topology for path computations across both domains, for example, to compute a TE path connecting C-11 to C-23.

4.4. Dealing with Multiple Abstract TE Topologies Provided by the Same Provider

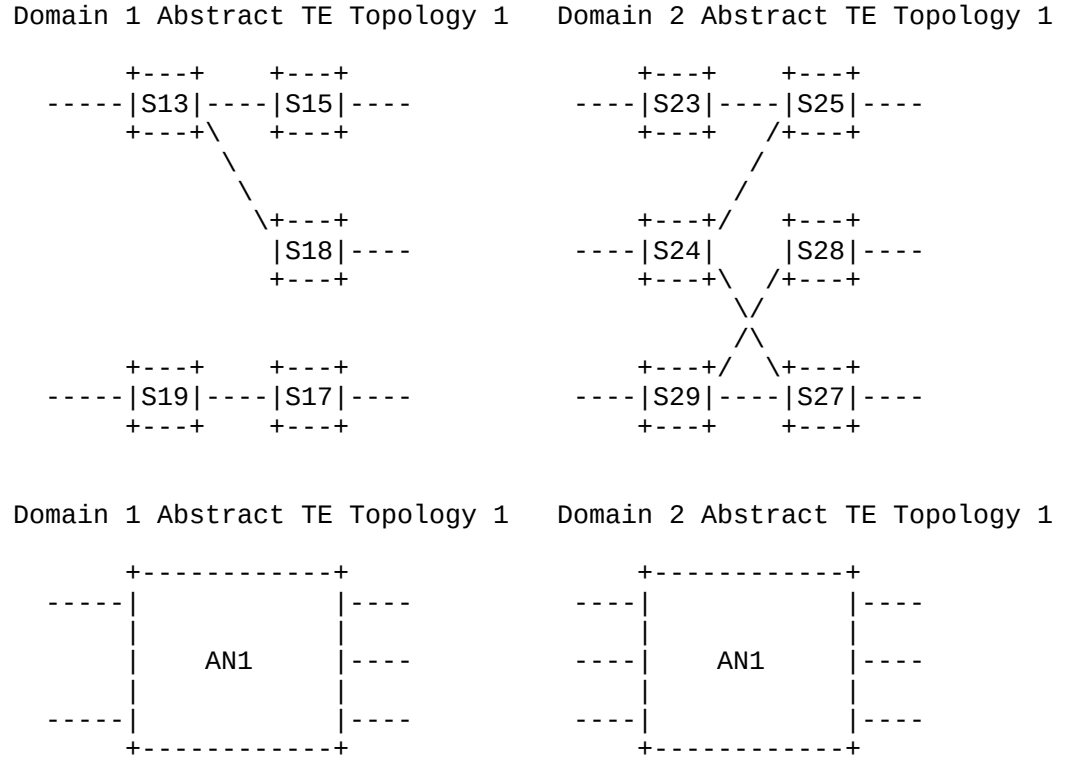


Figure 10: Merging Domain TE Topologies

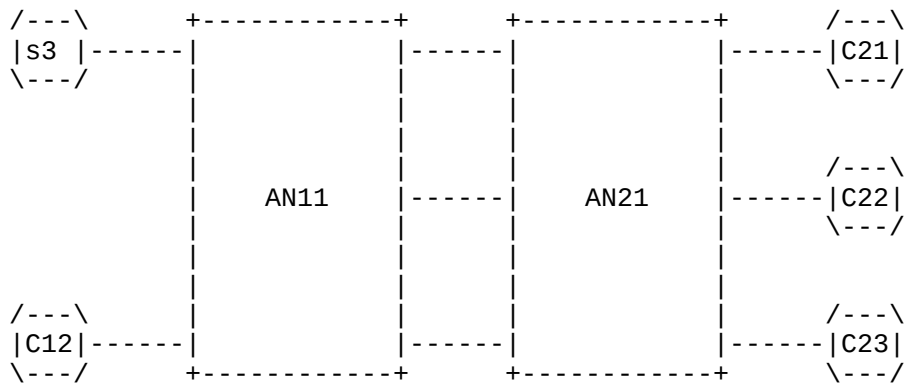
Based on local configuration, templates and/or policies pushed by the client, a given provider may expose more than one abstract TE topology to the client. For example, one abstract TE topology could be optimized based on a lowest-cost criterion, while another one could be based on best possible delay metrics, while yet another one could be based on maximum bandwidth availability for the client services. Furthermore, the client may request all or some providers to expose additional abstract TE topologies, possibly of a different type and/or optimized differently, as compared to already-provided TE topologies. In any case, the client should be prepared for a provider to offer to the client more than one abstract TE topology.

It should be up to the client (based on the client's local configuration and/or policies conveyed to the client by the client's

clients) to decide how to mix-and-match multiple abstract TE topologies provided by each or some of the providers, as well as how to merge them into the client's native TE topologies. The client also decides how many such merged TE topologies it needs to produce and maintain. For example, in addition to the merged TE topology depicted in the upper part of Figure 1, the client may merge the abstract TE topologies received from the two providers, as shown in Figure 10, into the client's additional native TE topologies, as shown in Figure 11.

Note that allowing for the client mix-n-matching of multiple TE topologies assumes that inter-domain plug IDs (rather than remote nodeID/linkID) option is used for identifying neighboring domains and inter-domain TE link resolution.

Client's Merged TE Topology 2



Client's Merged TE Topology 3

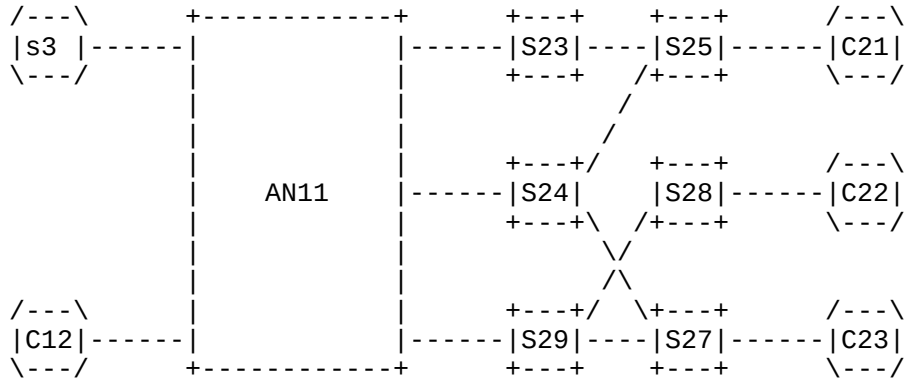


Figure 11: Multiple Native (Merged) Client's TE Topologies

It is important to note that each of the three native (merged) TE topologies could be used by the client for computing TE paths for any of the multi-domain services. The choice as to which topology to use for a given service depends on the service parameters/requirements and the topology's style, optimization criteria and the level of details.

5. Modeling Considerations

5.1. Network topology building blocks

The network topology building blocks are discussed in [YANG-NET-TOPO]. The TE Topology model proposed in this document augments and uses the ietf-network-topology module defined in [YANG-NET-TOPO].

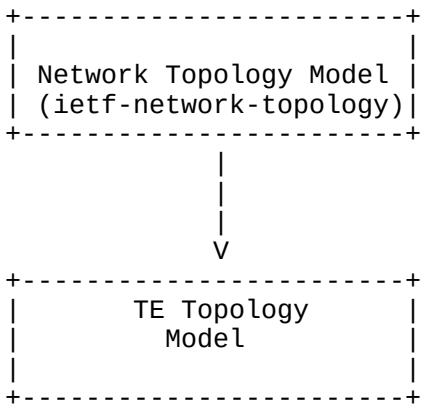


Figure 12: Augmenting the Network Topology Model

5.2. Technology agnostic TE Topology model

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

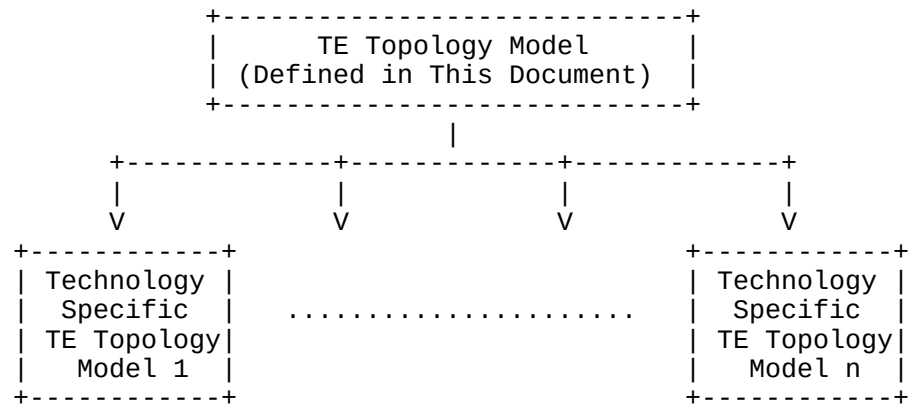


Figure 13: Augmenting the Technology agnostic TE Topology model

5.3. Model Structure

The high-level model structure proposed by this document is as shown below:

```
module: ietf-te-topology
augment /nw:networks/nw:network/nw:network-types:
   +--rw te-topology!

augment /nw:networks:
   +--rw te!
      +--rw templates
         +--rw node-template* [name] {template}?
         |  .............
         +--rw link-template* [name] {template}?
            .............

augment /nw:networks/nw:network:
   +--rw provider-id?      te-types:te-global-id
   +--rw client-id?        te-types:te-global-id
   +--rw te-topology-id?   te-types:te-topology-id
   +--rw te!
      +--rw config
      |  .............
      +--ro state
         .............

augment /nw:networks/nw:network/nw:node:
   +--rw te-node-id?   te-types:te-node-id
   +--rw te!
```

```
         +--rw config
         |  ............
         +--ro state
         |  ............
         +--rw tunnel-termination-point* [tunnel-tp-id]
            +--rw tunnel-tp-id    binary
            +--rw config
            |  ............
            +--ro state

   augment /nw:networks/nw:network/nt:link:
      +--rw te!
         +--rw config
         |  ..........
         +--ro state
            ..........

   augment /nw:networks/nw:network/nw:node/nt:termination-point:
      +--rw te-tp-id?   te-types:te-tp-id
      +--rw te!
         +--rw config
         |  ............
         +--ro state
            ............
```

5.4. Topology Identifiers

The TE-Topology is uniquely identified by a key that has 3 constituents - te-topology-id, provider-id and client-id. The combination of provider-id and te-topology-id uniquely identifies a native TE Topology on a given provider. The client-id is used only when Customized TE Topologies come into play; a value of "0" is used as the client-id for native TE Topologies.

```
   augment /nw:networks/nw:network:
      +--rw te!
         +--rw provider-id       te-global-id
         +--rw client-id         te-global-id
         +--rw te-topology-id    te-topology-id
```

5.5. Generic TE Link Attributes

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

```
     +--rw te-link-attributes
```

```
        ......................
        +--rw admin-status?                     te-admin-status
        |  ......................
        +--rw link-index?                       uint64
        +--rw administrative-group?             te-types:admin-groups
        +--rw link-protection-type?             enumeration
        +--rw max-link-bandwidth?               te-bandwidth
        +--rw max-resv-link-bandwidth?          te-bandwidth
        +--rw unreserved-bandwidth* [priority]
        |  ......................
        +--rw te-default-metric?                uint32
        |  ......................
        +--rw te-srlgs
        +--rw te-nsrlgs {nsrlg}?        ......................
```

5.6. Generic TE Node Attributes

The model covers the definitions for generic TE Node attributes.

The definition of a generic connectivity matrix is shown below:

```
  +--rw te-node-attributes
     ............
     +--rw connectivity-matrices
     ............
     |  +--rw connectivity-matrix* [id]
     |  |  +--rw id            uint32
     |  |  +--rw from
     |  |  |  +--rw tp-ref?        leafref
     |  |  +--rw to
     |  |  |  +--rw tp-ref?        leafref
     |  |  +--rw is-allowed?   boolean
     |  |  +--rw label-restriction* [inclusive-exclusive label-start]
     ............
     |  |  +--rw underlay! {te-topology-hierarchy}?
     ............
     |  |  +--rw max-link-bandwidth?        te-bandwidth
     |  |  +--rw max-resv-link-bandwidth?   te-bandwidth
     |  |  +--rw unreserved-bandwidth* [priority]
     ............
     |  |  +--rw te-default-metric?         uint32
     |  |  +--rw te-delay-metric?           uint32
     |  |  +--rw te-srlgs
     |  |  +--rw te-nsrlgs {nsrlg}?     ............The definition of
  a TTP Local Link Connectivity List is shown below:
```

```
     +--rw tunnel-termination-point* [tunnel-tp-id]
        +--rw tunnel-tp-id    binary
        +--rw config
        |  +--rw switching-capability?      identityref
        |  +--rw encoding?                  identityref
        |  +--rw inter-layer-lock-id?       uint32
        |  +--rw protection-type?           identityref
        |  +--rw local-link-connectivities
        ...........
        |     +--rw local-link-connectivity* [link-tp-ref]
        |        +--rw link-tp-ref                leafref
        |        +--rw label-restriction* [inclusive-exclusive label-
   start]
        ...........
        |        +--rw max-lsp-bandwidth* [priority]
        ...........
        |        +--rw max-link-bandwidth?        te-bandwidth
        |        +--rw max-resv-link-bandwidth?   te-bandwidth
        |        +--rw unreserved-bandwidth* [priority]
        ...........
        |        +--rw te-default-metric?         uint32
        |        +--rw te-delay-metric?           uint32
        |        +--rw te-srlgs
        |        +--rw te-nsrlgs {nsrlg}?
        ...........
        +--ro state
        |  +--ro switching-capability?      identityref
        |  +--ro encoding?                  identityref
        |  +--ro inter-layer-lock-id?       uint32
        |  +--ro protection-type?           identityref
        |  +--ro local-link-connectivities
        ...........
        |     +--ro local-link-connectivity* [link-tp-ref]
        |     |  +--ro link-tp-ref                leafref
        |     |  +--ro label-restriction* [inclusive-exclusive label-
   start]
        ...........
        |     |  +--ro max-lsp-bandwidth* [priority]
        ...........
        |     |  +--ro max-link-bandwidth?        te-bandwidth
        |     |  +--ro max-resv-link-bandwidth?   te-bandwidth
        |     |  +--ro unreserved-bandwidth* [priority]
        ...........
        |     |  +--ro te-default-metric?         uint32
        |     |  +--ro te-delay-metric?           uint32
        |     |  +--ro te-srlgs
        |     |  +--ro te-nsrlgs {nsrlg}?
```

```
        ...........
        +--rw supporting-tunnel-termination-point* [node-ref tunnel-tp-
   ref]
           +--rw node-ref         union
           +--rw tunnel-tp-ref    union
```

The attributes directly under container connectivity-matrices are the default attributes for all connectivity-matrix entries when the per entry corresponding attribute is not specified. When a per entry attribute is specified, it overrides the cooresponding attribute directly under the container connectivity-matrices. The same rule applies to the attributes directly under container local-link-connectivities.

Each TTP (Tunnel Termination Point) MAY be supported by one or more supporting TTPs. If the TE node hosting the TTP in question refers to a supporting TE node, then the supporting TTPs are hosted by the supporting TE node. If the TE node refers to an underlay TE topology, the supporting TTPs are hosted by one or more specified TE nodes of the underlay TE topology.

5.7. TED Information Sources

The model allows each TE topological element to have multiple TE information sources (OSPF-TE, ISIS-TE, BGP-LS, User-Configured, System-Processed, Other). Each information source is associated with a credibility preference to indicate precedence. In scenarios where a customized TE Topology is merged into a Client’s native TE Topology, the merged topological elements would point to the corresponding customized TE Topology as its information source.

```
augment /nw:networks/nw:network/nw:node:
   +--rw te!
      ...........
      +--ro state
         ........
      |  +--ro information-source?         te-info-source
      |  +--ro information-source-state
      |  |  +--ro credibility-preference?    uint16
      |  |  +--ro logical-network-element?   string
      |  |  +--ro network-instance?          string
      |  |  +--ro topology
      |  |     +--ro network-ref?       leafref
      |  |     +--ro node-ref?          leafref
      |  +--ro information-source-entry* [information-source]
            ............
```

```
augment /nw:networks/nw:network/nt:link:
   +--rw te!
      ...........
      +--ro state
         .........
      |  +--ro information-source?         te-info-source
      |  +--ro information-source-state
      |  |  +--ro credibility-preference?    uint16
      |  |  +--ro logical-network-element?   string
      |  |  +--ro network-instance?          string
      |  |  +--ro topology
      |  |     +--ro network-ref?       leafref
      |  |     +--ro link-ref?          leafref
      |  +--ro information-source-entry* [information-source]
            ............
```

5.8. Overlay/Underlay Relationship

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.

This relationship is captured via the "underlay-topology" field for the node and via the "underlay" field for the link. The use of these fields is optional and this functionality is tagged as a "feature" ("te-topology-hierarchy").

```
augment /nw:networks/nw:network/nw:node:
   +--rw te!
      +--rw te-node-id    te-node-id
      +--rw config
      |  +--rw te-node-template*     leafref {template}?
      |  +--rw te-node-attributes
      |     .....................
      |     +--rw underlay-topology {te-topology-hierarchy}?
      |        +--rw network-ref?       leafref

augment /nw:networks/nw:network/nt:link:
   +--rw te!
      +--rw config
      |  .........
      |  +--rw te-link-attributes
      |     .....................
```

```
         |     +--rw underlay! {te-topology-hierarchy}?
         |     |  +--rw primary-path
         |     |  |  +--rw network-ref?       leafref
         |     |  |  +--rw path-element* [path-element-id]
         |     |  |     ...............
         |     |  +--rw backup-path* [index]
         |     |  |  +--rw index              uint32
         |     |  |  +--rw network-ref?       leafref
         |     |  |  +--rw path-element* [path-element-id]
         |     |  |     ...............
         |     |  +--rw underlay-protection-type?   uint16
         |     |  +--rw underlay-tunnel-src
         |     |  |  ...........

         |     |  +--rw underlay-tunnel-des
         |     |     ...........
```

5.9. Templates

The data model provides the users with the ability to define templates and apply them to link and node configurations. The use of "template" configuration is optional and this functionality is tagged as a "feature" ("template").

```
  +--rw topology* [provider-id client-id te-topology-id]
  |  ...........
  |  +--rw node* [te-node-id]
  |  |  +--rw te-node-template?     leafref {template}?
  |  |     ..........
  |  +--rw link* [source-te-node-id source-te-link-id dest-te-node-id
dest-te-link-id]
  |     +--rw te-link-template?     leafref {template}?
  |        ..........

augment /nw:networks:
   +--rw te!
      +--rw templates
         +--rw node-template* [name] {template}?
         |  +--rw name                        te-types:te-template-
name
         |  +--rw priority?                   uint16
         |  +--rw reference-change-policy?    enumeration
         |  +--rw te-node-attributes
            ..........
         +--rw link-template* [name] {template}?
```

```
            +--rw name                          te-types:te-template-
   name
            +--rw priority?                     uint16
            +--rw reference-change-policy?      enumeration
            +--rw te-link-attributes
            ..........
```

Multiple templates can be specified to a configuration element. When two or more templates specify values for the same configuration field, the value from the template with the highest priority is used. The reference-change-policy specifies the action that needs to be taken when the template changes on a configuration element that has a reference to this template. The choices of action include taking no action, rejecting the change to the template and applying the change to the corresponding configuration. [Editor’s Note: The notion of "templates" has wider applicability. It is possible for this to be discussed in a separate document.]

5.10. Scheduling Parameters

The model allows time scheduling parameters to be specified for each topological element or for the topology as a whole. These parameters allow the provider to present different topological views to the client at different time slots. The use of "scheduling parameters" is optional.

The YANG data model for configuration scheduling is defined in [YANG-SCHEDULE], which allows specifying configuration schedules without altering this data model.

5.11. Notifications

Notifications are a key component of any topology data model.

[YANG-PUSH] and [RFC5277bis] define a subscription and push mechanism for YANG datastores. This mechanism currently allows the user to:

- Subscribe notifications on a per client basis
- Specify subtree filters or xpath filters so that only interested contents will be sent.
- Specify either periodic or on-demand notifications.

6. Tree Structure

```
module: ietf-te-topology
augment /nw:networks/nw:network/nw:network-types:
   +--rw te-topology!
```

```
   augment /nw:networks:
      +--rw te!
         +--rw templates
            +--rw node-template* [name] {template}?
            |  +--rw name                        te-types:te-template-
   name
            |  +--rw priority?                   uint16
            |  +--rw reference-change-policy?    enumeration
            |  +--rw te-node-attributes
            |     +--rw admin-status?         te-types:te-admin-status
            |     +--rw domain-id?            uint32
            |     +--rw is-abstract?          empty
            |     +--rw name?                 inet:domain-name
            |     +--rw signaling-address*    inet:ip-address
            |     +--rw underlay-topology {te-topology-hierarchy}?
            |        +--rw network-ref?   leafref
            +--rw link-template* [name] {template}?
               +--rw name                        te-types:te-template-
   name
               +--rw priority?                   uint16
               +--rw reference-change-policy?    enumeration
               +--rw te-link-attributes
                  +--rw access-type?                      te-types:te-
   link-access-type
                  +--rw external-domain
                  |  +--rw network-ref?            leafref
                  |  +--rw remote-te-node-id?      te-types:te-node-id
                  |  +--rw remote-te-link-tp-id?   te-types:te-tp-id
                  |  +--rw plug-id?                uint32
                  +--rw is-abstract?                      empty
                  +--rw name?                             string
                  +--rw underlay {te-topology-hierarchy}?
                  |  +--rw enabled?                       boolean
                  |  +--rw primary-path
                  |  |  +--rw network-ref?    leafref
                  |  |  +--rw path-element* [path-element-id]
                  |  |     +--rw path-element-id    uint32
                  |  |     +--rw index?             uint32
                  |  |     +--rw (type)?
                  |  |        +--:(ip-address)
                  |  |        |  +--rw ip-address-hop
                  |  |        |     +--rw address?    inet:ip-address
                  |  |        |     +--rw hop-type?   te-hop-type
                  |  |        +--:(as-number)
                  |  |        |  +--rw as-number-hop
                  |  |        |     +--rw as-number?   binary
                  |  |        |     +--rw hop-type?    te-hop-type
```

```
                |  |        +--:(unnumbered-link)
                |  |        |  +--rw unnumbered-hop
                |  |        |     +--rw router-id?      inet:ip-
   address
                |  |        |     +--rw interface-id?   uint32
                |  |        |     +--rw hop-type?       te-hop-type
                |  |        +--:(label)
                |  |        |  +--rw label-hop
                |  |        |     +--rw value?   rt-types:generalized-
   label
                |  |        +--:(sid)
                |  |           +--rw sid-hop
                |  |              +--rw sid?   rt-types:generalized-
   label
                |  +--rw backup-path* [index]
                |  |  +--rw index           uint32
                |  |  +--rw network-ref?    leafref
                |  |  +--rw path-element* [path-element-id]
                |  |     +--rw path-element-id    uint32
                |  |     +--rw index?             uint32
                |  |     +--rw (type)?
                |  |        +--:(ip-address)
                |  |        |  +--rw ip-address-hop
                |  |        |     +--rw address?    inet:ip-address
                |  |        |     +--rw hop-type?   te-hop-type
                |  |        +--:(as-number)
                |  |        |  +--rw as-number-hop
                |  |        |     +--rw as-number?   binary
                |  |        |     +--rw hop-type?    te-hop-type
                |  |        +--:(unnumbered-link)
                |  |        |  +--rw unnumbered-hop
                |  |        |     +--rw router-id?      inet:ip-
   address
                |  |        |     +--rw interface-id?   uint32
                |  |        |     +--rw hop-type?       te-hop-type
                |  |        +--:(label)
                |  |        |  +--rw label-hop
                |  |        |     +--rw value?   rt-types:generalized-
   label
                |  |        +--:(sid)
                |  |           +--rw sid-hop
                |  |              +--rw sid?   rt-types:generalized-
   label
                |  +--rw protection-type?             identityref
                |  +--rw tunnel-termination-points
                |  |  +--rw source?        binary
                |  |  +--rw destination?   binary
```

```
                |  +--rw tunnels
                |     +--rw sharing?   boolean
                |     +--rw tunnel* [tunnel-name]
                |        +--rw tunnel-name    string
                |        +--rw sharing?       boolean
                +--rw admin-status?                       te-types:te-
   admin-status
                +--rw link-index?                         uint64
                +--rw administrative-group?               te-
   types:admin-groups
                +--rw interface-switching-capability* [switching-
   capability encoding]
                |  +--rw switching-capability    identityref
                |  +--rw encoding                identityref
                |  +--rw max-lsp-bandwidth* [priority]
                |     +--rw priority     uint8
                |     +--rw bandwidth?   te-bandwidth
                +--rw link-protection-type?               enumeration
                +--rw max-link-bandwidth?                 te-bandwidth
                +--rw max-resv-link-bandwidth?            te-bandwidth
                +--rw unreserved-bandwidth* [priority]
                |  +--rw priority     uint8
                |  +--rw bandwidth?   te-bandwidth
                +--rw te-default-metric?                  uint32
                +--rw te-delay-metric?                    uint32
                +--rw te-srlgs
                |  +--rw value*   te-types:srlg
                +--rw te-nsrlgs {nsrlg}?
                   +--rw id*   uint32
   augment /nw:networks/nw:network:
      +--rw provider-id?      te-types:te-global-id
      +--rw client-id?        te-types:te-global-id
      +--rw te-topology-id?   te-types:te-topology-id
      +--rw te!
         +--rw config
         |  +--rw preference?                uint8
         |  +--rw optimization-criterion?   identityref
         |  +--rw nsrlg* [id] {nsrlg}?
         |     +--rw id              uint32
         |     +--rw disjointness?   te-path-disjointness
         +--ro state
            +--ro preference?                uint8
            +--ro optimization-criterion?   identityref
            +--ro nsrlg* [id] {nsrlg}?
            |  +--ro id              uint32
            |  +--ro disjointness?   te-path-disjointness
            +--ro geolocation
```

```
            +--ro altitude?    int64
            +--ro latitude?    geographic-coordinate-degree
            +--ro longitude?   geographic-coordinate-degree
   augment /nw:networks/nw:network/nw:node:
      +--rw te-node-id?   te-types:te-node-id
      +--rw te!
         +--rw config
         |  +--rw te-node-template*     leafref {template}?
         |  +--rw te-node-attributes
         |     +--rw admin-status?            te-types:te-admin-status
         |     +--rw connectivity-matrices
         |     |  +--rw number-of-entries?         uint16
         |     |  +--rw label-restriction* [inclusive-exclusive label-
   start]
         |     |  |  +--rw inclusive-exclusive    enumeration
         |     |  |  +--rw label-start            rt-types:generalized-
   label
         |     |  |  +--rw label-end?             rt-types:generalized-
   label
         |     |  |  +--rw range-bitmap?          binary
         |     |  +--rw is-allowed?                boolean
         |     |  +--rw underlay {te-topology-hierarchy}?
         |     |  |  +--rw enabled?                    boolean
         |     |  |  +--rw primary-path
         |     |  |  |  +--rw network-ref?    leafref
         |     |  |  |  +--rw path-element* [path-element-id]
         |     |  |  |     +--rw path-element-id    uint32
         |     |  |  |     +--rw index?             uint32
         |     |  |  |     +--rw (type)?
         |     |  |  |        +--:(ip-address)
         |     |  |  |        |  +--rw ip-address-hop
         |     |  |  |        |     +--rw address?    inet:ip-address
         |     |  |  |        |     +--rw hop-type?   te-hop-type
         |     |  |  |        +--:(as-number)
         |     |  |  |        |  +--rw as-number-hop
         |     |  |  |        |     +--rw as-number?   binary
         |     |  |  |        |     +--rw hop-type?    te-hop-type
         |     |  |  |        +--:(unnumbered-link)
         |     |  |  |        |  +--rw unnumbered-hop
         |     |  |  |        |     +--rw router-id?      inet:ip-
   address
         |     |  |  |        |     +--rw interface-id?   uint32
         |     |  |  |        |     +--rw hop-type?       te-hop-type
         |     |  |  |        +--:(label)
         |     |  |  |        |  +--rw label-hop
         |     |  |  |        |     +--rw value?   rt-types:generalized-
   label
```

```
          |      |  |  |        +--:(sid)
          |      |  |  |           +--rw sid-hop
          |      |  |  |              +--rw sid?   rt-types:generalized-
   label
          |      |  |  +--rw backup-path* [index]
          |      |  |  |  +--rw index           uint32
          |      |  |  |  +--rw network-ref?    leafref
          |      |  |  |  +--rw path-element* [path-element-id]
          |      |  |  |     +--rw path-element-id    uint32
          |      |  |  |     +--rw index?             uint32
          |      |  |  |     +--rw (type)?
          |      |  |  |        +--:(ip-address)
          |      |  |  |        |  +--rw ip-address-hop
          |      |  |  |        |     +--rw address?    inet:ip-address
          |      |  |  |        |     +--rw hop-type?   te-hop-type
          |      |  |  |        +--:(as-number)
          |      |  |  |        |  +--rw as-number-hop
          |      |  |  |        |     +--rw as-number?   binary
          |      |  |  |        |     +--rw hop-type?    te-hop-type
          |      |  |  |        +--:(unnumbered-link)
          |      |  |  |        |  +--rw unnumbered-hop
          |      |  |  |        |     +--rw router-id?      inet:ip-
   address
          |      |  |  |        |     +--rw interface-id?   uint32
          |      |  |  |        |     +--rw hop-type?       te-hop-type
          |      |  |  |        +--:(label)
          |      |  |  |        |  +--rw label-hop
          |      |  |  |        |     +--rw value?   rt-types:generalized-
   label
          |      |  |  |        +--:(sid)
          |      |  |  |           +--rw sid-hop
          |      |  |  |              +--rw sid?   rt-types:generalized-
   label
          |      |  |  +--rw protection-type?             identityref
          |      |  |  +--rw tunnel-termination-points
          |      |  |  |  +--rw source?        binary
          |      |  |  |  +--rw destination?   binary
          |      |  |  +--rw tunnels
          |      |  |     +--rw sharing?   boolean
          |      |  |     +--rw tunnel* [tunnel-name]
          |      |  |        +--rw tunnel-name    string
          |      |  |        +--rw sharing?       boolean
          |      |  +--rw max-lsp-bandwidth* [priority]
          |      |  |  +--rw priority     uint8
          |      |  |  +--rw bandwidth?   te-bandwidth
          |      |  +--rw max-link-bandwidth?        te-bandwidth
          |      |  +--rw max-resv-link-bandwidth?   te-bandwidth
```

```
          |      |  +--rw unreserved-bandwidth* [priority]
          |      |  |  +--rw priority     uint8
          |      |  |  +--rw bandwidth?   te-bandwidth
          |      |  +--rw te-default-metric?         uint32
          |      |  +--rw te-delay-metric?           uint32
          |      |  +--rw te-srlgs
          |      |  |  +--rw value*   te-types:srlg
          |      |  +--rw te-nsrlgs {nsrlg}?
          |      |  |  +--rw id*   uint32
          |      |  +--rw connectivity-matrix* [id]
          |      |     +--rw id                          uint32
          |      |     +--rw from
          |      |     |  +--rw tp-ref?              leafref
          |      |     |  +--rw label-restriction* [inclusive-exclusive
    label-start]
          |      |     |     +--rw inclusive-exclusive    enumeration
          |      |     |     +--rw label-start            rt-
    types:generalized-label
          |      |     |     +--rw label-end?             rt-
    types:generalized-label
          |      |     |     +--rw range-bitmap?          binary
          |      |     +--rw to
          |      |     |  +--rw tp-ref?              leafref
          |      |     |  +--rw label-restriction* [inclusive-exclusive
    label-start]
          |      |     |     +--rw inclusive-exclusive    enumeration
          |      |     |     +--rw label-start            rt-
    types:generalized-label
          |      |     |     +--rw label-end?             rt-
    types:generalized-label
          |      |     |     +--rw range-bitmap?          binary
          |      |     +--rw is-allowed?                boolean
          |      |     +--rw underlay {te-topology-hierarchy}?
          |      |     |  +--rw enabled?                     boolean
          |      |     |  +--rw primary-path
          |      |     |  |  +--rw network-ref?    leafref
          |      |     |  |  +--rw path-element* [path-element-id]
          |      |     |  |     +--rw path-element-id    uint32
          |      |     |  |     +--rw index?             uint32
          |      |     |  |     +--rw (type)?
          |      |     |  |        +--:(ip-address)
          |      |     |  |        |  +--rw ip-address-hop
          |      |     |  |        |     +--rw address?    inet:ip-address
          |      |     |  |        |     +--rw hop-type?   te-hop-type
          |      |     |  |        +--:(as-number)
          |      |     |  |        |  +--rw as-number-hop
          |      |     |  |        |     +--rw as-number?   binary
```

```
          |     |     |  |        |     +--rw hop-type?    te-hop-type
          |     |     |  |        +--:(unnumbered-link)
          |     |     |  |        |  +--rw unnumbered-hop
          |     |     |  |        |     +--rw router-id?      inet:ip-
   address
          |     |     |  |        |     +--rw interface-id?   uint32
          |     |     |  |        |     +--rw hop-type?       te-hop-type
          |     |     |  |        +--:(label)
          |     |     |  |        |  +--rw label-hop
          |     |     |  |        |     +--rw value?   rt-
   types:generalized-label
          |     |     |  |        +--:(sid)
          |     |     |  |           +--rw sid-hop
          |     |     |  |              +--rw sid?   rt-
   types:generalized-label
          |     |     |  +--rw backup-path* [index]
          |     |     |  |  +--rw index           uint32
          |     |     |  |  +--rw network-ref?    leafref
          |     |     |  |  +--rw path-element* [path-element-id]
          |     |     |  |     +--rw path-element-id    uint32
          |     |     |  |     +--rw index?             uint32
          |     |     |  |     +--rw (type)?
          |     |     |  |        +--:(ip-address)
          |     |     |  |        |  +--rw ip-address-hop
          |     |     |  |        |     +--rw address?    inet:ip-address
          |     |     |  |        |     +--rw hop-type?   te-hop-type
          |     |     |  |        +--:(as-number)
          |     |     |  |        |  +--rw as-number-hop
          |     |     |  |        |     +--rw as-number?   binary
          |     |     |  |        |     +--rw hop-type?    te-hop-type
          |     |     |  |        +--:(unnumbered-link)
          |     |     |  |        |  +--rw unnumbered-hop
          |     |     |  |        |     +--rw router-id?      inet:ip-
   address
          |     |     |  |        |     +--rw interface-id?   uint32
          |     |     |  |        |     +--rw hop-type?       te-hop-type
          |     |     |  |        +--:(label)
          |     |     |  |        |  +--rw label-hop
          |     |     |  |        |     +--rw value?   rt-
   types:generalized-label
          |     |     |  |        +--:(sid)
          |     |     |  |           +--rw sid-hop
          |     |     |  |              +--rw sid?   rt-
   types:generalized-label
          |     |     |  +--rw protection-type?             identityref
          |     |     |  +--rw tunnel-termination-points
          |     |     |  |  +--rw source?        binary
```

```
          |     |     |  |  +--rw destination?   binary
          |     |     |  +--rw tunnels
          |     |     |     +--rw sharing?   boolean
          |     |     |     +--rw tunnel* [tunnel-name]
          |     |     |        +--rw tunnel-name    string
          |     |     |        +--rw sharing?       boolean
          |     |     +--rw max-lsp-bandwidth* [priority]
          |     |     |  +--rw priority     uint8
          |     |     |  +--rw bandwidth?   te-bandwidth
          |     |     +--rw max-link-bandwidth?        te-bandwidth
          |     |     +--rw max-resv-link-bandwidth?   te-bandwidth
          |     |     +--rw unreserved-bandwidth* [priority]
          |     |     |  +--rw priority     uint8
          |     |     |  +--rw bandwidth?   te-bandwidth
          |     |     +--rw te-default-metric?         uint32
          |     |     +--rw te-delay-metric?           uint32
          |     |     +--rw te-srlgs
          |     |     |  +--rw value*   te-types:srlg
          |     |     +--rw te-nsrlgs {nsrlg}?
          |     |        +--rw id*   uint32
          |     +--rw domain-id?              uint32
          |     +--rw is-abstract?            empty
          |     +--rw name?                   inet:domain-name
          |     +--rw signaling-address*      inet:ip-address
          |     +--rw underlay-topology {te-topology-hierarchy}?
          |        +--rw network-ref?   leafref
          +--ro state
          |  +--ro te-node-template*          leafref {template}?
          |  +--ro te-node-attributes
          |  |  +--ro admin-status?           te-types:te-admin-status
          |  |  +--ro connectivity-matrices
          |  |  |  +--ro number-of-entries?         uint16
          |  |  |  +--ro label-restriction* [inclusive-exclusive label-
   start]
          |  |  |  |  +--ro inclusive-exclusive    enumeration
          |  |  |  |  +--ro label-start            rt-types:generalized-
   label
          |  |  |  |  +--ro label-end?             rt-types:generalized-
   label
          |  |  |  |  +--ro range-bitmap?          binary
          |  |  |  +--ro is-allowed?                boolean
          |  |  |  +--ro underlay {te-topology-hierarchy}?
          |  |  |  |  +--ro enabled?                     boolean
          |  |  |  |  +--ro primary-path
          |  |  |  |  |  +--ro network-ref?    leafref
          |  |  |  |  |  +--ro path-element* [path-element-id]
          |  |  |  |  |     +--ro path-element-id    uint32
```

```
          |  |  |  |  |     +--ro index?            uint32
          |  |  |  |  |     +--ro (type)?
          |  |  |  |  |        +--:(ip-address)
          |  |  |  |  |        |  +--ro ip-address-hop
          |  |  |  |  |        |     +--ro address?    inet:ip-address
          |  |  |  |  |        |     +--ro hop-type?   te-hop-type
          |  |  |  |  |        +--:(as-number)
          |  |  |  |  |        |  +--ro as-number-hop
          |  |  |  |  |        |     +--ro as-number?   binary
          |  |  |  |  |        |     +--ro hop-type?    te-hop-type
          |  |  |  |  |        +--:(unnumbered-link)
          |  |  |  |  |        |  +--ro unnumbered-hop
          |  |  |  |  |        |     +--ro router-id?      inet:ip-
   address
          |  |  |  |  |        |     +--ro interface-id?   uint32
          |  |  |  |  |        |     +--ro hop-type?       te-hop-type
          |  |  |  |  |        +--:(label)
          |  |  |  |  |        |  +--ro label-hop
          |  |  |  |  |        |     +--ro value?   rt-types:generalized-
   label
          |  |  |  |  |        +--:(sid)
          |  |  |  |  |           +--ro sid-hop
          |  |  |  |  |              +--ro sid?   rt-types:generalized-
   label
          |  |  |  |  +--ro backup-path* [index]
          |  |  |  |  |  +--ro index           uint32
          |  |  |  |  |  +--ro network-ref?    leafref
          |  |  |  |  |  +--ro path-element* [path-element-id]
          |  |  |  |  |     +--ro path-element-id    uint32
          |  |  |  |  |     +--ro index?             uint32
          |  |  |  |  |     +--ro (type)?
          |  |  |  |  |        +--:(ip-address)
          |  |  |  |  |        |  +--ro ip-address-hop
          |  |  |  |  |        |     +--ro address?    inet:ip-address
          |  |  |  |  |        |     +--ro hop-type?   te-hop-type
          |  |  |  |  |        +--:(as-number)
          |  |  |  |  |        |  +--ro as-number-hop
          |  |  |  |  |        |     +--ro as-number?   binary
          |  |  |  |  |        |     +--ro hop-type?    te-hop-type
          |  |  |  |  |        +--:(unnumbered-link)
          |  |  |  |  |        |  +--ro unnumbered-hop
          |  |  |  |  |        |     +--ro router-id?      inet:ip-
   address
          |  |  |  |  |        |     +--ro interface-id?   uint32
          |  |  |  |  |        |     +--ro hop-type?       te-hop-type
          |  |  |  |  |        +--:(label)
          |  |  |  |  |        |  +--ro label-hop
```

```
      |  |  |  |  |        |     +--ro value?   rt-types:generalized-
   label
      |  |  |  |  |        +--:(sid)
      |  |  |  |  |           +--ro sid-hop
      |  |  |  |  |              +--ro sid?   rt-types:generalized-
   label
      |  |  |  |  +--ro protection-type?             identityref
      |  |  |  |  +--ro tunnel-termination-points
      |  |  |  |  |  +--ro source?        binary
      |  |  |  |  |  +--ro destination?   binary
      |  |  |  |  +--ro tunnels
      |  |  |  |     +--ro sharing?   boolean
      |  |  |  |     +--ro tunnel* [tunnel-name]
      |  |  |  |        +--ro tunnel-name    string
      |  |  |  |        +--ro sharing?       boolean
      |  |  |  +--ro max-lsp-bandwidth* [priority]
      |  |  |  |  +--ro priority     uint8
      |  |  |  |  +--ro bandwidth?   te-bandwidth
      |  |  |  +--ro max-link-bandwidth?        te-bandwidth
      |  |  |  +--ro max-resv-link-bandwidth?   te-bandwidth
      |  |  |  +--ro unreserved-bandwidth* [priority]
      |  |  |  |  +--ro priority     uint8
      |  |  |  |  +--ro bandwidth?   te-bandwidth
      |  |  |  +--ro te-default-metric?         uint32
      |  |  |  +--ro te-delay-metric?           uint32
      |  |  |  +--ro te-srlgs
      |  |  |  |  +--ro value*   te-types:srlg
      |  |  |  +--ro te-nsrlgs {nsrlg}?
      |  |  |  |  +--ro id*   uint32
      |  |  |  +--ro connectivity-matrix* [id]
      |  |  |     +--ro id                           uint32
      |  |  |     +--ro from
      |  |  |     |  +--ro tp-ref?              leafref
      |  |  |     |  +--ro label-restriction* [inclusive-exclusive
   label-start]
      |  |  |     |     +--ro inclusive-exclusive    enumeration
      |  |  |     |     +--ro label-start            rt-
   types:generalized-label
      |  |  |     |     +--ro label-end?             rt-
   types:generalized-label
      |  |  |     |     +--ro range-bitmap?          binary
      |  |  |     +--ro to
      |  |  |     |  +--ro tp-ref?              leafref
      |  |  |     |  +--ro label-restriction* [inclusive-exclusive
   label-start]
      |  |  |     |     +--ro inclusive-exclusive    enumeration
```

```
        |  |  |     |     +--ro label-start           rt-
   types:generalized-label
        |  |  |     |     +--ro label-end?            rt-
   types:generalized-label
        |  |  |     |     +--ro range-bitmap?         binary
        |  |  |     +--ro is-allowed?               boolean
        |  |  |     +--ro underlay {te-topology-hierarchy}?
        |  |  |     |  +--ro enabled?                    boolean
        |  |  |     |  +--ro primary-path
        |  |  |     |  |  +--ro network-ref?    leafref
        |  |  |     |  |  +--ro path-element* [path-element-id]
        |  |  |     |  |     +--ro path-element-id    uint32
        |  |  |     |  |     +--ro index?             uint32
        |  |  |     |  |     +--ro (type)?
        |  |  |     |  |        +--:(ip-address)
        |  |  |     |  |        |  +--ro ip-address-hop
        |  |  |     |  |        |     +--ro address?    inet:ip-address
        |  |  |     |  |        |     +--ro hop-type?   te-hop-type
        |  |  |     |  |        +--:(as-number)
        |  |  |     |  |        |  +--ro as-number-hop
        |  |  |     |  |        |     +--ro as-number?   binary
        |  |  |     |  |        |     +--ro hop-type?    te-hop-type
        |  |  |     |  |        +--:(unnumbered-link)
        |  |  |     |  |        |  +--ro unnumbered-hop
        |  |  |     |  |        |     +--ro router-id?      inet:ip-
   address
        |  |  |     |  |        |     +--ro interface-id?   uint32
        |  |  |     |  |        |     +--ro hop-type?       te-hop-type
        |  |  |     |  |        +--:(label)
        |  |  |     |  |        |  +--ro label-hop
        |  |  |     |  |        |     +--ro value?   rt-
   types:generalized-label
        |  |  |     |  |        +--:(sid)
        |  |  |     |  |           +--ro sid-hop
        |  |  |     |  |              +--ro sid?   rt-
   types:generalized-label
        |  |  |     |  +--ro backup-path* [index]
        |  |  |     |  |  +--ro index           uint32
        |  |  |     |  |  +--ro network-ref?    leafref
        |  |  |     |  |  +--ro path-element* [path-element-id]
        |  |  |     |  |     +--ro path-element-id    uint32
        |  |  |     |  |     +--ro index?             uint32
        |  |  |     |  |     +--ro (type)?
        |  |  |     |  |        +--:(ip-address)
        |  |  |     |  |        |  +--ro ip-address-hop
        |  |  |     |  |        |     +--ro address?    inet:ip-address
        |  |  |     |  |        |     +--ro hop-type?   te-hop-type
```

```
          |  |  |     |  |        +--:(as-number)
          |  |  |     |  |        |  +--ro as-number-hop
          |  |  |     |  |        |     +--ro as-number?   binary
          |  |  |     |  |        |     +--ro hop-type?    te-hop-type
          |  |  |     |  |        +--:(unnumbered-link)
          |  |  |     |  |        |  +--ro unnumbered-hop
          |  |  |     |  |        |     +--ro router-id?      inet:ip-
   address
          |  |  |     |  |        |     +--ro interface-id?   uint32
          |  |  |     |  |        |     +--ro hop-type?       te-hop-type
          |  |  |     |  |        +--:(label)
          |  |  |     |  |        |  +--ro label-hop
          |  |  |     |  |        |     +--ro value?   rt-
   types:generalized-label
          |  |  |     |  |        +--:(sid)
          |  |  |     |  |           +--ro sid-hop
          |  |  |     |  |              +--ro sid?   rt-
   types:generalized-label
          |  |  |     |  +--ro protection-type?             identityref
          |  |  |     |  +--ro tunnel-termination-points
          |  |  |     |  |  +--ro source?        binary
          |  |  |     |  |  +--ro destination?   binary
          |  |  |     |  +--ro tunnels
          |  |  |     |     +--ro sharing?   boolean
          |  |  |     |     +--ro tunnel* [tunnel-name]
          |  |  |     |        +--ro tunnel-name    string
          |  |  |     |        +--ro sharing?       boolean
          |  |  |     +--ro max-lsp-bandwidth* [priority]
          |  |  |     |  +--ro priority     uint8
          |  |  |     |  +--ro bandwidth?   te-bandwidth
          |  |  |     +--ro max-link-bandwidth?        te-bandwidth
          |  |  |     +--ro max-resv-link-bandwidth?   te-bandwidth
          |  |  |     +--ro unreserved-bandwidth* [priority]
          |  |  |     |  +--ro priority     uint8
          |  |  |     |  +--ro bandwidth?   te-bandwidth
          |  |  |     +--ro te-default-metric?         uint32
          |  |  |     +--ro te-delay-metric?           uint32
          |  |  |     +--ro te-srlgs
          |  |  |     |  +--ro value*   te-types:srlg
          |  |  |     +--ro te-nsrlgs {nsrlg}?
          |  |  |        +--ro id*   uint32
          |  |  +--ro domain-id?               uint32
          |  |  +--ro is-abstract?             empty
          |  |  +--ro name?                    inet:domain-name
          |  |  +--ro signaling-address*       inet:ip-address
          |  |  +--ro underlay-topology {te-topology-hierarchy}?
          |  |     +--ro network-ref?   leafref
```

```
         |  +--ro oper-status?               te-types:te-oper-status
         |  +--ro geolocation
         |  |  +--ro altitude?    int64
         |  |  +--ro latitude?    geographic-coordinate-degree
         |  |  +--ro longitude?   geographic-coordinate-degree
         |  +--ro is-multi-access-dr?         empty
         |  +--ro information-source?         te-info-source
         |  +--ro information-source-state
         |  |  +--ro credibility-preference?    uint16
         |  |  +--ro logical-network-element?   string
         |  |  +--ro network-instance?          string
         |  |  +--ro topology
         |  |     +--ro network-ref?   leafref
         |  |     +--ro node-ref?      leafref
         |  +--ro information-source-entry* [information-source]
         |     +--ro information-source          te-info-source
         |     +--ro information-source-state
         |     |  +--ro credibility-preference?    uint16
         |     |  +--ro logical-network-element?   string
         |     |  +--ro network-instance?          string
         |     |  +--ro topology
         |     |     +--ro network-ref?   leafref
         |     |     +--ro node-ref?      leafref
         |     +--ro connectivity-matrices
         |     |  +--ro number-of-entries?         uint16
         |     |  +--ro label-restriction* [inclusive-exclusive label-
   start]
         |     |  |  +--ro inclusive-exclusive    enumeration
         |     |  |  +--ro label-start            rt-types:generalized-
   label
         |     |  |  +--ro label-end?             rt-types:generalized-
   label
         |     |  |  +--ro range-bitmap?          binary
         |     |  +--ro is-allowed?                boolean
         |     |  +--ro underlay {te-topology-hierarchy}?
         |     |  |  +--ro enabled?                      boolean
         |     |  |  +--ro primary-path
         |     |  |  |  +--ro network-ref?    leafref
         |     |  |  |  +--ro path-element* [path-element-id]
         |     |  |  |     +--ro path-element-id    uint32
         |     |  |  |     +--ro index?             uint32
         |     |  |  |     +--ro (type)?
         |     |  |  |        +--:(ip-address)
         |     |  |  |        |  +--ro ip-address-hop
         |     |  |  |        |     +--ro address?    inet:ip-address
         |     |  |  |        |     +--ro hop-type?   te-hop-type
         |     |  |  |        +--:(as-number)
```

```
          |      |  |  |        |  +--ro as-number-hop
          |      |  |  |        |     +--ro as-number?   binary
          |      |  |  |        |     +--ro hop-type?    te-hop-type
          |      |  |  |        +--:(unnumbered-link)
          |      |  |  |        |  +--ro unnumbered-hop
          |      |  |  |        |     +--ro router-id?      inet:ip-
   address
          |      |  |  |        |     +--ro interface-id?   uint32
          |      |  |  |        |     +--ro hop-type?       te-hop-type
          |      |  |  |        +--:(label)
          |      |  |  |        |  +--ro label-hop
          |      |  |  |        |     +--ro value?   rt-types:generalized-
   label
          |      |  |  |        +--:(sid)
          |      |  |  |           +--ro sid-hop
          |      |  |  |              +--ro sid?   rt-types:generalized-
   label
          |      |  |  +--ro backup-path* [index]
          |      |  |  |  +--ro index           uint32
          |      |  |  |  +--ro network-ref?    leafref
          |      |  |  |  +--ro path-element* [path-element-id]
          |      |  |  |     +--ro path-element-id    uint32
          |      |  |  |     +--ro index?             uint32
          |      |  |  |     +--ro (type)?
          |      |  |  |        +--:(ip-address)
          |      |  |  |        |  +--ro ip-address-hop
          |      |  |  |        |     +--ro address?    inet:ip-address
          |      |  |  |        |     +--ro hop-type?   te-hop-type
          |      |  |  |        +--:(as-number)
          |      |  |  |        |  +--ro as-number-hop
          |      |  |  |        |     +--ro as-number?   binary
          |      |  |  |        |     +--ro hop-type?    te-hop-type
          |      |  |  |        +--:(unnumbered-link)
          |      |  |  |        |  +--ro unnumbered-hop
          |      |  |  |        |     +--ro router-id?      inet:ip-
   address
          |      |  |  |        |     +--ro interface-id?   uint32
          |      |  |  |        |     +--ro hop-type?       te-hop-type
          |      |  |  |        +--:(label)
          |      |  |  |        |  +--ro label-hop
          |      |  |  |        |     +--ro value?   rt-types:generalized-
   label
          |      |  |  |        +--:(sid)
          |      |  |  |           +--ro sid-hop
          |      |  |  |              +--ro sid?   rt-types:generalized-
   label
          |      |  |  +--ro protection-type?             identityref
```

```
          |     |  |  +--ro tunnel-termination-points
          |     |  |  |  +--ro source?        binary
          |     |  |  |  +--ro destination?   binary
          |     |  |  +--ro tunnels
          |     |  |     +--ro sharing?   boolean
          |     |  |     +--ro tunnel* [tunnel-name]
          |     |  |        +--ro tunnel-name    string
          |     |  |        +--ro sharing?       boolean
          |     |  +--ro max-lsp-bandwidth* [priority]
          |     |  |  +--ro priority     uint8
          |     |  |  +--ro bandwidth?   te-bandwidth
          |     |  +--ro max-link-bandwidth?        te-bandwidth
          |     |  +--ro max-resv-link-bandwidth?   te-bandwidth
          |     |  +--ro unreserved-bandwidth* [priority]
          |     |  |  +--ro priority     uint8
          |     |  |  +--ro bandwidth?   te-bandwidth
          |     |  +--ro te-default-metric?         uint32
          |     |  +--ro te-delay-metric?           uint32
          |     |  +--ro te-srlgs
          |     |  |  +--ro value*   te-types:srlg
          |     |  +--ro te-nsrlgs {nsrlg}?
          |     |  |  +--ro id*   uint32
          |     |  +--ro connectivity-matrix* [id]
          |     |     +--ro id                          uint32
          |     |     +--ro from
          |     |     |  +--ro tp-ref?              leafref
          |     |     |  +--ro label-restriction* [inclusive-exclusive
   label-start]
          |     |     |     +--ro inclusive-exclusive    enumeration
          |     |     |     +--ro label-start            rt-
   types:generalized-label
          |     |     |     +--ro label-end?             rt-
   types:generalized-label
          |     |     |     +--ro range-bitmap?          binary
          |     |     +--ro to
          |     |     |  +--ro tp-ref?              leafref
          |     |     |  +--ro label-restriction* [inclusive-exclusive
   label-start]
          |     |     |     +--ro inclusive-exclusive    enumeration
          |     |     |     +--ro label-start            rt-
   types:generalized-label
          |     |     |     +--ro label-end?             rt-
   types:generalized-label
          |     |     |     +--ro range-bitmap?          binary
          |     |     +--ro is-allowed?                boolean
          |     |     +--ro underlay {te-topology-hierarchy}?
          |     |     |  +--ro enabled?                    boolean
```

```
          |      |      |  +--ro primary-path
          |      |      |  |  +--ro network-ref?    leafref
          |      |      |  |  +--ro path-element* [path-element-id]
          |      |      |  |     +--ro path-element-id    uint32
          |      |      |  |     +--ro index?             uint32
          |      |      |  |     +--ro (type)?
          |      |      |  |        +--:(ip-address)
          |      |      |  |        |  +--ro ip-address-hop
          |      |      |  |        |     +--ro address?    inet:ip-address
          |      |      |  |        |     +--ro hop-type?   te-hop-type
          |      |      |  |        +--:(as-number)
          |      |      |  |        |  +--ro as-number-hop
          |      |      |  |        |     +--ro as-number?   binary
          |      |      |  |        |     +--ro hop-type?    te-hop-type
          |      |      |  |        +--:(unnumbered-link)
          |      |      |  |        |  +--ro unnumbered-hop
          |      |      |  |        |     +--ro router-id?      inet:ip-
   address
          |      |      |  |        |     +--ro interface-id?   uint32
          |      |      |  |        |     +--ro hop-type?       te-hop-type
          |      |      |  |        +--:(label)
          |      |      |  |        |  +--ro label-hop
          |      |      |  |        |     +--ro value?   rt-
   types:generalized-label
          |      |      |  |        +--:(sid)
          |      |      |  |           +--ro sid-hop
          |      |      |  |              +--ro sid?   rt-
   types:generalized-label
          |      |      |  +--ro backup-path* [index]
          |      |      |  |  +--ro index           uint32
          |      |      |  |  +--ro network-ref?    leafref
          |      |      |  |  +--ro path-element* [path-element-id]
          |      |      |  |     +--ro path-element-id    uint32
          |      |      |  |     +--ro index?             uint32
          |      |      |  |     +--ro (type)?
          |      |      |  |        +--:(ip-address)
          |      |      |  |        |  +--ro ip-address-hop
          |      |      |  |        |     +--ro address?    inet:ip-address
          |      |      |  |        |     +--ro hop-type?   te-hop-type
          |      |      |  |        +--:(as-number)
          |      |      |  |        |  +--ro as-number-hop
          |      |      |  |        |     +--ro as-number?   binary
          |      |      |  |        |     +--ro hop-type?    te-hop-type
          |      |      |  |        +--:(unnumbered-link)
          |      |      |  |        |  +--ro unnumbered-hop
          |      |      |  |        |     +--ro router-id?      inet:ip-
   address
```

```
          |     |     |  |        |     +--ro interface-id?   uint32
          |     |     |  |        |     +--ro hop-type?       te-hop-type
          |     |     |  |        +--:(label)
          |     |     |  |        |  +--ro label-hop
          |     |     |  |        |     +--ro value?   rt-
   types:generalized-label
          |     |     |  |        +--:(sid)
          |     |     |  |           +--ro sid-hop
          |     |     |  |              +--ro sid?   rt-
   types:generalized-label
          |     |     |  +--ro protection-type?             identityref
          |     |     |  +--ro tunnel-termination-points
          |     |     |  |  +--ro source?        binary
          |     |     |  |  +--ro destination?   binary
          |     |     |  +--ro tunnels
          |     |     |     +--ro sharing?   boolean
          |     |     |     +--ro tunnel* [tunnel-name]
          |     |     |        +--ro tunnel-name    string
          |     |     |        +--ro sharing?       boolean
          |     |     +--ro max-lsp-bandwidth* [priority]
          |     |     |  +--ro priority     uint8
          |     |     |  +--ro bandwidth?   te-bandwidth
          |     |     +--ro max-link-bandwidth?        te-bandwidth
          |     |     +--ro max-resv-link-bandwidth?   te-bandwidth
          |     |     +--ro unreserved-bandwidth* [priority]
          |     |     |  +--ro priority     uint8
          |     |     |  +--ro bandwidth?   te-bandwidth
          |     |     +--ro te-default-metric?         uint32
          |     |     +--ro te-delay-metric?           uint32
          |     |     +--ro te-srlgs
          |     |     |  +--ro value*   te-types:srlg
          |     |     +--ro te-nsrlgs {nsrlg}?
          |     |        +--ro id*   uint32
          |     +--ro domain-id?                  uint32
          |     +--ro is-abstract?                empty
          |     +--ro name?                       inet:domain-name
          |     +--ro signaling-address*          inet:ip-address
          |     +--ro underlay-topology {te-topology-hierarchy}?
          |        +--ro network-ref?   leafref
          +--ro statistics
          |  +--ro discontinuity-time          yang:date-and-time
          |  +--ro node
          |  |  +--ro disables?             yang:counter32
          |  |  +--ro enables?              yang:counter32
          |  |  +--ro maintenance-sets?     yang:counter32
          |  |  +--ro maintenance-clears?   yang:counter32
          |  |  +--ro modifies?             yang:counter32
```

```
         |  +--ro connectivity-matrix-entry
         |     +--ro creates?    yang:counter32
         |     +--ro deletes?    yang:counter32
         |     +--ro disables?   yang:counter32
         |     +--ro enables?    yang:counter32
         |     +--ro modifies?   yang:counter32
         +--rw tunnel-termination-point* [tunnel-tp-id]
            +--rw tunnel-tp-id                          binary
            +--rw config
            |  +--rw admin-status?               te-types:te-admin-
   status
            |  +--rw name?                       string
            |  +--rw switching-capability?       identityref
            |  +--rw encoding?                   identityref
            |  +--rw inter-layer-lock-id*        uint32
            |  +--rw protection-type?            identityref
            |  +--rw client-layer-adaptation
            |  |  +--rw switching-capability* [switching-capability
   encoding]
            |  |     +--rw switching-capability    identityref
            |  |     +--rw encoding                identityref
            |  |     +--rw bandwidth?              te-bandwidth
            |  +--rw local-link-connectivities
            |     +--rw number-of-entries?         uint16
            |     +--rw label-restriction* [inclusive-exclusive label-
   start]
            |     |  +--rw inclusive-exclusive    enumeration
            |     |  +--rw label-start            rt-types:generalized-
   label
            |     |  +--rw label-end?             rt-types:generalized-
   label
            |     |  +--rw range-bitmap?          binary
            |     +--rw is-allowed?                boolean
            |     +--rw underlay {te-topology-hierarchy}?
            |     |  +--rw enabled?                     boolean
            |     |  +--rw primary-path
            |     |  |  +--rw network-ref?    leafref
            |     |  |  +--rw path-element* [path-element-id]
            |     |  |     +--rw path-element-id    uint32
            |     |  |     +--rw index?             uint32
            |     |  |     +--rw (type)?
            |     |  |        +--:(ip-address)
            |     |  |        |  +--rw ip-address-hop
            |     |  |        |     +--rw address?    inet:ip-address
            |     |  |        |     +--rw hop-type?   te-hop-type
            |     |  |        +--:(as-number)
            |     |  |        |  +--rw as-number-hop
```

```
            |     |  |        |     +--rw as-number?   binary
            |     |  |        |     +--rw hop-type?    te-hop-type
            |     |  |        +--:(unnumbered-link)
            |     |  |        |  +--rw unnumbered-hop
            |     |  |        |     +--rw router-id?      inet:ip-
   address
            |     |  |        |     +--rw interface-id?   uint32
            |     |  |        |     +--rw hop-type?       te-hop-type
            |     |  |        +--:(label)
            |     |  |        |  +--rw label-hop
            |     |  |        |     +--rw value?   rt-types:generalized-
   label
            |     |  |        +--:(sid)
            |     |  |           +--rw sid-hop
            |     |  |              +--rw sid?   rt-types:generalized-
   label
            |     |  +--rw backup-path* [index]
            |     |  |  +--rw index           uint32
            |     |  |  +--rw network-ref?    leafref
            |     |  |  +--rw path-element* [path-element-id]
            |     |  |     +--rw path-element-id    uint32
            |     |  |     +--rw index?             uint32
            |     |  |     +--rw (type)?
            |     |  |        +--:(ip-address)
            |     |  |        |  +--rw ip-address-hop
            |     |  |        |     +--rw address?    inet:ip-address
            |     |  |        |     +--rw hop-type?   te-hop-type
            |     |  |        +--:(as-number)
            |     |  |        |  +--rw as-number-hop
            |     |  |        |     +--rw as-number?   binary
            |     |  |        |     +--rw hop-type?    te-hop-type
            |     |  |        +--:(unnumbered-link)
            |     |  |        |  +--rw unnumbered-hop
            |     |  |        |     +--rw router-id?      inet:ip-
   address
            |     |  |        |     +--rw interface-id?   uint32
            |     |  |        |     +--rw hop-type?       te-hop-type
            |     |  |        +--:(label)
            |     |  |        |  +--rw label-hop
            |     |  |        |     +--rw value?   rt-types:generalized-
   label
            |     |  |        +--:(sid)
            |     |  |           +--rw sid-hop
            |     |  |              +--rw sid?   rt-types:generalized-
   label
            |     |  +--rw protection-type?             identityref
            |     |  +--rw tunnel-termination-points
```

```
            |     |  |  +--rw source?        binary
            |     |  |  +--rw destination?   binary
            |     |  +--rw tunnels
            |     |     +--rw sharing?   boolean
            |     |     +--rw tunnel* [tunnel-name]
            |     |        +--rw tunnel-name    string
            |     |        +--rw sharing?       boolean
            |     +--rw max-lsp-bandwidth* [priority]
            |     |  +--rw priority     uint8
            |     |  +--rw bandwidth?   te-bandwidth
            |     +--rw max-link-bandwidth?        te-bandwidth
            |     +--rw max-resv-link-bandwidth?   te-bandwidth
            |     +--rw unreserved-bandwidth* [priority]
            |     |  +--rw priority     uint8
            |     |  +--rw bandwidth?   te-bandwidth
            |     +--rw te-default-metric?         uint32
            |     +--rw te-delay-metric?           uint32
            |     +--rw te-srlgs
            |     |  +--rw value*   te-types:srlg
            |     +--rw te-nsrlgs {nsrlg}?
            |     |  +--rw id*   uint32
            |     +--rw local-link-connectivity* [link-tp-ref]
            |        +--rw link-tp-ref                leafref
            |        +--rw label-restriction* [inclusive-exclusive
   label-start]
            |        |  +--rw inclusive-exclusive    enumeration
            |        |  +--rw label-start            rt-
   types:generalized-label
            |        |  +--rw label-end?             rt-
   types:generalized-label
            |        |  +--rw range-bitmap?          binary
            |        +--rw is-allowed?                boolean
            |        +--rw underlay {te-topology-hierarchy}?
            |        |  +--rw enabled?                      boolean
            |        |  +--rw primary-path
            |        |  |  +--rw network-ref?    leafref
            |        |  |  +--rw path-element* [path-element-id]
            |        |  |     +--rw path-element-id    uint32
            |        |  |     +--rw index?             uint32
            |        |  |     +--rw (type)?
            |        |  |        +--:(ip-address)
            |        |  |        |  +--rw ip-address-hop
            |        |  |        |     +--rw address?    inet:ip-address
            |        |  |        |     +--rw hop-type?   te-hop-type
            |        |  |        +--:(as-number)
            |        |  |        |  +--rw as-number-hop
            |        |  |        |     +--rw as-number?   binary
```

```
           |        |  |        |     +--rw hop-type?    te-hop-type
           |        |  |        +--:(unnumbered-link)
           |        |  |        |  +--rw unnumbered-hop
           |        |  |        |     +--rw router-id?      inet:ip-
   address
           |        |  |        |     +--rw interface-id?   uint32
           |        |  |        |     +--rw hop-type?       te-hop-type
           |        |  |        +--:(label)
           |        |  |        |  +--rw label-hop
           |        |  |        |     +--rw value?   rt-
   types:generalized-label
           |        |  |        +--:(sid)
           |        |  |           +--rw sid-hop
           |        |  |              +--rw sid?   rt-
   types:generalized-label
           |        |  +--rw backup-path* [index]
           |        |  |  +--rw index           uint32
           |        |  |  +--rw network-ref?    leafref
           |        |  |  +--rw path-element* [path-element-id]
           |        |  |     +--rw path-element-id    uint32
           |        |  |     +--rw index?             uint32
           |        |  |     +--rw (type)?
           |        |  |        +--:(ip-address)
           |        |  |        |  +--rw ip-address-hop
           |        |  |        |     +--rw address?    inet:ip-address
           |        |  |        |     +--rw hop-type?   te-hop-type
           |        |  |        +--:(as-number)
           |        |  |        |  +--rw as-number-hop
           |        |  |        |     +--rw as-number?   binary
           |        |  |        |     +--rw hop-type?    te-hop-type
           |        |  |        +--:(unnumbered-link)
           |        |  |        |  +--rw unnumbered-hop
           |        |  |        |     +--rw router-id?      inet:ip-
   address
           |        |  |        |     +--rw interface-id?   uint32
           |        |  |        |     +--rw hop-type?       te-hop-type
           |        |  |        +--:(label)
           |        |  |        |  +--rw label-hop
           |        |  |        |     +--rw value?   rt-
   types:generalized-label
           |        |  |        +--:(sid)
           |        |  |           +--rw sid-hop
           |        |  |              +--rw sid?   rt-
   types:generalized-label
           |        |  +--rw protection-type?             identityref
           |        |  +--rw tunnel-termination-points
           |        |  |  +--rw source?        binary
```

```
            |        |  |  +--rw destination?   binary
            |        |  +--rw tunnels
            |        |     +--rw sharing?   boolean
            |        |     +--rw tunnel* [tunnel-name]
            |        |        +--rw tunnel-name    string
            |        |        +--rw sharing?       boolean
            |        +--rw max-lsp-bandwidth* [priority]
            |        |  +--rw priority     uint8
            |        |  +--rw bandwidth?   te-bandwidth
            |        +--rw max-link-bandwidth?        te-bandwidth
            |        +--rw max-resv-link-bandwidth?   te-bandwidth
            |        +--rw unreserved-bandwidth* [priority]
            |        |  +--rw priority     uint8
            |        |  +--rw bandwidth?   te-bandwidth
            |        +--rw te-default-metric?         uint32
            |        +--rw te-delay-metric?           uint32
            |        +--rw te-srlgs
            |        |  +--rw value*   te-types:srlg
            |        +--rw te-nsrlgs {nsrlg}?
            |           +--rw id*   uint32
            +--ro state
            |  +--ro admin-status?                te-types:te-admin-
   status
            |  +--ro name?                        string
            |  +--ro switching-capability?        identityref
            |  +--ro encoding?                    identityref
            |  +--ro inter-layer-lock-id*         uint32
            |  +--ro protection-type?             identityref
            |  +--ro client-layer-adaptation
            |  |  +--ro switching-capability* [switching-capability
   encoding]
            |  |     +--ro switching-capability    identityref
            |  |     +--ro encoding                identityref
            |  |     +--ro bandwidth?              te-bandwidth
            |  +--ro local-link-connectivities
            |  |  +--ro number-of-entries?         uint16
            |  |  +--ro label-restriction* [inclusive-exclusive label-
   start]
            |  |  |  +--ro inclusive-exclusive    enumeration
            |  |  |  +--ro label-start            rt-types:generalized-
   label
            |  |  |  +--ro label-end?             rt-types:generalized-
   label
            |  |  |  +--ro range-bitmap?          binary
            |  |  +--ro is-allowed?                boolean
            |  |  +--ro underlay {te-topology-hierarchy}?
            |  |  |  +--ro enabled?                      boolean
```

```
            |  |  |  +--ro primary-path
            |  |  |  |  +--ro network-ref?    leafref
            |  |  |  |  +--ro path-element* [path-element-id]
            |  |  |  |     +--ro path-element-id    uint32
            |  |  |  |     +--ro index?             uint32
            |  |  |  |     +--ro (type)?
            |  |  |  |        +--:(ip-address)
            |  |  |  |        |  +--ro ip-address-hop
            |  |  |  |        |     +--ro address?    inet:ip-address
            |  |  |  |        |     +--ro hop-type?   te-hop-type
            |  |  |  |        +--:(as-number)
            |  |  |  |        |  +--ro as-number-hop
            |  |  |  |        |     +--ro as-number?   binary
            |  |  |  |        |     +--ro hop-type?    te-hop-type
            |  |  |  |        +--:(unnumbered-link)
            |  |  |  |        |  +--ro unnumbered-hop
            |  |  |  |        |     +--ro router-id?      inet:ip-
   address
            |  |  |  |        |     +--ro interface-id?   uint32
            |  |  |  |        |     +--ro hop-type?       te-hop-type
            |  |  |  |        +--:(label)
            |  |  |  |        |  +--ro label-hop
            |  |  |  |        |     +--ro value?   rt-types:generalized-
   label
            |  |  |  |        +--:(sid)
            |  |  |  |           +--ro sid-hop
            |  |  |  |              +--ro sid?   rt-types:generalized-
   label
            |  |  |  +--ro backup-path* [index]
            |  |  |  |  +--ro index           uint32
            |  |  |  |  +--ro network-ref?    leafref
            |  |  |  |  +--ro path-element* [path-element-id]
            |  |  |  |     +--ro path-element-id    uint32
            |  |  |  |     +--ro index?             uint32
            |  |  |  |     +--ro (type)?
            |  |  |  |        +--:(ip-address)
            |  |  |  |        |  +--ro ip-address-hop
            |  |  |  |        |     +--ro address?    inet:ip-address
            |  |  |  |        |     +--ro hop-type?   te-hop-type
            |  |  |  |        +--:(as-number)
            |  |  |  |        |  +--ro as-number-hop
            |  |  |  |        |     +--ro as-number?   binary
            |  |  |  |        |     +--ro hop-type?    te-hop-type
            |  |  |  |        +--:(unnumbered-link)
            |  |  |  |        |  +--ro unnumbered-hop
            |  |  |  |        |     +--ro router-id?      inet:ip-
   address
```

```
          |  |  |  |        |     +--ro interface-id?   uint32
          |  |  |  |        |     +--ro hop-type?       te-hop-type
          |  |  |  |        +--:(label)
          |  |  |  |        |  +--ro label-hop
          |  |  |  |        |     +--ro value?   rt-types:generalized-
   label
          |  |  |  |        +--:(sid)
          |  |  |  |           +--ro sid-hop
          |  |  |  |              +--ro sid?   rt-types:generalized-
   label
          |  |  |  +--ro protection-type?             identityref
          |  |  |  +--ro tunnel-termination-points
          |  |  |  |  +--ro source?        binary
          |  |  |  |  +--ro destination?   binary
          |  |  |  +--ro tunnels
          |  |  |     +--ro sharing?   boolean
          |  |  |     +--ro tunnel* [tunnel-name]
          |  |  |        +--ro tunnel-name    string
          |  |  |        +--ro sharing?       boolean
          |  |  +--ro max-lsp-bandwidth* [priority]
          |  |  |  +--ro priority     uint8
          |  |  |  +--ro bandwidth?   te-bandwidth
          |  |  +--ro max-link-bandwidth?        te-bandwidth
          |  |  +--ro max-resv-link-bandwidth?   te-bandwidth
          |  |  +--ro unreserved-bandwidth* [priority]
          |  |  |  +--ro priority     uint8
          |  |  |  +--ro bandwidth?   te-bandwidth
          |  |  +--ro te-default-metric?         uint32
          |  |  +--ro te-delay-metric?           uint32
          |  |  +--ro te-srlgs
          |  |  |  +--ro value*   te-types:srlg
          |  |  +--ro te-nsrlgs {nsrlg}?
          |  |  |  +--ro id*   uint32
          |  |  +--ro local-link-connectivity* [link-tp-ref]
          |  |     +--ro link-tp-ref                leafref
          |  |     +--ro label-restriction* [inclusive-exclusive
   label-start]
          |  |     |  +--ro inclusive-exclusive    enumeration
          |  |     |  +--ro label-start            rt-
   types:generalized-label
          |  |     |  +--ro label-end?             rt-
   types:generalized-label
          |  |     |  +--ro range-bitmap?          binary
          |  |     +--ro is-allowed?                boolean
          |  |     +--ro underlay {te-topology-hierarchy}?
          |  |     |  +--ro enabled?                     boolean
          |  |     |  +--ro primary-path
```

```
            |  |     |  |  +--ro network-ref?    leafref
            |  |     |  |  +--ro path-element* [path-element-id]
            |  |     |  |     +--ro path-element-id    uint32
            |  |     |  |     +--ro index?             uint32
            |  |     |  |     +--ro (type)?
            |  |     |  |        +--:(ip-address)
            |  |     |  |        |  +--ro ip-address-hop
            |  |     |  |        |     +--ro address?    inet:ip-address
            |  |     |  |        |     +--ro hop-type?   te-hop-type
            |  |     |  |        +--:(as-number)
            |  |     |  |        |  +--ro as-number-hop
            |  |     |  |        |     +--ro as-number?   binary
            |  |     |  |        |     +--ro hop-type?    te-hop-type
            |  |     |  |        +--:(unnumbered-link)
            |  |     |  |        |  +--ro unnumbered-hop
            |  |     |  |        |     +--ro router-id?      inet:ip-
   address
            |  |     |  |        |     +--ro interface-id?   uint32
            |  |     |  |        |     +--ro hop-type?       te-hop-type
            |  |     |  |        +--:(label)
            |  |     |  |        |  +--ro label-hop
            |  |     |  |        |     +--ro value?   rt-
   types:generalized-label
            |  |     |  |        +--:(sid)
            |  |     |  |           +--ro sid-hop
            |  |     |  |              +--ro sid?   rt-
   types:generalized-label
            |  |     |  +--ro backup-path* [index]
            |  |     |  |  +--ro index           uint32
            |  |     |  |  +--ro network-ref?    leafref
            |  |     |  |  +--ro path-element* [path-element-id]
            |  |     |  |     +--ro path-element-id    uint32
            |  |     |  |     +--ro index?             uint32
            |  |     |  |     +--ro (type)?
            |  |     |  |        +--:(ip-address)
            |  |     |  |        |  +--ro ip-address-hop
            |  |     |  |        |     +--ro address?    inet:ip-address
            |  |     |  |        |     +--ro hop-type?   te-hop-type
            |  |     |  |        +--:(as-number)
            |  |     |  |        |  +--ro as-number-hop
            |  |     |  |        |     +--ro as-number?   binary
            |  |     |  |        |     +--ro hop-type?    te-hop-type
            |  |     |  |        +--:(unnumbered-link)
            |  |     |  |        |  +--ro unnumbered-hop
            |  |     |  |        |     +--ro router-id?      inet:ip-
   address
            |  |     |  |        |     +--ro interface-id?   uint32
```

```
            |  |     |  |        |     +--ro hop-type?       te-hop-type
            |  |     |  |        +--:(label)
            |  |     |  |        |  +--ro label-hop
            |  |     |  |        |     +--ro value?   rt-
   types:generalized-label
            |  |     |  |        +--:(sid)
            |  |     |  |           +--ro sid-hop
            |  |     |  |              +--ro sid?   rt-
   types:generalized-label
            |  |     |  +--ro protection-type?             identityref
            |  |     |  +--ro tunnel-termination-points
            |  |     |  |  +--ro source?        binary
            |  |     |  |  +--ro destination?   binary
            |  |     |  +--ro tunnels
            |  |     |     +--ro sharing?   boolean
            |  |     |     +--ro tunnel* [tunnel-name]
            |  |     |        +--ro tunnel-name    string
            |  |     |        +--ro sharing?       boolean
            |  |     +--ro max-lsp-bandwidth* [priority]
            |  |     |  +--ro priority     uint8
            |  |     |  +--ro bandwidth?   te-bandwidth
            |  |     +--ro max-link-bandwidth?        te-bandwidth
            |  |     +--ro max-resv-link-bandwidth?   te-bandwidth
            |  |     +--ro unreserved-bandwidth* [priority]
            |  |     |  +--ro priority     uint8
            |  |     |  +--ro bandwidth?   te-bandwidth
            |  |     +--ro te-default-metric?         uint32
            |  |     +--ro te-delay-metric?           uint32
            |  |     +--ro te-srlgs
            |  |     |  +--ro value*   te-types:srlg
            |  |     +--ro te-nsrlgs {nsrlg}?
            |  |        +--ro id*   uint32
            |  +--ro oper-status?                 te-types:te-oper-
   status
            |  +--ro geolocation
            |     +--ro altitude?    int64
            |     +--ro latitude?    geographic-coordinate-degree
            |     +--ro longitude?   geographic-coordinate-degree
            +--ro statistics
            |  +--ro discontinuity-time          yang:date-and-time
            |  +--ro tunnel-termination-point
            |  |  +--ro disables?             yang:counter32
            |  |  +--ro enables?              yang:counter32
            |  |  +--ro maintenance-clears?   yang:counter32
            |  |  +--ro maintenance-sets?     yang:counter32
            |  |  +--ro modifies?             yang:counter32
            |  |  +--ro downs?                yang:counter32
```

```
            |  |  +--ro ups?                  yang:counter32
            |  |  +--ro in-service-clears?    yang:counter32
            |  |  +--ro in-service-sets?      yang:counter32
            |  +--ro local-link-connectivity
            |     +--ro creates?    yang:counter32
            |     +--ro deletes?    yang:counter32
            |     +--ro disables?   yang:counter32
            |     +--ro enables?    yang:counter32
            |     +--ro modifies?   yang:counter32
           +--rw supporting-tunnel-termination-point* [node-ref tunnel-
   tp-ref]
              +--rw node-ref         inet:uri
              +--rw tunnel-tp-ref    binary
   augment /nw:networks/nw:network/nt:link:
      +--rw te!
         +--rw config
         |  +--rw (bundle-stack-level)?
         |  |  +--:(bundle)
         |  |  |  +--rw bundled-links
         |  |  |     +--rw bundled-link* [sequence]
         |  |  |        +--rw sequence      uint32
         |  |  |        +--rw src-tp-ref?   leafref
         |  |  |        +--rw des-tp-ref?   leafref
         |  |  +--:(component)
         |  |     +--rw component-links
         |  |        +--rw component-link* [sequence]
         |  |           +--rw sequence             uint32
         |  |           +--rw src-interface-ref?   string
         |  |           +--rw des-interface-ref?   string
         |  +--rw te-link-template*     leafref {template}?
         |  +--rw te-link-attributes
         |     +--rw access-type?                      te-types:te-link-
   access-type
         |     +--rw external-domain
         |     |  +--rw network-ref?            leafref
         |     |  +--rw remote-te-node-id?      te-types:te-node-id
         |     |  +--rw remote-te-link-tp-id?   te-types:te-tp-id
         |     |  +--rw plug-id?                uint32
         |     +--rw is-abstract?                      empty
         |     +--rw name?                             string
         |     +--rw underlay {te-topology-hierarchy}?
         |     |  +--rw enabled?                    boolean
         |     |  +--rw primary-path
         |     |  |  +--rw network-ref?    leafref
         |     |  |  +--rw path-element* [path-element-id]
         |     |  |     +--rw path-element-id    uint32
         |     |  |     +--rw index?             uint32
```

```
          |      |  |     +--rw (type)?
          |      |  |        +--:(ip-address)
          |      |  |        |  +--rw ip-address-hop
          |      |  |        |     +--rw address?    inet:ip-address
          |      |  |        |     +--rw hop-type?   te-hop-type
          |      |  |        +--:(as-number)
          |      |  |        |  +--rw as-number-hop
          |      |  |        |     +--rw as-number?   binary
          |      |  |        |     +--rw hop-type?    te-hop-type
          |      |  |        +--:(unnumbered-link)
          |      |  |        |  +--rw unnumbered-hop
          |      |  |        |     +--rw router-id?      inet:ip-address
          |      |  |        |     +--rw interface-id?   uint32
          |      |  |        |     +--rw hop-type?       te-hop-type
          |      |  |        +--:(label)
          |      |  |        |  +--rw label-hop
          |      |  |        |     +--rw value?   rt-types:generalized-
   label
          |      |  |        +--:(sid)
          |      |  |           +--rw sid-hop
          |      |  |              +--rw sid?   rt-types:generalized-label
          |      |  +--rw backup-path* [index]
          |      |  |  +--rw index           uint32
          |      |  |  +--rw network-ref?    leafref
          |      |  |  +--rw path-element* [path-element-id]
          |      |  |     +--rw path-element-id    uint32
          |      |  |     +--rw index?             uint32
          |      |  |     +--rw (type)?
          |      |  |        +--:(ip-address)
          |      |  |        |  +--rw ip-address-hop
          |      |  |        |     +--rw address?    inet:ip-address
          |      |  |        |     +--rw hop-type?   te-hop-type
          |      |  |        +--:(as-number)
          |      |  |        |  +--rw as-number-hop
          |      |  |        |     +--rw as-number?   binary
          |      |  |        |     +--rw hop-type?    te-hop-type
          |      |  |        +--:(unnumbered-link)
          |      |  |        |  +--rw unnumbered-hop
          |      |  |        |     +--rw router-id?      inet:ip-address
          |      |  |        |     +--rw interface-id?   uint32
          |      |  |        |     +--rw hop-type?       te-hop-type
          |      |  |        +--:(label)
          |      |  |        |  +--rw label-hop
          |      |  |        |     +--rw value?   rt-types:generalized-
   label
          |      |  |        +--:(sid)
          |      |  |           +--rw sid-hop
```

```
          |     |  |              +--rw sid?   rt-types:generalized-label
          |     |  +--rw protection-type?             identityref
          |     |  +--rw tunnel-termination-points
          |     |  |  +--rw source?        binary
          |     |  |  +--rw destination?   binary
          |     |  +--rw tunnels
          |     |     +--rw sharing?   boolean
          |     |     +--rw tunnel* [tunnel-name]
          |     |        +--rw tunnel-name    string
          |     |        +--rw sharing?       boolean
          |     +--rw admin-status?                      te-types:te-
   admin-status
          |     +--rw link-index?                        uint64
          |     +--rw administrative-group?              te-types:admin-
   groups
          |     +--rw interface-switching-capability* [switching-
   capability encoding]
          |     |  +--rw switching-capability    identityref
          |     |  +--rw encoding                identityref
          |     |  +--rw max-lsp-bandwidth* [priority]
          |     |     +--rw priority     uint8
          |     |     +--rw bandwidth?   te-bandwidth
          |     +--rw link-protection-type?              enumeration
          |     +--rw max-link-bandwidth?                te-bandwidth
          |     +--rw max-resv-link-bandwidth?           te-bandwidth
          |     +--rw unreserved-bandwidth* [priority]
          |     |  +--rw priority     uint8
          |     |  +--rw bandwidth?   te-bandwidth
          |     +--rw te-default-metric?                 uint32
          |     +--rw te-delay-metric?                   uint32
          |     +--rw te-srlgs
          |     |  +--rw value*   te-types:srlg
          |     +--rw te-nsrlgs {nsrlg}?
          |        +--rw id*   uint32
          +--ro state
          |  +--ro (bundle-stack-level)?
          |  |  +--:(bundle)
          |  |  |  +--ro bundled-links
          |  |  |     +--ro bundled-link* [sequence]
          |  |  |        +--ro sequence      uint32
          |  |  |        +--ro src-tp-ref?   leafref
          |  |  |        +--ro des-tp-ref?   leafref
          |  |  +--:(component)
          |  |     +--ro component-links
          |  |        +--ro component-link* [sequence]
          |  |           +--ro sequence             uint32
          |  |           +--ro src-interface-ref?   string
```

```
         |  |          +--ro des-interface-ref?   string
         |  +--ro te-link-template*           leafref {template}?
         |  +--ro te-link-attributes
         |  |  +--ro access-type?                      te-types:te-link-
   access-type
         |  |  +--ro external-domain
         |  |  |  +--ro network-ref?            leafref
         |  |  |  +--ro remote-te-node-id?      te-types:te-node-id
         |  |  |  +--ro remote-te-link-tp-id?   te-types:te-tp-id
         |  |  |  +--ro plug-id?                uint32
         |  |  +--ro is-abstract?                      empty
         |  |  +--ro name?                             string
         |  |  +--ro underlay {te-topology-hierarchy}?
         |  |  |  +--ro enabled?                    boolean
         |  |  |  +--ro primary-path
         |  |  |  |  +--ro network-ref?    leafref
         |  |  |  |  +--ro path-element* [path-element-id]
         |  |  |  |     +--ro path-element-id    uint32
         |  |  |  |     +--ro index?             uint32
         |  |  |  |     +--ro (type)?
         |  |  |  |        +--:(ip-address)
         |  |  |  |        |  +--ro ip-address-hop
         |  |  |  |        |     +--ro address?    inet:ip-address
         |  |  |  |        |     +--ro hop-type?   te-hop-type
         |  |  |  |        +--:(as-number)
         |  |  |  |        |  +--ro as-number-hop
         |  |  |  |        |     +--ro as-number?   binary
         |  |  |  |        |     +--ro hop-type?    te-hop-type
         |  |  |  |        +--:(unnumbered-link)
         |  |  |  |        |  +--ro unnumbered-hop
         |  |  |  |        |     +--ro router-id?      inet:ip-address
         |  |  |  |        |     +--ro interface-id?   uint32
         |  |  |  |        |     +--ro hop-type?       te-hop-type
         |  |  |  |        +--:(label)
         |  |  |  |        |  +--ro label-hop
         |  |  |  |        |     +--ro value?   rt-types:generalized-
   label
         |  |  |  |        +--:(sid)
         |  |  |  |           +--ro sid-hop
         |  |  |  |              +--ro sid?   rt-types:generalized-label
         |  |  |  +--ro backup-path* [index]
         |  |  |  |  +--ro index           uint32
         |  |  |  |  +--ro network-ref?    leafref
         |  |  |  |  +--ro path-element* [path-element-id]
         |  |  |  |     +--ro path-element-id    uint32
         |  |  |  |     +--ro index?             uint32
         |  |  |  |     +--ro (type)?
```

```
          |  |  |  |        +--:(ip-address)
          |  |  |  |        |  +--ro ip-address-hop
          |  |  |  |        |     +--ro address?    inet:ip-address
          |  |  |  |        |     +--ro hop-type?   te-hop-type
          |  |  |  |        +--:(as-number)
          |  |  |  |        |  +--ro as-number-hop
          |  |  |  |        |     +--ro as-number?   binary
          |  |  |  |        |     +--ro hop-type?    te-hop-type
          |  |  |  |        +--:(unnumbered-link)
          |  |  |  |        |  +--ro unnumbered-hop
          |  |  |  |        |     +--ro router-id?      inet:ip-address
          |  |  |  |        |     +--ro interface-id?   uint32
          |  |  |  |        |     +--ro hop-type?       te-hop-type
          |  |  |  |        +--:(label)
          |  |  |  |        |  +--ro label-hop
          |  |  |  |        |     +--ro value?   rt-types:generalized-
   label
          |  |  |  |        +--:(sid)
          |  |  |  |           +--ro sid-hop
          |  |  |  |              +--ro sid?   rt-types:generalized-label
          |  |  |  +--ro protection-type?             identityref
          |  |  |  +--ro tunnel-termination-points
          |  |  |  |  +--ro source?        binary
          |  |  |  |  +--ro destination?   binary
          |  |  |  +--ro tunnels
          |  |  |     +--ro sharing?   boolean
          |  |  |     +--ro tunnel* [tunnel-name]
          |  |  |        +--ro tunnel-name    string
          |  |  |        +--ro sharing?       boolean
          |  |  +--ro admin-status?                      te-types:te-
   admin-status
          |  |  +--ro link-index?                        uint64
          |  |  +--ro administrative-group?              te-types:admin-
   groups
          |  |  +--ro interface-switching-capability* [switching-
   capability encoding]
          |  |  |  +--ro switching-capability    identityref
          |  |  |  +--ro encoding                identityref
          |  |  |  +--ro max-lsp-bandwidth* [priority]
          |  |  |     +--ro priority     uint8
          |  |  |     +--ro bandwidth?   te-bandwidth
          |  |  +--ro link-protection-type?              enumeration
          |  |  +--ro max-link-bandwidth?                te-bandwidth
          |  |  +--ro max-resv-link-bandwidth?           te-bandwidth
          |  |  +--ro unreserved-bandwidth* [priority]
          |  |  |  +--ro priority     uint8
          |  |  |  +--ro bandwidth?   te-bandwidth
```

```
         |  |  +--ro te-default-metric?               uint32
         |  |  +--ro te-delay-metric?                 uint32
         |  |  +--ro te-srlgs
         |  |  |  +--ro value*   te-types:srlg
         |  |  +--ro te-nsrlgs {nsrlg}?
         |  |     +--ro id*   uint32
         |  +--ro oper-status?               te-types:te-oper-status
         |  +--ro is-transitional?           empty
         |  +--ro information-source?        te-info-source
         |  +--ro information-source-state
         |  |  +--ro credibility-preference?    uint16
         |  |  +--ro logical-network-element?   string
         |  |  +--ro network-instance?          string
         |  |  +--ro topology
         |  |     +--ro network-ref?   leafref
         |  |     +--ro link-ref?      leafref
         |  +--ro information-source-entry* [information-source]
         |  |  +--ro information-source               te-info-source
         |  |  +--ro information-source-state
         |  |  |  +--ro credibility-preference?    uint16
         |  |  |  +--ro logical-network-element?   string
         |  |  |  +--ro network-instance?          string
         |  |  |  +--ro topology
         |  |  |     +--ro network-ref?   leafref
         |  |  |     +--ro link-ref?      leafref
         |  |  +--ro link-index?                      uint64
         |  |  +--ro administrative-group?            te-types:admin-
   groups
         |  |  +--ro interface-switching-capability* [switching-
   capability encoding]
         |  |  |  +--ro switching-capability    identityref
         |  |  |  +--ro encoding                identityref
         |  |  |  +--ro max-lsp-bandwidth* [priority]
         |  |  |     +--ro priority     uint8
         |  |  |     +--ro bandwidth?   te-bandwidth
         |  |  +--ro link-protection-type?            enumeration
         |  |  +--ro max-link-bandwidth?              te-bandwidth
         |  |  +--ro max-resv-link-bandwidth?         te-bandwidth
         |  |  +--ro unreserved-bandwidth* [priority]
         |  |  |  +--ro priority     uint8
         |  |  |  +--ro bandwidth?   te-bandwidth
         |  |  +--ro te-default-metric?               uint32
         |  |  +--ro te-delay-metric?                 uint32
         |  |  +--ro te-srlgs
         |  |  |  +--ro value*   te-types:srlg
         |  |  +--ro te-nsrlgs {nsrlg}?
         |  |     +--ro id*   uint32
```

```
         |  +--ro recovery
         |  |  +--ro restoration-status?   te-types:te-recovery-status
         |  |  +--ro protection-status?    te-types:te-recovery-status
         |  +--ro underlay {te-topology-hierarchy}?
         |     +--ro dynamic?     boolean
         |     +--ro committed?   boolean
         +--ro statistics
            +--ro discontinuity-time                yang:date-and-time
            +--ro disables?                         yang:counter32
            +--ro enables?                          yang:counter32
            +--ro maintenance-clears?               yang:counter32
            +--ro maintenance-sets?                 yang:counter32
            +--ro modifies?                         yang:counter32
            +--ro downs?                            yang:counter32
            +--ro ups?                              yang:counter32
            +--ro fault-clears?                     yang:counter32
            +--ro fault-detects?                    yang:counter32
            +--ro protection-switches?              yang:counter32
            +--ro protection-reverts?               yang:counter32
            +--ro restoration-failures?             yang:counter32
            +--ro restoration-starts?               yang:counter32
            +--ro restoration-successes?            yang:counter32
            +--ro restoration-reversion-failures?   yang:counter32
            +--ro restoration-reversion-starts?     yang:counter32
            +--ro restoration-reversion-successes?  yang:counter32
   augment /nw:networks/nw:network/nw:node/nt:termination-point:
      +--rw te-tp-id?   te-types:te-tp-id
      +--rw te!
         +--rw config
         |  +--rw admin-status?                    te-types:te-admin-
   status
         |  +--rw name?                            string
         |  +--rw interface-switching-capability* [switching-capability
   encoding]
         |  |  +--rw switching-capability    identityref
         |  |  +--rw encoding                identityref
         |  |  +--rw max-lsp-bandwidth* [priority]
         |  |     +--rw priority     uint8
         |  |     +--rw bandwidth?   te-bandwidth
         |  +--rw inter-layer-lock-id*             uint32
         +--ro state
            +--ro admin-status?                    te-types:te-admin-
   status
            +--ro name?                            string
            +--ro interface-switching-capability* [switching-capability
   encoding]
            |  +--ro switching-capability    identityref
```

```
            |  +--ro encoding                identityref
            |  +--ro max-lsp-bandwidth* [priority]
            |     +--ro priority     uint8
            |     +--ro bandwidth?   te-bandwidth
            +--ro inter-layer-lock-id*              uint32
            +--ro oper-status?                      te-types:te-oper-
   status
            +--ro geolocation
               +--ro altitude?    int64
               +--ro latitude?    geographic-coordinate-degree
               +--ro longitude?   geographic-coordinate-degree
```

7. TE Topology Yang Module

```
   <CODE BEGINS> file "ietf-te-topology@2017-06-10.yang"
   module ietf-te-topology {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-te-topology";

     prefix "tet";

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

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

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

     import ietf-network {
       prefix "nw";
     }

     import ietf-network-topology {
       prefix "nt";
     }

     import ietf-routing-types {
       prefix "rt-types";
```

```
     }

     organization
       "Traffic Engineering Architecture and Signaling (TEAS)
        Working Group";

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

        WG Chair: Lou Berger
                  <mailto:lberger@labn.net>

        WG Chair: Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Xufeng Liu
                  <mailto:Xufeng_Liu@jabil.com>

        Editor:   Igor Bryskin
                  <mailto:Igor.Bryskin@huawei.com>

        Editor:   Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

        Editor:   Tarek Saad
                  <mailto:tsaad@cisco.com>

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

        Editor:   Oscar Gonzalez De Dios
                  <mailto:oscar.gonzalezdedios@telefonica.com>";

     description "TE topology model";

     revision "2017-06-10" {
       description "Initial revision";
       reference "TBD";
     }
```

```
     /*
      * Features
      */
     feature nsrlg {
       description
         "This feature indicates that the system supports NSRLG
          (Not Sharing Risk Link Group).";
     }

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

     feature template {
       description
         "This feature indicates that the system supports
          template configuration.";
     }

     /*
      * Typedefs
      */
     typedef geographic-coordinate-degree {
         type decimal64 {
           fraction-digits 8;
         }
         description
           "Decimal degree (DD) used to express latitude and longitude
            geographic coordinates.";
     } // geographic-coordinate-degree

     typedef te-bandwidth {
       type string {
         pattern
           '0[xX](0((\.0?)?[pP](\+)?0?|(\.0?))|'
         + '1(\.([\da-fA-F]{0,5}[02468aAcCeE]?)?)?[pP](\+)?(12[0-7]|'
         + '1[01]\d|0?\d?\d)?)|0[xX][\da-fA-F]{1,8}|\d+'
         + '(,(0[xX](0((\.0?)?[pP](\+)?0?|(\.0?))|'
         + '1(\.([\da-fA-F]{0,5}[02468aAcCeE]?)?)?[pP](\+)?(12[0-7]|'
```

```
        + '1[01]\d|0?\d?\d)?)|0[xX][\da-fA-F]{1,8}|\d+))*';
      }
      description
        "This is the generic bandwidth type that is a string containing
         a list of numbers separated by commas, with each of these
         number can be non-negative decimal, hex integer, or hex float:
         (dec | hex | float)[*(','(dec | hex | float))]
         For packet switching type, a float number is used, such as
         0x1p10.
         For OTN switching type, a list of integers can be used, such
         as '0,2,3,1', indicating 2 odu0's and 1 odu3.
         For DWDM, a list of pairs of slot number and width can be
         used, such as '0, 2, 3, 3', indicating a frequency slot 0 with
         slot width 2 and a frequency slot 3 with slot width 3.";
    } // te-bandwidth

    typedef te-info-source {
      type enumeration {
        enum "unknown" {
          description "The source is unknown.";
        }
        enum "locally-configured" {
          description "Configured entity.";
        }
        enum "ospfv2" {
          description "OSPFv2.";
        }
        enum "ospfv3" {
          description "OSPFv3.";
        }
        enum "isis" {
          description "ISIS.";
        }
        enum "bgp-ls" {
          description "BGP-LS.";
          reference
            "RFC7752: North-Bound Distribution of Link-State and
             Traffic Engineering (TE) Information Using BGP";
        }
        enum "system-processed" {
          description "System processed entity.";
```

```
        }
        enum "other" {
          description "Other source.";
        }
      }
      description
        "Describining the type of source that has provided the
         related information, and the source credibility.";
    } // te-info-source

    typedef te-path-disjointness {
      type bits {
        bit node {
          position 0;
          description "Node disjoint.";
        }
        bit link {
          position 1;
          description "Link disjoint.";
        }
        bit srlg {
          position 2;
          description "SRLG (Shared Risk Link Group) disjoint.";
        }
      }
      description
        "Type of the resource disjointness for a TE tunnel path.";
      reference
        "RFC4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS)
         Recovery";
    } // te-path-disjointness

    /*
     * Groupings
     */
    grouping connectivity-label-restriction-list {
      description
        "List of abel restrictions specifying what labels may or may
         not be used on a link connectivity.";
      list label-restriction {
```

```
        key "inclusive-exclusive label-start";
        description
          "List of abel restrictions specifying what labels may or may
           not be used on a link connectivity.";
        reference
          "RFC7579: General Network Element Constraint Encoding
           for GMPLS-Controlled Networks";
        leaf inclusive-exclusive {
          type enumeration {
            enum inclusive {
              description "The label or label range is inclusive.";
            }
            enum exclusive {
              description "The label or label range is exclusive.";
            }
          }
          description
            "Whether the list item is inclusive or exclusive.";
        }
        leaf label-start {
          type rt-types:generalized-label;
          description
            "This is the starting lable if a lable range is specified.
             This is the lable value if a single lable is specified,
             in which case, attribute 'label-end' is not set.";
        }
        leaf label-end {
          type rt-types:generalized-label;
          description
            "The ending lable if a lable range is specified;
             This attribute is not set, If a single lable is
             specified.";
        }
        leaf range-bitmap {
          type binary;
          description
            "When there are gaps between label-start and label-end,
             this attribute is used to specified the possitions
             of the used labels.";
        }
      }
```

```
    } // connectivity-label-restriction-list

    grouping connectivity-matrix-entry-attributes {
      description
        "Attributes of connectivity matrix entry.";
      leaf is-allowed {
        type boolean;
        description
          "true  - switching is allowed,
           false - switching is disallowed.";
      }
      container underlay {
        if-feature te-topology-hierarchy;
        description "Attributes of the te-link underlay.";
        reference
          "RFC4206: Label Switched Paths (LSP) Hierarchy with
           Generalized Multi-Protocol Label Switching (GMPLS)
           Traffic Engineering (TE)";

        uses te-link-underlay-attributes;
      } // underlay
      uses te-link-iscd-attributes;
      uses te-link-connectivity-attributes;
    } // connectivity-matrix-entry-attributes

    grouping geolocation-container {
      description
        "A container containing a GPS location.";
      container geolocation{
        description
          "A container containing a GPS location.";
        leaf altitude {
          type int64;
          units millimeter;
          description
            "Distance above the sea level.";
        }
        leaf latitude {
          type geographic-coordinate-degree {
            range "-90..90";
          }
```

```
          description
            "Relative position north or south on the Earth's surface.";
        }
        leaf longitude {
          type geographic-coordinate-degree {
            range "-180..180";
          }
          description
            "Angular distance east or west on the Earth's surface.";
        }
      } // gps-location
    } // geolocation-container

    grouping information-source-state-attributes {
      description
        "The attributes identifying source that has provided the
         related information, and the source credibility.";
      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.";
      }
      leaf logical-network-element {
        type string;
        description
          "When applicable, this is the name of a logical network
           element from which the information is learned.";
      } // logical-network-element
      leaf network-instance {
        type string;
        description
          "When applicable, this is the name of a network-instance
           from which the information is learned.";
      } // network-instance
    } // information-source-state-attributes

    grouping information-source-per-link-attributes {
```

```
      description
        "Per node container of the attributes identifying source that
         has provided the related information, and the source
         credibility.";
      leaf information-source {
        type te-info-source;
        description
          "Indicates the source of the information.";
      }
      container information-source-state {
        description
          "The container contains state attributes related to
           the information source.";
        uses information-source-state-attributes;
        container topology {
          description
            "When the information is processed by the system,
             the attributes in this container indicate which topology
             is used to process to generate the result information.";
          uses te-topology-ref;
          leaf link-ref {
            type leafref {
              path "/nw:networks/nw:network[nw:network-id = "
                + "current()/../network-ref]/nt:link/nt:link-id";
              require-instance false;
            }
            description
              "A reference to a link-id.";
          }
        } // topology
      } // information-source-state
    } // information-source-per-link-attributes

    grouping information-source-per-node-attributes {
      description
        "Per node container of the attributes identifying source that
         has provided the related information, and the source
         credibility.";
      leaf information-source {
        type te-info-source;
        description
```

```
            "Indicates the source of the information.";
        }
        container information-source-state {
          description
            "The container contains state attributes related to
             the information source.";
          uses information-source-state-attributes;
          container topology {
            description
              "When the information is processed by the system,
               the attributes in this container indicate which topology
               is used to process to generate the result information.";
            uses te-topology-ref;
            leaf node-ref {
              type leafref {
                path "/nw:networks/nw:network[nw:network-id = "
                  + "current()/../network-ref]/nw:node/nw:node-id";
                require-instance false;
              }
              description
                "A reference to a node-id.";
            }
          } // topology
        } // information-source-state
      } // information-source-per-node-attributes

      grouping interface-switching-capability-list {
        description
          "List of Interface Switching Capabilities Descriptors (ISCD)";

        list interface-switching-capability {
          key "switching-capability encoding";
          description
            "List of Interface Switching Capabilities Descriptors (ISCD)
             for this link.";
          reference
            "RFC3471: Generalized Multi-Protocol Label Switching (GMPLS)
             Signaling Functional Description.
             RFC4203: OSPF Extensions in Support of Generalized
             Multi-Protocol Label Switching (GMPLS).";
          leaf switching-capability {
```

```
          type identityref {
            base te-types:switching-capabilities;
          }
          description
            "Switching Capability for this interface.";
        }
        leaf encoding {
          type identityref {
            base te-types:lsp-encoding-types;
          }
          description
            "Encoding supported by this interface.";
        }
        uses te-link-iscd-attributes;
      } // interface-switching-capability
    } // interface-switching-capability-list

    grouping statistics-per-link {
      description
        "Statistics attributes per TE link.";
      leaf discontinuity-time {
        type yang:date-and-time;
        mandatory true;
        description
          "The time on the most recent occasion at which any one or
           more of this interface's counters suffered a
           discontinuity.  If no such discontinuities have occurred
           since the last re-initialization of the local management
           subsystem, then this node contains the time the local
           management subsystem re-initialized itself.";
      }
      /* Administrative attributes */
      leaf disables {
        type yang:counter32;
        description
          "Number of times that link was disabled.";
      }
      leaf enables {
        type yang:counter32;
        description
          "Number of times that link was enabled.";
```

```
      }
      leaf maintenance-clears {
        type yang:counter32;
        description
          "Number of times that link was put out of maintenance.";
      }
      leaf maintenance-sets {
        type yang:counter32;
        description
          "Number of times that link was put in maintenance.";
        }
      leaf modifies {
        type yang:counter32;
        description
          "Number of times that link was modified.";
      }
      /* Operational attributes */
      leaf downs {
        type yang:counter32;
        description
          "Number of times that link was set to operational down.";
      }
      leaf ups {
        type yang:counter32;
        description
          "Number of times that link was set to operational up.";
      }
      /* Recovery attributes */
      leaf fault-clears {
        type yang:counter32;
        description
          "Number of times that link experienced fault clear event.";
      }
      leaf fault-detects {
        type yang:counter32;
        description
          "Number of times that link experienced fault detection.";
      }
      leaf protection-switches {
        type yang:counter32;
        description
```

```
            "Number of times that link experienced protection
             switchover.";
          }
          leaf protection-reverts {
            type yang:counter32;
            description
              "Number of times that link experienced protection
               reversion.";
          }
          leaf restoration-failures {
            type yang:counter32;
            description
              "Number of times that link experienced restoration
               failure.";
          }
          leaf restoration-starts {
            type yang:counter32;
            description
              "Number of times that link experienced restoration
               start.";
          }
          leaf restoration-successes {
            type yang:counter32;
            description
              "Number of times that link experienced restoration
               success.";
          }
          leaf restoration-reversion-failures {
            type yang:counter32;
            description
              "Number of times that link experienced restoration reversion
               failure.";
          }
          leaf restoration-reversion-starts {
            type yang:counter32;
            description
              "Number of times that link experienced restoration reversion
               start.";
          }
          leaf restoration-reversion-successes {
            type yang:counter32;
```

```
          description
            "Number of times that link experienced restoration reversion
             success.";
        }
      } // statistics-per-link

      grouping statistics-per-node {
        description
          "Statistics attributes per TE node.";
        leaf discontinuity-time {
          type yang:date-and-time;
          mandatory true;
          description
            "The time on the most recent occasion at which any one or
             more of this interface's counters suffered a
             discontinuity.  If no such discontinuities have occurred
             since the last re-initialization of the local management
             subsystem, then this node contains the time the local
             management subsystem re-initialized itself.";
        }
        container node {
          description
            "Containing TE node level statistics attributes.";
          leaf disables {
            type yang:counter32;
            description
              "Number of times that node was disabled.";
          }
          leaf enables {
            type yang:counter32;
            description
              "Number of times that node was enabled.";
          }
          leaf maintenance-sets {
            type yang:counter32;
            description
              "Number of times that node was put in maintenance.";
          }
          leaf maintenance-clears {
            type yang:counter32;
            description
```

```
            "Number of times that node was put out of maintenance.";
          }
          leaf modifies {
            type yang:counter32;
            description
              "Number of times that node was modified.";
          }
        } // node
        container connectivity-matrix-entry {
          description
            "Containing connectivity matrix entry level statistics
             attributes.";
          leaf creates {
            type yang:counter32;
            description
              "Number of times that a connectivity matrix entry was
               created.";
            reference
              "RFC6241. Section 7.2 for 'create' operation. ";
          }
          leaf deletes {
            type yang:counter32;
            description
              "Number of times that a connectivity matrix entry was
               deleted.";
            reference
              "RFC6241. Section 7.2 for 'delete' operation. ";
          }
          leaf disables {
            type yang:counter32;
            description
              "Number of times that a connectivity matrix entry was
               disabled.";
          }
          leaf enables {
            type yang:counter32;
            description
              "Number of times that a connectivity matrix entry was
               enabled.";
          }
          leaf modifies {
```

```
            type yang:counter32;
            description
              "Number of times that a connectivity matrix entry was
               modified.";
          }
        } // connectivity-matrix-entry
      } // statistics-per-node

      grouping statistics-per-ttp {
        description
          "Statistics attributes per TE TTP (Tunnel Termination Point).";
        leaf discontinuity-time {
          type yang:date-and-time;
          mandatory true;
          description
            "The time on the most recent occasion at which any one or
             more of this interface's counters suffered a
             discontinuity.  If no such discontinuities have occurred
             since the last re-initialization of the local management
             subsystem, then this node contains the time the local
             management subsystem re-initialized itself.";
        }
        container tunnel-termination-point {
          description
            "Containing TE TTP (Tunnel Termination Point) level
             statistics attributes.";
          /* Administrative attributes */
          leaf disables {
            type yang:counter32;
            description
              "Number of times that TTP was disabled.";
          }
          leaf enables {
            type yang:counter32;
            description
              "Number of times that TTP was enabled.";
          }
          leaf maintenance-clears {
            type yang:counter32;
            description
              "Number of times that TTP was put out of maintenance.";
```

```
          }
          leaf maintenance-sets {
            type yang:counter32;
            description
              "Number of times that TTP was put in maintenance.";
          }
          leaf modifies {
            type yang:counter32;
            description
              "Number of times that TTP was modified.";
          }
          /* Operational attributes */
          leaf downs {
            type yang:counter32;
            description
              "Number of times that TTP was set to operational down.";
          }
          leaf ups {
            type yang:counter32;
            description
              "Number of times that TTP was set to operational up.";
          }
          leaf in-service-clears {
            type yang:counter32;
            description
              "Number of times that TTP was taken out of service
               (TE tunnel was released).";
          }
          leaf in-service-sets {
            type yang:counter32;
            description
              "Number of times that TTP was put in service by a TE
               tunnel (TE tunnel was set up).";
          }
        } // tunnel-termination-point

        container local-link-connectivity {
          description
            "Containing TE LLCL (Local Link Connectivity List) level
             statistics attributes.";
          leaf creates {
```

```
          type yang:counter32;
          description
            "Number of times that an LLCL entry was created.";
          reference
            "RFC6241. Section 7.2 for 'create' operation. ";
        }
        leaf deletes {
          type yang:counter32;
          description
            "Number of times that an LLCL entry was deleted.";
          reference
            "RFC6241. Section 7.2 for 'delete' operation.";
        }
        leaf disables {
          type yang:counter32;
          description
            "Number of times that an LLCL  entry was disabled.";
        }
        leaf enables {
          type yang:counter32;
          description
            "Number of times that an LLCL  entry was enabled.";
        }
        leaf modifies {
          type yang:counter32;
          description
            "Number of times that an LLCL  entry was modified.";
        }
      } // local-link-connectivity
    } // statistics-per-ttp

    grouping te-link-config {
      description
        "TE link configuration grouping.";
      choice bundle-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.";
          reference
            "RFC4201: Link Bundling in MPLS Traffic Engineering
            (TE).";
          list bundled-link {
            key "sequence";
            description
              "Specify a bundled interface that is
               further partitioned.";
            leaf sequence {
              type uint32;
              description
                "Identify the sequence in the bundle.";
            }
            leaf src-tp-ref {
              type leafref {
                path "../../../../../../nw:node[nw:node-id = "
                  + "current()/../../../../../nt:source/"
                  + "nt:source-node]/"
                  + "nt:termination-point/nt:tp-id";
                require-instance true;
              }
              description
                "Reference to another TE termination point on the
                 same souruce node.";
            }
            leaf des-tp-ref {
              type leafref {
                path "../../../../../../nw:node[nw:node-id = "
                  + "current()/../../../../../nt:destination/"
                  + "nt:dest-node]/"
                  + "nt:termination-point/nt:tp-id";
                require-instance true;
              }
              description
                "Reference to another TE termination point on the
                 same destination node.";
            }
          } // list bundled-link
        }
      }
```

```
          case component {
            container component-links {
              description
                "A set of component links";
              list component-link {
                key "sequence";
                description
                  "Specify a component interface that is
                   sufficient to unambiguously identify the
                   appropriate resources";

                leaf sequence {
                  type uint32;
                  description
                    "Identify the sequence in the bundle.";
                }
                leaf src-interface-ref {
                  type string;
                  description
                    "Reference to component link interface on the
                     source node.";
                }
                leaf des-interface-ref {
                  type string;
                  description
                    "Reference to component link interface on the
                     destinatioin node.";
                }
              }
            }
          }
        } // bundle-stack-level

        leaf-list te-link-template {
          if-feature template;
          type leafref {
            path "../../../../../te/templates/link-template/name";
          }
          description
            "The reference to a TE link template.";
        }
```

```
      uses te-link-config-attributes;
    } // te-link-config

    grouping te-link-config-attributes {
      description
        "Link configuration attributes in a TE topology.";
      container te-link-attributes {
        description "Link attributes in a TE topology.";
        leaf access-type {
          type te-types:te-link-access-type;
          description
            "Link access type, which can be point-to-point or
             multi-access.";
        }
        container external-domain {
          description
            "For an inter-domain link, specify the attributes of
             the remote end of link, to facilitate the signalling at
             local end.";
          uses te-topology-ref;
          leaf remote-te-node-id {
            type te-types:te-node-id;
            description
              "Remote TE node identifier, used together with
               remote-te-link-id to identify the remote link
               termination point in a different domain.";
          }
          leaf remote-te-link-tp-id {
            type te-types:te-tp-id;
            description
              "Remote TE link termination point identifier, used
               together with remote-te-node-id to identify the remote
               link termination point in a different domain.";
          }
          leaf plug-id {
            type uint32;
            description
              "A topology-wide unique number that identifies on the
               network a connectivity supporting a given inter-domain
               TE link. This is more flexible alternative to specifying
               remote-te-node-id and remote-te-link-tp-id, when the
```

```
                provider does not know remote-te-node-id and
                remote-te-link-tp-id or need to give client the
                flexibility to mix-n-match multiple topologies.";
            }
          }
          leaf is-abstract {
            type empty;
            description "Present if the link is abstract.";
          }
          leaf name {
            type string;
            description "Link Name.";
          }
          container underlay {
            if-feature te-topology-hierarchy;
            description "Attributes of the te-link underlay.";
            reference
              "RFC4206: Label Switched Paths (LSP) Hierarchy with
               Generalized Multi-Protocol Label Switching (GMPLS)
               Traffic Engineering (TE)";

            uses te-link-underlay-attributes;
          } // underlay
          leaf admin-status {
            type te-types:te-admin-status;
            description
              "The administrative state of the link.";
          }

          uses te-link-info-attributes;
        } // te-link-attributes
      } // te-link-config-attributes

      grouping te-link-connectivity-attributes {
        description
          "Advertised TE connectivity attributes.";
        leaf max-link-bandwidth {
          type te-bandwidth;
          description
            "Maximum bandwidth that can be seen on this link in this
             direction. Units in bytes per second.";
```

```
          reference
            "RFC3630: Traffic Engineering (TE) Extensions to OSPF
             Version 2.
             RFC5305: IS-IS Extensions for Traffic Engineering.";
        }
        leaf max-resv-link-bandwidth {
          type te-bandwidth;
          description
            "Maximum amount of bandwidth that can be reserved in this
             direction in this link. Units in bytes per second.";
          reference
            "RFC3630: Traffic Engineering (TE) Extensions to OSPF
             Version 2.
             RFC5305: IS-IS Extensions for Traffic Engineering.";
        }
        list unreserved-bandwidth {
          key "priority";
          max-elements "8";
          description
            "Unreserved bandwidth for 0-7 priority levels. Units in
             bytes per second.";
          reference
            "RFC3630: Traffic Engineering (TE) Extensions to OSPF
             Version 2.
             RFC5305: IS-IS Extensions for Traffic Engineering.";
          leaf priority {
            type uint8 {
              range "0..7";
            }
            description "Priority.";
          }
          leaf bandwidth {
            type te-bandwidth;
            description
              "Unreserved bandwidth for this level.";
          }
        }
        leaf te-default-metric {
          type uint32;
          description
            "Traffic engineering metric.";
```

```
      }
      leaf te-delay-metric {
        type uint32;
        description
          "Traffic engineering delay metric.";
      }
      container te-srlgs {
        description
          "Containing a list of SLRGs.";
        leaf-list value {
          type te-types:srlg;
          description "SRLG value.";
          reference
            "RFC4202: Routing Extensions in Support of
             Generalized Multi-Protocol Label Switching (GMPLS).";
        }
      }
      container te-nsrlgs {
        if-feature nsrlg;
        description
          "Containing a list of NSRLGs (Not Sharing Risk Link
           Groups).
           When an abstract TE link is configured, this list specifies
           the request that underlay TE paths need to be mutually
           disjoint with other TE links in the same groups.";
        leaf-list id {
          type uint32;
          description
            "NSRLG ID, uniquely configured within a topology.";
          reference
            "RFC4872: RSVP-TE Extensions in Support of End-to-End
             Generalized Multi-Protocol Label Switching (GMPLS)
             Recovery";
        }
      }
    } // te-link-connectivity-attributes

    grouping te-link-info-attributes {
      description
        "Advertised TE information attributes.";
      leaf link-index {
```

```
          type uint64;
          description
            "The link identifier.  If OSPF is used, this represents an
             ospfLsdbID.  If IS-IS is used, this represents an isisLSPID.
             If a locally configured link is used, this object represents
             a unique value, which is locally defined in a router.";
        }
        leaf administrative-group {
          type te-types:admin-groups;
          description
            "Administrative group or color of the link.
             This attribute covers both administrative group (defined in
             RFC3630, RFC5329, and RFC5305), and extended administrative
             group (defined in RFC7308).";
        }
        uses interface-switching-capability-list;
        leaf link-protection-type {
          type enumeration {
            enum "unprotected" {
              description "Unprotected.";
            }
            enum "extra-traffic" {
              description "Extra traffic.";
            }
            enum "shared" {
              description "Shared.";
            }
            enum "1-for-1" {
              description "One for one protection.";
            }
            enum "1-plus-1" {
              description "One plus one protection.";
            }
            enum "enhanced" {
              description "Enhanced protection.";
            }
          }
          description
            "Link Protection Type desired for this link.";
          reference
            "RFC4202: Routing Extensions in Support of
```

```
            Generalized Multi-Protocol Label Switching (GMPLS).";
        }
        uses te-link-connectivity-attributes;
      } // te-link-info-attributes

      grouping te-link-iscd-attributes {
        description
          "TE link ISCD (Interface Switching Capability Descriptor)
           attributes.";
        reference
          "Sec 1.4, RFC4203: OSPF Extensions in Support of Generalized
           Multi-Protocol Label Switching (GMPLS). Section 1.4.";
        list max-lsp-bandwidth {
          key "priority";
          max-elements "8";
          description
            "Maximum LSP Bandwidth at priorities 0-7.";
          leaf priority {
            type uint8 {
              range "0..7";
            }
            description "Priority.";
          }
          leaf bandwidth {
            type te-bandwidth;
            description
              "Max LSP Bandwidth for this level";
          }
        }
      } // te-link-iscd-attributes

      grouping te-link-state-derived {
        description
          "Link state attributes in a TE topology.";
        leaf oper-status {
          type te-types:te-oper-status;
          description
            "The current operational state of the link.";
        }
        leaf is-transitional {
          type empty;
```

```
          description
            "Present if the link is transitional, used as an
             alternative approach in lieu of inter-layer-lock-id
             for path computation in a TE topology covering multiple
             layers or multiple regions.";
          reference
            "RFC5212: Requirements for GMPLS-Based Multi-Region and
             Multi-Layer Networks (MRN/MLN).
             RFC6001: Generalized MPLS (GMPLS) Protocol Extensions
             for Multi-Layer and Multi-Region Networks (MLN/MRN).";
        }
        uses information-source-per-link-attributes;
        list information-source-entry {
          key "information-source";
          description
            "A list of information sources learned, including the one
             used.";
          uses information-source-per-link-attributes;
          uses te-link-info-attributes;
        }
        container recovery {
          description
            "Status of the recovery process.";
          leaf restoration-status {
            type te-types:te-recovery-status;
            description
              "Restoration status.";
          }
          leaf protection-status {
            type te-types:te-recovery-status;
            description
              "Protection status.";
          }
        }
        container underlay {
          if-feature te-topology-hierarchy;
          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-derived

    grouping te-link-underlay-attributes {
      description "Attributes for  te-link underlay.";
      reference
        "RFC4206: Label Switched Paths (LSP) Hierarchy with
         Generalized Multi-Protocol Label Switching (GMPLS)
         Traffic Engineering (TE)";
      leaf enabled {
        type boolean;
        description
          "'true' if the underlay is enabled.
           'false' if the underlay is disabled.";
      }
      container primary-path {
        description
          "The service path on the underlay topology that
           supports this link.";
        uses te-topology-ref;
        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;
        }
      } // primary-path
      list backup-path {
        key "index";
        description
          "A list of backup service paths on the underlay topology that
```

```
            protect the underlay primary path. If the primary path is
            not protected, the list contains zero elements. If the
            primary path is protected, the list contains one or more
            elements.";
          leaf index {
            type uint32;
            description
              "A sequence number to identify a backup path.";
          }
          uses te-topology-ref;
          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 protection-type {
          type identityref {
            base te-types:lsp-prot-type;
          }
          description
            "Underlay protection type desired for this link.";
        }
        container tunnel-termination-points {
          description
            "Underlay TTP(Tunnel Termination Points) desired for this
             link.";
          leaf source {
            type binary;
            description
              "Source tunnel termination point identifier.";
          }
          leaf destination {
            type binary;
            description
```

```
            "Destination tunnel termination point identifier.";
        }
      }
      container tunnels {
        description
          "Underlay TE tunnels supporting this TE link.";
        leaf sharing {
          type boolean;
          default true;
          description
            "'true' if the underlay tunnel can be shared with other
             TE links;
             'false' if the underlay tunnel is dedicated to this
             TE link.
             This leaf is the default option for all TE tunnels,
             and may be overridden by the per TE tunnel value.";
        }
        list tunnel {
          key "tunnel-name";
          description
            "Zero, one or more underlay TE tunnels that support this TE
             link.";
          leaf tunnel-name {
            type string;
            description
              "A tunnel name uniquely identifies an underlay TE tunnel,
               used together with the source-node of this link.
               The detailed information of this tunnel can be retrieved
               from the ietf-te model.";
            reference "RFC3209";
          }
          leaf sharing {
            type boolean;
            description
              "'true' if the underlay tunnel can be shared with other
               TE links;
               'false' if the underlay tunnel is dedicated to this
               TE link.";
          }
        } // tunnel
      } // tunnels
```

```
    } // te-link-underlay-attributes

    grouping te-node-config {
      description "TE node configuration grouping.";

      leaf-list te-node-template {
        if-feature template;
        type leafref {
          path "../../../../../te/templates/node-template/name";
        }
        description
          "The reference to a TE node template.";
      }
      uses te-node-config-attributes;
    } // te-node-config

    grouping te-node-config-attributes {
      description "Configuration node attributes in a TE topology.";
      container te-node-attributes {
        description "Containing node attributes in a TE topology.";
        leaf admin-status {
          type te-types:te-admin-status;
          description
            "The administrative state of the link.";
        }
        uses te-node-connectivity-matrix;
        uses te-node-info-attributes;
      } // te-node-attributes
    } // te-node-config-attributes

    grouping te-node-config-attributes-template {
      description
        "Configuration node attributes for template in a TE topology.";
      container te-node-attributes {
        description "Containing node attributes in a TE topology.";
        leaf admin-status {
          type te-types:te-admin-status;
          description
            "The administrative state of the link.";
        }
        uses te-node-info-attributes;
```

```
      } // te-node-attributes
    } // te-node-config-attributes-template

    grouping te-node-connectivity-matrix {
      description "Connectivity matrix on a TE node.";
      container connectivity-matrices {
        description
          "Containing connectivity matrix on a TE node.";
        leaf number-of-entries {
          type uint16;
          description
            "The number of connectivity matrix entries.
             If this number is speficied in the configuration request,
             the number is requested number of entries, which may not
             all be listed in the list;
             if this number is reported in the state data,
             the number is the current number of operational entries.";
        }
        uses connectivity-label-restriction-list;
        uses connectivity-matrix-entry-attributes;
        list connectivity-matrix {
          key "id";
          description
            "Represents node's switching limitations, i.e. limitations
             in interconnecting network TE links across the node.";
          reference
            "RFC7579: General Network Element Constraint Encoding
             for GMPLS-Controlled Networks.";
          leaf id {
            type uint32;
            description "Identifies the connectivity-matrix entry.";
          }
          container from {
            description
              "Reference to source link termination point.";
            leaf tp-ref {
              type leafref {
                path "../../../../../../../nt:termination-point/"+
                  "nt:tp-id";
              }
              description
```

```
                "Relative reference to source termination point.";
            }
            uses connectivity-label-restriction-list;
          }
          container to {
            description
              "Reference to destination link termination point.";
            leaf tp-ref {
              type leafref {
                path "../../../../../../../nt:termination-point/"+
                  "nt:tp-id";
              }
              description
                "Relative reference to destination termination point.";
            }
            uses connectivity-label-restriction-list;
          }
          uses connectivity-matrix-entry-attributes;
        } // connectivity-matrix
      } // connectivity-matrices
    } // te-node-connectivity-matrix

    grouping te-node-connectivity-matrix-abs {
      description
        "Connectivity matrix on a TE node, using absolute
         paths to reference termination points.";
      list connectivity-matrix {
        key "id";
        description
          "Represents node's switching limitations, i.e. limitations
           in interconnecting network TE links across the node.";
        reference
          "RFC7579: General Network Element Constraint Encoding
           for GMPLS-Controlled Networks.";
        leaf id {
          type uint32;
          description "Identifies the connectivity-matrix entry.";
        }
        container from {
          uses nt:tp-ref;
          description
```

```
            "Reference to source NTP.";
        }
        container to {
          uses nt:tp-ref;
          description
            "Reference to destination NTP.";
        }
        leaf is-allowed {
          type boolean;
          description
            "true  - switching is allowed,
             false - switching is disallowed.";
        }
      }
    } // te-node-connectivity-matrix-abs

    grouping te-node-info-attributes {
      description
        "Advertised TE information attributes.";
      leaf domain-id {
        type uint32;
        description
          "Identifies the domain that this node belongs.
           This attribute is used to support inter-domain links.";
        reference
          "RFC5152: A Per-Domain Path Computation Method for
           Establishing Inter-Domain Traffic Engineering (TE)
           Label Switched Paths (LSPs).
           RFC5392: OSPF Extensions in Support of Inter-Autonomous
           System (AS) MPLS and GMPLS Traffic Engineering.
           RFC5316: ISIS Extensions in Support of Inter-Autonomous
           System (AS) MPLS and GMPLS Traffic Engineering.";
      }
      leaf is-abstract {
        type empty;
        description
          "Present if the node is abstract, not present if the node
           is actual.";
      }
      leaf name {
        type inet:domain-name;
```

```
        description "Node name.";
      }
      leaf-list signaling-address {
        type inet:ip-address;
        description "Node signaling address.";
      }
      container underlay-topology {
        if-feature te-topology-hierarchy;
        description
          "When an abstract node encapsulates a topology,
           the attributes in this container point to said topology.";
        uses te-topology-ref;
      }
    } // te-node-info-attributes

    grouping te-node-state-derived {
      description "Node state attributes in a TE topology.";
      leaf oper-status {
        type te-types:te-oper-status;
        description
          "The current operational state of the node.";
      }
      uses geolocation-container;
      leaf is-multi-access-dr {
        type empty;
        description
          "The presence of this attribute indicates that this TE node
           is a pseudonode elected as a designated router.";
        reference
          "RFC3630: Traffic Engineering (TE) Extensions to OSPF
           Version 2.
           RFC1195: Use of OSI IS-IS for Routing in TCP/IP and Dual
           Environments.";
      }
      uses information-source-per-node-attributes;
      list information-source-entry {
        key "information-source";
        description
          "A list of information sources learned, including the one
           used.";
        uses information-source-per-node-attributes;
```

```
        uses te-node-connectivity-matrix;
        uses te-node-info-attributes;
      }
    } // te-node-state-derived

    grouping te-node-tunnel-termination-attributes {
      description
        "Termination capability of a tunnel termination point on a
         TE node.";

      leaf admin-status {
        type te-types:te-admin-status;
        description
          "The administrative state of the tunnel termination point.";
      }
      leaf name {
        type string;
        description
          "A descriptive name for the tunnel termination point.";
      }
      leaf switching-capability {
        type identityref {
          base te-types:switching-capabilities;
        }
        description
          "Switching Capability for this interface.";
      }
      leaf encoding {
        type identityref {
          base te-types:lsp-encoding-types;
        }
        description
          "Encoding supported by this interface.";
      }
      leaf-list inter-layer-lock-id {
        type uint32;
        description
          "Inter layer lock ID, used for path computation in a TE
           topology covering multiple layers or multiple regions.";
        reference
          "RFC5212: Requirements for GMPLS-Based Multi-Region and
```

```
            Multi-Layer Networks (MRN/MLN).
            RFC6001: Generalized MPLS (GMPLS) Protocol Extensions
            for Multi-Layer and Multi-Region Networks (MLN/MRN).";
        }
        leaf protection-type {
          type identityref {
            base te-types:lsp-prot-type;
          }
          description
            "The protection type that this tunnel termination point
             is capable of.";
        }

        container client-layer-adaptation {
          description
            "Containing capability information to support a client layer
             adaption in multi-layer topology.";
          list switching-capability {
            key "switching-capability encoding";
            description
              "List of supported switching capabilities";
            reference
              "RFC6001: Generalized MPLS (GMPLS) Protocol Extensions
               for Multi-Layer and Multi-Region Networks (MLN/MRN).
               RFC4202: Routing Extensions in Support of
               Generalized Multi-Protocol Label Switching (GMPLS).";
            leaf switching-capability {
              type identityref {
                base te-types:switching-capabilities;
              }
              description
                "Switching Capability for the client layer adaption.";
            }
            leaf encoding {
              type identityref {
                base te-types:lsp-encoding-types;
              }
              description
                "Encoding supported by the client layer adaption.";
            }
            leaf bandwidth {
```

```
            type te-bandwidth;
            description
              "Bandwidth available for the client layer adaption.";
          }
        }
      }

      container local-link-connectivities {
        description
          "Containing local link connectivity list for
           a tunnel termination point on a TE node.";
        leaf number-of-entries {
          type uint16;
          description
            "The number of local link connectivity list entries.
             If this number is speficied in the configuration request,
             the number is requested number of entries, which may not
             all be listed in the list;
             if this number is reported in the state data,
             the number is the current number of operational entries.";
        }
        uses connectivity-label-restriction-list;
        uses connectivity-matrix-entry-attributes;
        list local-link-connectivity {
          key "link-tp-ref";
          description
            "The termination capabilities between
             tunnel-termination-point and link termination-point.
             The capability information can be used to compute
             the tunnel path.
             The Interface Adjustment Capability Descriptors (IACD)
             [RFC6001] on each link-tp can be derived from this
             local-link-connectivity list.";
          reference
            "RFC6001: Generalized MPLS (GMPLS) Protocol Extensions
             for Multi-Layer and Multi-Region Networks (MLN/MRN).";
          leaf link-tp-ref {
            type leafref {
              path "../../../../../../nt:termination-point/nt:tp-id";
            }
            description
```

```
              "Link termination point.";
          }

          uses connectivity-label-restriction-list;
          uses connectivity-matrix-entry-attributes;
        } // local-link-connectivity
      } // local-link-connectivities
    } // te-node-tunnel-termination-attributes

    grouping te-path-element {
      description
        "A group of attributes defining an element in a TE path
         such as TE node, TE link, TE atomic resource or label.";
      uses te-types:explicit-route-hop_config;
    } // te-path-element

    grouping te-termination-point-config {
      description
        "TE termination point configuration grouping.";
      leaf admin-status {
        type te-types:te-admin-status;
        description
          "The administrative state of the link termination point.";
      }
      leaf name {
        type string;
        description
          "A descriptive name for the link termination point.";
      }
      uses interface-switching-capability-list;
      leaf-list inter-layer-lock-id {
        type uint32;
        description
          "Inter layer lock ID, used for path computation in a TE
           topology covering multiple layers or multiple regions.";
        reference
          "RFC5212: Requirements for GMPLS-Based Multi-Region and
           Multi-Layer Networks (MRN/MLN).
           RFC6001: Generalized MPLS (GMPLS) Protocol Extensions
           for Multi-Layer and Multi-Region Networks (MLN/MRN).";
      }
```

```
    } // te-termination-point-config

    grouping te-topology-config {
      description
        "TE topology configuration grouping.";
      leaf preference {
        type uint8 {
          range "1..255";
        }
        description
          "Specifies a preference for this topology. A lower number
           indicates a higher preference.";
      }
      leaf optimization-criterion {
        type identityref {
          base te-types:te-optimization-criterion;
        }
        description
          "Optimization criterion applied to this topology.";
        reference
          "RFC3272: Overview and Principles of Internet Traffic
           Engineering.";
      }
      list nsrlg {
        if-feature nsrlg;
        key "id";
        description
          "List of NSRLGs (Not Sharing Risk Link Groups).";
        reference
          "RFC4872: RSVP-TE Extensions in Support of End-to-End
           Generalized Multi-Protocol Label Switching (GMPLS)
           Recovery";
        leaf id {
          type uint32;
          description
            "Identify the NSRLG entry.";
        }
        leaf disjointness {
          type te-path-disjointness;
          description
            "The type of resource disjointness.";
```

```
        }
      } // nsrlg
    } // te-topology-config

    grouping te-topology-ref {
      description
        "References a TE topology.";
      leaf network-ref {
        type leafref {
          path "/nw:networks/nw:network/nw:network-id";
          require-instance false;
        }
        description
          "A reference to a network-id in base ietf-network module.";
      }
    } // te-topology-ref

    grouping template-attributes {
      description
        "Common attributes for all templates.";

      leaf priority {
        type uint16;
        description
          "The preference value to resolve conflicts between different
           templates. When two or more templates specify values for
           one configuration attribute, the value from the template
           with the highest priority is used.";
      }
      leaf reference-change-policy {
        type enumeration {
          enum no-action {
            description
              "When an attribute changes in this template, the
               configuration node referring to this template does
               not take any action.";
          }
          enum not-allowed {
            description
              "When any configuration object has a reference to this
               template, changing this template is not allowed.";
```

```
            }
            enum cascade {
              description
                "When an attribute changes in this template, the
                 configuration object referring to this template applies
                 the new attribute value to the corresponding
                 configuration.";
            }
          }
          description
            "This attribute specifies the action taken to a configuration
             node that has a reference to this template.";
        }
      } // template-attributes

      /*
       * Configuration data nodes
       */
      augment "/nw:networks/nw:network/nw:network-types" {
        description
          "Introduce new network type for TE topology.";
        container te-topology {
          presence "Indicates TE topology.";
          description
            "Its presence identifies the TE topology type.";
        }
      }

      augment "/nw:networks" {
        description
          "Augmentation parameters for TE topologies.";

        container te {
          presence "TE support.";
          description
            "Indicates TE support.";

          container templates {
            description
              "Configuration parameters for templates used for TE
               topology.";
```

```
          list node-template {
            if-feature template;
            key "name";
            leaf name {
              type te-types: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 template-attributes;
            uses te-node-config-attributes-template;
          } // node-template

          list link-template {
            if-feature template;
            key "name";
            leaf name {
              type te-types: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 template-attributes;
            uses te-link-config-attributes;
          } // link-template
        } // templates
      } // te
    }

    augment "/nw:networks/nw:network" {
      when "nw:network-types/te-topology" {
        description
          "Augmentation parameters apply only for networks with
           TE topology type.";
      }
      description
```

```
        "Configuration parameters for TE topology.";

      leaf provider-id {
        type te-types:te-global-id;
        description
          "An identifier to uniquely identify a provider.";
      }
      leaf client-id {
        type te-types:te-global-id;
        description
          "An identifier to uniquely identify a client.";
      }
      leaf te-topology-id {
        type te-types: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 te {
        must "../provider-id and ../client-id and ../te-topology-id";
        presence "TE support.";
        description
          "Indicates TE support.";

        container config {
          description
            "Configuration data.";
          uses te-topology-config;
        } // config
        container state {
          config false;
          description
            "Operational state data.";
          uses te-topology-config;
          uses geolocation-container;
        } // state
      } // te
    }
```

```
     augment "/nw:networks/nw:network/nw:node" {
       when "../nw:network-types/te-topology" {
         description
           "Augmentation parameters apply only for networks with
            TE topology type.";
       }
       description
         "Configuration parameters for TE at node level.";

       leaf te-node-id {
         type te-types:te-node-id;
         description
           "The identifier of a node in the TE topology.
            A node is specific to a topology to which it belongs.";
       }

       container te {
         must "../te-node-id" {
           description
             "te-node-id is mandatory.";
         }
         must "count(../nw:supporting-node)<=1" {
           description
             "For a node in a TE topology, there cannot be more
              than 1 supporting node. If multiple nodes are abstracted,
              the underlay-topology is used.";
         }
         presence "TE support.";
         description
           "Indicates TE support.";

         container config {
           description
             "Configuration data.";
           uses te-node-config;
         } // config
         container state {
           config false;
           description
             "Operational state data.";
```

```
          uses te-node-config;
          uses te-node-state-derived;
        } // state
        container statistics {
          config false;
          description
            "Statistics data.";
          uses statistics-per-node;
        } // statistics

        list tunnel-termination-point {
          key "tunnel-tp-id";
          description
            "A termination point can terminate a tunnel.";
          leaf tunnel-tp-id {
            type binary;
            description
              "Tunnel termination point identifier.";
          }

          container config {
            description
              "Configuration data.";
            uses te-node-tunnel-termination-attributes;
          }
          container state {
            config false;
            description
              "Operational state data.";

            uses te-node-tunnel-termination-attributes;
            leaf oper-status {
              type te-types:te-oper-status;
              description
                "The current operational state of the tunnel
                 termination point.";
            }
            uses geolocation-container;
          } // state
          container statistics {
            config false;
```

```
            description
              "Statistics data.";
            uses statistics-per-ttp;
          } // statistics

          // Relations to other tunnel termination points
          list supporting-tunnel-termination-point {
            key "node-ref tunnel-tp-ref";
            description
              "Identifies the tunnel termination points, that this
               tunnel termination point is depending on.";
            leaf node-ref {
              type inet:uri;
              description
                "This leaf identifies the node in which the supporting
                 tunnel termination point is present.
                 This node is either the supporting node or a node in
                 an underlay topology.";
            }
            leaf tunnel-tp-ref {
              type binary;
              description
                "Reference to a tunnel terminiation point, which is
                 either in the supporting node or a node in an
                 underlay topology.";
            }
          } // supporting-tunnel-termination-point
        } // tunnel-termination-point
      } // te
    }

    augment "/nw:networks/nw:network/nt:link" {
      when "../nw:network-types/te-topology" {
        description
          "Augmentation parameters apply only for networks with
           TE topology type.";
      }
      description
        "Configuration parameters for TE at link level";

      container te {
```

```
        must "count(../nt:supporting-link)<=1" {
          description
            "For a link in a TE topology, there cannot be more
             than 1 supporting link. If one or more link paths are
             abstracted, the underlay is used.";
        }
        presence "TE support.";
        description
          "Indicates TE support.";

        container config {
          description
            "Configuration data.";
          uses te-link-config;
        } // config
        container state {
          config false;
          description
            "Operational state data.";
          uses te-link-config;
          uses te-link-state-derived;
        } // state
        container statistics {
          config false;
          description
            "Statistics data.";
          uses statistics-per-link;
        } // statistics
      } // te
    }

    augment "/nw:networks/nw:network/nw:node/"
          + "nt:termination-point" {
      when "../../nw:network-types/te-topology" {
        description
          "Augmentation parameters apply only for networks with
           TE topology type.";
      }
      description
        "Configuration parameters for TE at termination point level";
```

```
       leaf te-tp-id {
         type te-types:te-tp-id;
         description
           "An identifier to uniquely identify a TE termination
            point.";
       }

       container te {
         must "../te-tp-id";
         presence "TE support.";
         description
           "Indicates TE support.";

         container config {
           description
             "Configuration data.";
           uses te-termination-point-config;
         } // config
         container state {
           config false;
           description
             "Operational state data.";
           uses te-termination-point-config;
           leaf oper-status {
             type te-types:te-oper-status;
             description
               "The current operational state of the link termination
                point.";
           }
           uses geolocation-container;
         } // state
       } // te
     }
   }
   <CODE ENDS>
```

8. Security Considerations

The transport protocol used for retrieving/manipulating the TE topology data MUST support authentication and SHOULD support encryption. The data-model by itself does not create any security implications.

9. 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-topology
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-topology
namespace: urn:ietf:params:xml:ns:yang:ietf-te-topology
prefix: tet

10. References

10.1. 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.

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

[YANG-NET-TOPO] Clemm, A., "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo (Work in Progress).

[YANG-PUSH] Clemm, A., "Subscribing to YANG datastore push updates", draft-clemm-netconf-yang-push (Work in Progress).

[RFC5277bis] Clemm, A., "Subscribing to Event Notifications", draft-ietf-netconf-rfc5277bis (Work in Progress).

[YANG-SCHEDULE] Liu, X., " A YANG Data Model for Configuration Scheduling", draft-liu-netmod-yang-schedule-00 (Work in Progress).

10.2. Informative References

[RFC2702] Awduche, D., "Requirements for Traffic Engineering Over MPLS", RFC 2702, September 1999.

11. Acknowledgments

The authors would like to thank Lou Berger, Sue Hares, Mazen Khaddam, Cyril Margaria and Zafar Ali for participating in design discussions and providing valuable insights.

Contributors

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

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

Authors' Addresses

Xufeng Liu
Jabil
Email: Xufeng_Liu@jabil.com

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

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

Tarek Saad
Cisco Systems Inc
Email: tsaad@cisco.com

Himanshu Shah

Ciena
Email: hshah@ciena.com

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

TEAS WG

Young Lee
Dhruv Dhody
Huawei

Daniele Ceccarelli
Ericsson

Oscar Gonzalez de Dios
Telefonica

Internet Draft
Intended status: Informational

Expires: December 2017

June 5, 2017

Abstraction and Control of TE Networks (ACTN) Abstraction Methods

draft-lee-teas-actn-abstraction-02

Status of this Memo

Copyright Notice

Abstract

Abstraction and Control of Traffic Engineering (TE) Networks(ACTN) refers to the set of virtual network operations needed to orchestrate, control and manage large-scale multi-domain TE networks, so as to facilitate network programmability, automation, and efficient resource sharing.

As the ACTN architecture considers abstraction as one of the important building blocks, this document describes a few alternatives methods of abstraction for both packet and optical networks. This is an important consideration since the choice of the abstraction method impacts protocol design and the information it carries.

Table of Contents

1. Introduction

Abstraction and Control of TE Networks (ACTN) describes a method for operating a Traffic Engineered (TE) network (such as an MPLS-TE network or a layer 1 transport network) to provide connectivity and virtual network services for customers of the TE network. The services provided can be tuned to meet the requirements (such as traffic patterns, quality, and reliability) of the applications hosted by the customers. More details about ACTN can be found in Section 2.

Abstraction is defined in [RFC7926] as:

> Abstraction is the process of applying policy to the available TE information within a domain, to produce selective information that represents the potential ability to connect across the domain. Thus, abstraction does not necessarily offer all possible connectivity options, but presents a general view of potential connectivity according to the policies that determine how the domain's administrator wants to allow the domain resources to be used.

Connectivity referred to this document is TE path through a series of connected domains as used in [RFC7926].

As the ACTN architecture considers abstraction as one of the important building blocks, this document discusses a few alternatives for the methods of abstraction for both packet and optical networks. This is an important consideration since the choice of the abstraction method impacts protocol design and the information it carries.

The purpose of this document is to find a common agreement on the factors and methods of abstraction. These abstraction factors and methods may in turn impact implementations and protocol design.

2. Abstraction Factors in ACTN Architecture

This section provides abstraction factors in the ACTN architecture. [ACTN-Frame] describes the architecture model for ACTN including the entities (Customer Network Controller (CNC), Multi-domain Service

Coordinator (MDSC), and Physical Network Controller (PNC) and their interfaces.

The MDSC oversees the specific aspects of the different domains and builds a single abstracted end-to-end network topology in order to coordinate end-to-end path computation and path/service provisioning. In order for the MDSC to perform its coordination function, it depends on the coordination with the PNCs which are the domain-level controllers especially as to what level of domain network resource abstraction is agreed upon between the MDSC and the PNCs.

As discussed in [RFC7926], abstraction is tied with policy of the networks. For instance, per an operational policy, the PNC would not be allowed to provide any technology specific details (e.g., optical parameters for WSON) in its update. In such case, the abstraction level of the update will be in a generic nature. In order for the MDSC to get technology specific topology information from the PNC, a request/reply mechanism may be employed.

In some cases, abstraction is also tied with the controller’s capability of abstraction as abstraction involves some rules and algorithms to be applied to the actual network resource information (which is also known as network topology).

[TE-Topology] describes YANG models for TE-network abstraction. [PCEP-LS] describes PCEP Link-state mechanism that also allows for transport of abstract topology in the context of Hierarchical PCE.

There are factors that may impact the choice of abstraction and presents a number of abstraction methods. It is important to understand that abstraction depends on several factors:

- The nature of underlying domain networks: Abstraction depends on the nature of the underlying domain networks. For instance, packet networks may have different level of abstraction requirements from that of optical networks. Within optical networks, WSON may have different level of abstraction requirements than the OTN networks.

- The capability of the PNC: Abstraction depends on the capability of the PNCs. As abstraction requires hiding details of the underlying resource network resource information, the PNC capability to run some internal optimization algorithm impacts the feasibility of abstraction. Some PNC may not have the ability to abstract native topology while other PNCs may have such an ability to abstract actual topology by using sophisticated algorithms.

- Scalability factor: Abstraction is a function of scalability. If the actual network resource information is of small size, then the need for abstraction would be less than the case where the native network resource information is of large size. In some cases, abstraction may not be needed at all.

- The frequency of topology updates: The proper abstraction level may depend on the frequency of topology updates and vice versa.

- The capability/nature of the MDSC: The nature of the MDSC impacts the degree/level of abstraction. If the MDSC is not capable of handling optical parameters such as those specific to OTN/WSON, then white topology abstraction may not work well.

- The confidentiality: In some cases where the PNC would like to hide key internal topological data from the MDSC, the abstraction method should consider this aspect.

- The scope of abstraction: All of the aforementioned factors are equally applicable to both the MPI (MDSC-PNC Interface) and the CMI (CNC-MDSC Interface).

[ACTN-Framework] defined the following three levels of topology abstraction and their descriptions:

. White topology: this is a case where the PNC provides the actual network topology to the MDSC without any hiding or filtering.

. Black topology: the entire domain network is abstracted as a single virtual node (see the definition of virtual node in [RFC7926]) with the access/egress links without disclosing any node internal connectivity information.

. Grey topology: this abstraction level is between black topology and white topology from a granularity point of view. we may further differentiate from a perspective of how to abstract internal TE resources between the pairs of border nodes:

  o Grey topology type A: border nodes with a TE links between them in a full mesh fashion
  o Grey topology type B: border nodes with some internal abstracted nodes and abstracted links

3. Build Method of Grey Topology

This section discusses two different methods of building a grey topology:

- Automatic generation of abstract topology by configuration (Section 3.1)
- On-demand generation of supplementary topology via path computation request/reply (Section 3.2)

3.1. Automatic generation of abstract topology by configuration

The "Automatic generation" method is based on the abstraction/summarization of the whole domain by the PNC and its advertisement on MPI interface once the abstraction level is configured. The level of abstraction advertisement can be decided based on some PNC configuration parameters (e.g. provide the potential connectivity between any PE and any ASBR in an MPLS-TE network as described in section 3.3.1)

Note that the configuration parameters for this potential topology can include available B/W, latency, or any combination of defined parameters. How to generate such tunnel information is beyond the scope of this document. Appendix A provides one example of this method for the WSON case.

Such potential topology needs to be periodically or incrementally/asynchronously updated every time that a failure, a recovery or the setup of new VNs causes a change in the characteristics of the advertised grey topology (e.g. in our previous case if due to changes in the network is it now possible to provide connectivity between a given PE and a given ASBR with a higher delay in the update).

3.2. On-demand generation of supplementary topology via path compute request/reply

The "on-demand generation" of supplementary topology is to be distinguished from automatic generation of abstract topology. While abstract topology is generated and updated automatically by configuration as explained in Section 3.1., additional supplementary topology may be obtained by the MDSC via path compute request/reply mechanism. Starting with a black topology advertisement from the

PNCs, the MDSC may need additional information beyond the level of black topology from the PNCs. It is assumed that the black topology advertisement from PNCs would give the MDSC each domain's the border node/link information as described in Figure 2. Under this scenario, when the MDSC needs to allocate a new VN, the MDSC can issue a number of Path Computation requests as described in [ACTN-YANG] to different PNCs with constraints matching the VN request.

An example is provided in Figure 4, where the MDSC is requesting to setup a P2P VN between AP1 and AP2. The MDSC can use two different inter-domain links to get from Domain X to Domain Y, namely the one between ASBRX.1 and ASBRY.1 and the one between ASBRX.2 and ASBRY.2, but in order to choose the best end to end path it needs to know what domain X and Y can offer in term of connectivity and constraints between the PE nodes and the ASBR nodes.

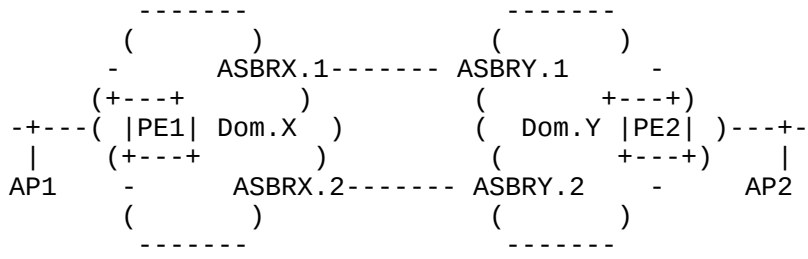


Figure 4: A multi-domain networks example

A path computation request will be issued to PNC.X asking for potential connectivity between PE1 and ASBRX.1 and between PE1 and ASBRX.2 with related objective functions and TE metric constraints. A similar request will be issued to PNC.Y and the results merged together at the MDSC to be able to compute the optimal end-to-end path including the inter domain links.

The info related to the potential connectivity may be cached by the MDSC for subsequent path computation processes or discarded, but in this case the PNCs are not requested to keep the grey topology updated.

4. Protocol/Data Model Requirements

This section provides a set of requirements that may impact the way protocol/data model is designed and the information elements thereof which are carried in the protocol/data model.

It is expected that the abstraction level be negotiated between the CNC and the MDSC (i.e., the CMI) depending on the capability of the CNC. This negotiated level of abstraction on the CMI may also impact the way the MDSC and the PNCs configure and encode the abstracted topology. For example, if the CNC is capable of sophisticated technology specific operation, then this would impact the level of abstraction at the MDSC with the PNCs. On the other hand, if the CNC asks for a generic topology abstraction, then the level of abstraction at the MDSC with the PNCs can be less technology specific than the former case.

The subsequent sections provide a list of possible abstraction levels for various technology domain networks.

4.1. Packet Networks

- For grey abstraction, the type of abstraction and its parameters MUST be defined and configured/negotiated.
    - o Abstraction Level 1: TE-tunnel abstraction for all (S-D) border pairs with:
        - . Maximum B/W available per Priority Level
        - . Minimum Latency
    - o Other Level (TBD)

4.2. OTN Networks

For OTN networks, max bandwidth available may be per ODU 0/1/2/3 switching level or aggregated across all ODU switching levels (i.e., ODUj/k).Clearly, there is a trade-off between these two abstraction methods. Some OTN switches can switch any level of ODUs and in such case there is no need for ODU level abstraction.

- For grey abstraction, the type of abstraction and its parameters MUST be defined and configured/negotiated.
    - o Abstraction Level 1: Per ODU Switching level (i.e., ODU type and number) TE-tunnel abstraction for all (S-D) border pairs with:

. Maximum B/W available per Priority Level
. Minimum Latency

o Abstraction Level 2: Aggregated TE-tunnel abstraction for all (S-D) border pairs with:
. Maximum B/W available per Priority Level
. Minimum Latency

o Other Level (TBD)

4.3. WSON Networks

For WSON networks, max bandwidth available may be per lambda/frequency level (OCh) or aggregated across all lambda/frequency level. Per OCh level abstraction gives more detailed data to the MDSC at the expense of more information processing. Either OCh-level or aggregated level abstraction should factor in the RWA constraint (i.e., wavelength continuity) at the PNC level. This means the PNC should have this capability and advertise it as such. See the Appendix for this abstraction method.

- For grey abstraction, the type of abstraction MUST and its parameters be defined and configured/negotiated.

o Abstraction Level 1: Per Lambda/Frequency level TE-tunnel abstraction for all (S-D) border pairs with:
. Maximum B/W available per Priority Level
. Minimum Latency

o Abstraction Level 2: Aggregated TE-tunnel abstraction for all (S-D) border pairs with:
. Maximum B/W available per Priority Level
. Minimum Latency

o Other Level (TBD)

Examples: these examples show how to compute WSON grey topology Abstraction Level 1 and Level 2. These examples illustrate that the encoding of an abstraction topology can be impacted by the configured/negotiated abstraction level in the ACTN interfaces.

This section provides how WSON grey topology abstraction levels 1 and 2 can be computed at a PNC. These examples illustrate that the

encoding of an abstraction topology can be impacted by the configured/negotiated abstraction level at the MPI.

. Abstraction Level 1: Per Lambda/Frequency level TE-tunnel abstraction for all (S-D) border pairs:

For each (S-D) border node pair,

1) The concept of a lambda plane: A lambda plane is a confined optical topology with respect to a given lambda value. If an OMS link has the wavelength of the given lambda available, it is included, otherwise excluded.

2) Calculate the maximal flow between S and D in every lambda plane. Max flow computation is restricted to each lambda plane is for OCh wavelength continuity.

3) Convert each feasible lambda plane with OCh wavelength continuity to B/W equivalent encoding; Send this per lambda level encoding for (S-D) to the MDSC;

. Abstraction Level 2: Aggregated TE-tunnel abstraction for WSON for all (S-D) border pairs

For each (S-D) border node pair,

1) The concept of a lambda plane: A lambda plane is a confined optical topology with respect to a given lambda value. If an OMS link has the wavelength of the given lambda available, it is included, otherwise excluded.

2) Calculate the maximal flow between S and D in every lambda plane. Max flow computation is restricted to each lambda plane is for OCh wavelength continuity.

3) Add up the max flow values across all lambda planes. This is the maximal number of OCh paths that can be setup between S and D at the same time.

4) Convert the max number of OCh paths to B/W equivalent encoding; Send this encoding as max B/W for (S-D) to the MDSC;

5. Acknowledgements

We thank Adrian Farrel and Italo Busi for providing useful comments and suggestions for this draft.

6. References

6.1. Informative References

[RFC7926] A. Farrel, Ed., "Problem Statement and Architecture for Information Exchange between Interconnected Traffic-Engineered Networks", RFC 7926, July 2016.

[ACTN-Frame] D. Cecarelli and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks", draft-ietf-teas-actn-framework, work in progress.

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

[PCEP-LS] D. Dhody, Y. Lee and D. Ceccarelli, "PCEP Extension for Distribution of Link-State and TE Information," draft-dhodylee-pce-pcep-ls, work in progress.

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

[ACTN-YANG] X. Zhang, et. Al., "Applicability of YANG models for Abstraction and Control of Traffic Engineered Networks", draft-zhang-teas-actn-yang, work in progress

7. Contributors

Contributor’s Addresses

Sergio Belotti
Nokia

Email: sergio.belotti@nokia.com

Xian Zhang
Huawei

Email: zhang.xian@huawei.com

Authors' Addresses

Young Lee
Huawei Technologies
5340 Legacy Drive
Plano, TX 75023, USA
Phone: (469)277-5838

Email: leeyoung@huawei.com

Dhruv Dhody
Huawei Technologies

Email: dhruv.ietf@gmail.com

Daniele Ceccarelli
Ericsson
Torshamnsgatan,48
Stockholm, Sweden

Email: daniele.ceccarelli@ericsson.com

Oscar Gonzalez de Dios
Telefonica

Email: oscar.gonzalezdedios@telefonica.com

TEAS WG
Internet Draft
Intended Status: standard

Young Lee
Dhruv Dhody
Satish Karunanithi
Huawei
Ricard Vilalta
CTTC
Daniel King
Lancaster University
Daniele Ceccarelli
Ericsson

Expires: September 2017

June 20, 2017

YANG models for ACTN TE Performance Monitoring Telemetry and Network Autonomics

draft-lee-teas-actn-pm-telemetry-autonomics-01

Status of this Memo

Abstract

Abstraction and Control of TE Networks (ACTN) refers to the set of virtual network operations needed to operate, control and manage large-scale multi-domain, multi-layer and multi-vendor TE networks, so as to facilitate network programmability, automation, efficient resource sharing.

This document provides YANG data models that describe Key Performance Indicator (KPI) telemetry and network autonomics for TE-tunnels and ACTN VNs.

Table of Contents

1. Introduction

Abstraction and Control of TE Networks (ACTN) describes a method for operating a Traffic Engineered (TE) network (such as an MPLS-TE network or a layer 1/0 transport network) to provide connectivity and virtual network services for customers of the TE network [ACTN-Frame]. The services provided can be optimized to meet the requirements (such as traffic patterns, quality, and reliability) of the applications hosted by the customers. Data models are a representation of objects that can be configured or monitored within a system. Within the IETF, YANG [RFC6020] is the language of choice for documenting data models, and YANG models have been produced to allow configuration or modeling of a variety of network devices, protocol instances, and network services. YANG data models have been classified in [Netmod-Yang-Model-Classification] and [Service-YANG].

[ACTN-VN-YANG] describes how customers or end to end orchestrators can request and/or instantiate a generic virtual network service. [ACTN-Applicability] describes a connection between IETF YANG model classifications to ACTN interfaces. In particular, it describes the customer service model can be mapped into the CMI (CNC-MDSC Interface) of the ACTN architecture.

The YANG model on the ACTN CMI is known as customer service model in [Service-YANG]. [PCEP-Service-Aware] describes key network performance data to be considered for end-to-end path computation in TE networks. Key performance indicator is a term that describes critical performance data that may affect VN/TE service.

2. Use-Cases

[ACTN-PERF] describes use-cases relevant to this draft. It introduces the dynamic creation, modification and optimization of services based on the performance monitoring in the Abstraction and Control of Transport Networks (ACTN) architecture. Figure 1 shows a high-level workflows for dynamic service control based on traffic monitoring.

Some of the key points from [ACTN-PERF] are as follows:

- Network traffic monitoring is important to facilitate automatic discovery of the imbalance of network traffic, and initiate the network optimization, thus helping the network operator or the virtual network service provider to use the network more efficiently and save CAPEX/OPEX.
- Customer services have various SLA requirements, such as service availability, latency, latency jitter, packet loss rate, BER, etc. The transport network can satisfy service availability and BER requirements by providing different protection and restoration mechanisms. However, for other performance parameters, there are no such mechanisms. In order to provide high quality services according to customer SLA, one possible solution is to measure the service SLA related performance parameters, and dynamically provision and optimize services based on the performance monitoring results.
- Performance monitoring in a large scale network could generate a huge amount of performance information. Therefore, the appropriate way to deliver the information in CMI and MPI interfaces should be carefully considered.

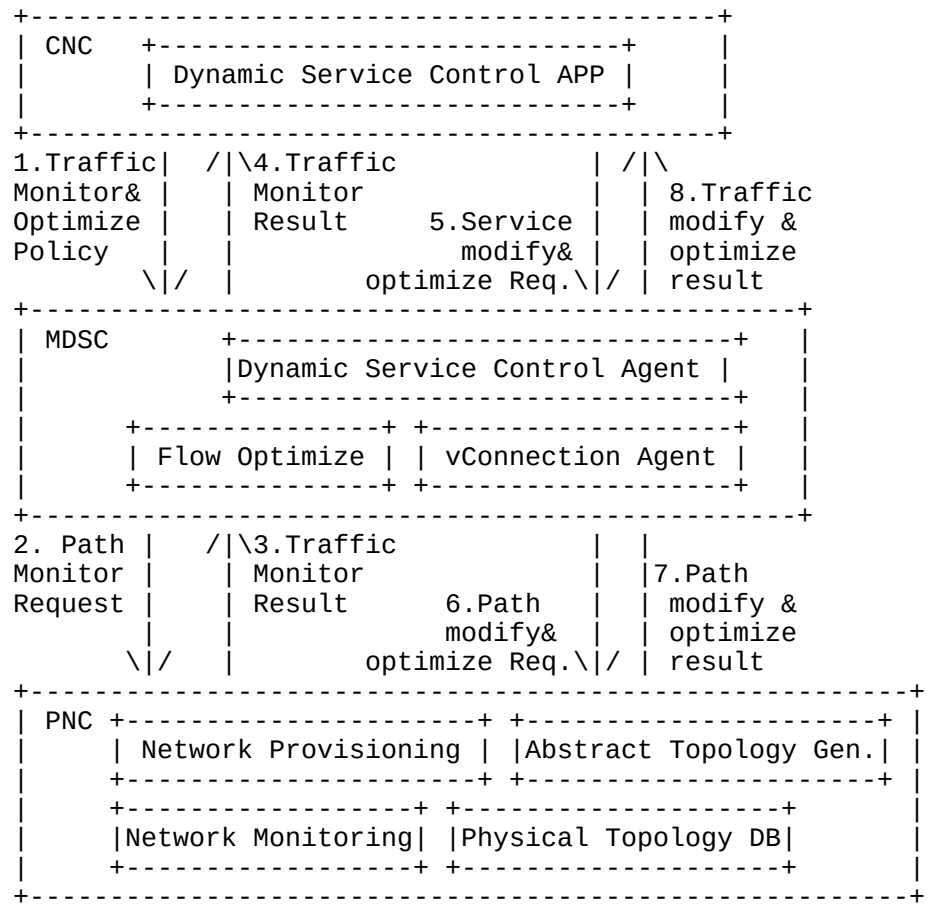


Figure 1 Workflows for dynamic service control based on traffic monitoring

3. Design of the Data Models

The YANG models developed in this document describe two models:

(i) TE KPI Telemetry Model which provides the TE-Tunnel level of performance monitoring mechanism (See Section 2.1 for details)

(ii) ACTN TE KPI Telemetry Model which provides the VN level of the aggregated performance monitoring mechanism (See Section 2.2 for details)

The models include -

(i) Performance Telemetry details as measured during the last interval, ex delay.

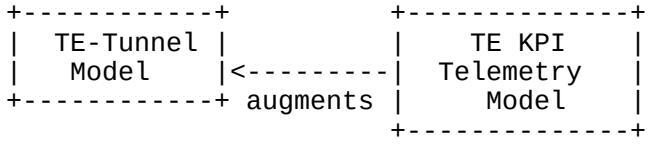
(ii) Scaling Intent based on with TE/VN could be scaled in/out.

[Editor’s Note - Need to decide if scaling and telemetry can be in the same model as per the current draft.]

TE KPI Telemetry Model

This module describes performance telemetry for TE-tunnel model. The telemetry data is augmented to tunnel state. This module also allows autonomic traffic engineering scaling intent configuration mechanism on the TE-tunnel level. Various conditions can be set for auto-scaling based on the telemetry data.

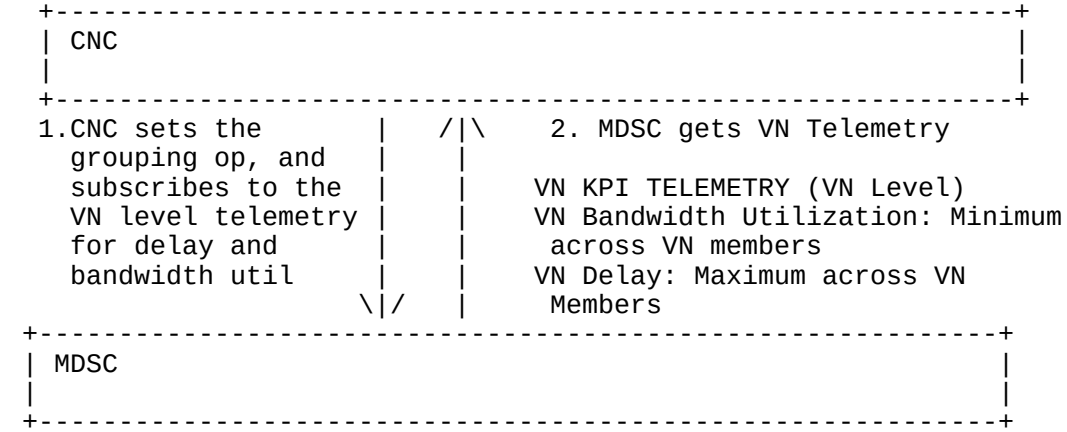
The TE KPI Telemetry Model augments the TE-Tunnel Model to enhance TE performance monitoring capability. This monitoring capability will facilitate proactive re-optimization and reconfiguration of TEs based on the performance monitoring data collected via the TE KPI Telemetry YANG model.



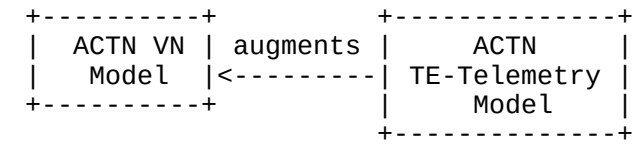
ACTN TE KPI Telemetry Model

This module describes performance telemetry for ACTN VN model. The telemetry data is augmented both at the VN Level as well as individual VN member level. This module also allows autonomic traffic engineering scaling intent configuration mechanism on the VN level. Scale in/out criteria might be used for network autonomics in order the controller to react to a certain set of variations in monitored parameters.

Moreover, this module also provides mechanism to define aggregated telemetry parameters as a grouping of underlying VN level telemetry parameters. Grouping operation (such as maximum, mean) could be set at the time of configuration. For example, if maximum grouping operation is used for delay at the VN level, the VN telemetry data is reported as the maximum {delay_vn_member_1, delay_vn_member_2, .. delay_vn_member_N}. Thus, this telemetry abstraction mechanism allows the grouping of a certain common set of telemetry values under a grouping operation. This can be done at the VN-member level to suggest how the E2E telemetry be inferred from the per domain tunnel created and monitored by PNCs. One proposed example is the following:



The ACTN VN TE-Telemetry Model augments the basic ACTN VN model to enhance VN monitoring capability. This monitoring capability will facilitate proactive re-optimization and reconfiguration of VNs based on the performance monitoring data collected via the ACTN VN Telemetry YANG model.



4. Notification

This model does not define specific notifications. To enable notifications, the mechanism defined in [I-D.ietf-netconf-yang-push] and [I-D.ietf-netconf-rfc5277bis] can be used. This mechanism currently allows the user to:

- Subscribe notifications on a per client basis.
- Specify subtree filters or xpath filters so that only interested contents will be sent.
- Specify either periodic or on-demand notifications.

YANG Push Subscription Examples

Below example shows the way for a client to subscribe for the telemetry information for a particular tunnel (Tunnel1). The telemetry parameter that the client is interested in is the utilized bandwidth.

```
<netconf:rpc netconf:message-id="101"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
    <establish-subscription
       xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
       <filter netconf:type="subtree">
          <te xmlns="urn:ietf:params:xml:ns:yang:ietf-te">
             <tunnels>
                <tunnel>
                  <name>Tunnel1</name>
                  <identifier/>
                  <state>
                    <te-telemetry
xmlns="urn:ietf:params:xml:ns:yang:ietf-te-kpi-telemetry">
```

```
                          <utilized-
   bandwidth/>

                       </te-telemetry>
                     </state>
                    </tunnel>
                 </tunnels>
              </te>
           </filter>
           <period>500</period>
           <encoding>encode-xml</encoding>
       </establish-subscription>
    </netconf:rpc>
```

This example shows the way for a client to subscribe for the telemetry information for all VNs. The telemetry parameter that the client is interested in is packet-loss and utilized bandwidth.

```
<netconf:rpc netconf:message-id="101"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
    <establish-subscription
       xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
       <filter netconf:type="subtree">
          <actn-state xmlns="urn:ietf:params:xml:ns:yang:ietf-actn-
vn">
             <vn>
                <vn-list>
                  <vn-id/>
                  <vn-name/>
                  <vn-
telemetry  xmlns="urn:ietf:params:xml:ns:yang:ietf-actn-te-kpi-
telemetry">
                       <one-way-packet-loss/>
                       <utilized-
bandwidth/>

                  </vn-telemetry >
                </vn-list>
             </vn>
          </actn-state>
       </filter>
       <period>500</period>
    </establish-subscription>
 </netconf:rpc>
```

5. YANG Data Tree

A graphical representation of the complete data tree is presented here. The meaning of the symbols in these diagrams is as follows and as per [I-D.ietf-netmod-rfc6087bis]. Each node is printed as: <status> <flags> <name> <opts> <type> <if-features>

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

<flags> is one of:
  rw for configuration data
  ro for non-configuration data
  -x for rpcs and actions
  -n for notifications

<name> is the name of the node
  (<name>) means that the node is a choice node
 :(<name>) means that the node is a case 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, choice, anydata or anyxml
  ! for a presence container
  * for a leaf-list or list
  [<keys>] for a list's keys

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

  If the type is a leafref, the type is printed as "-> TARGET",

  where TARGET is either the leafref path, with prefixed removed if possible.

<if-features> is the list of features this node depends on,

printed within curly brackets and a question mark "{...}?

```
module: ietf-te-kpi-telemetry
  augment /te:te/te:tunnels/te:tunnel/te:config:
    +--rw te-scaling-intent
       +--rw scale-in
       |  +--rw scale-in-operation-type?
       |  |       scaling-criteria-operation
       |  +--rw threshold-time?            uint32
       |  +--rw scale-in-condition* [performance-type]
       |     +--rw performance-type    identityref
       |     +--rw performance-data?   binary
       +--rw scale-down
          +--rw cooldown-time?              uint32
          +--rw scale-out-operation-type?
          |       scaling-criteria-operation
          +--rw scale-out-condition* [performance-type]
             +--rw performance-type    identityref
             +--rw performance-data?   binary
  augment /te:te/te:tunnels/te:tunnel/te:state:
    +--ro te-telemetry
       +--ro data
          +--ro one-way-delay?             uint32
          +--ro two-way-delay?             uint32
          +--ro one-way-delay-min?         uint32
          +--ro one-way-delay-max?         uint32
          +--ro two-way-delay-min?         uint32
          +--ro two-way-delay-max?         uint32
          +--ro one-way-delay-variation?   uint32
          +--ro two-way-delay-variation?   uint32
          +--ro utilized-bandwidth?        rt:bandwidth-ieee-float32
```

```
module: ietf-actn-te-kpi-telemetry
  augment /actn-vn:actn/actn-vn:vn/actn-vn:vn-list:
    +--rw vn-telemetry
    |  +--rw grouping-op
    |     +--rw delay-op?                grouping-operation
    |     +--rw delay-variation-op?      grouping-operation
    |     +--rw utilized-bandwidth-op?   grouping-operation
    +--rw vn-scaling-intent
       +--rw scale-in
       |  +--rw scale-in-operation-type?
       |  |       scaling-criteria-operation
       |  +--rw threshold-time?            uint32
```

```
      |  +--rw scale-in-condition* [performance-type]
      |     +--rw performance-type    identityref
      |     +--rw performance-data?   binary
      +--rw scale-down
         +--rw cooldown-time?              uint32
         +--rw scale-out-operation-type?
         |       scaling-criteria-operation
         +--rw scale-out-condition* [performance-type]
            +--rw performance-type    identityref
            +--rw performance-data?   binary
  augment /actn-vn:actn-state/actn-vn:vn/actn-vn:vn-list:
    +--ro vn-telemetry
    |  +--ro grouping-op
    |  |  +--ro delay-op?                identityref
    |  |  +--ro delay-variation-op?      identityref
    |  |  +--ro utilized-bandwidth-op?   identityref
    |  +--ro data
    |     +--ro one-way-delay?             uint32
    |     +--ro two-way-delay?             uint32
    |     +--ro one-way-delay-min?         uint32
    |     +--ro one-way-delay-max?         uint32
    |     +--ro two-way-delay-min?         uint32
    |     +--ro two-way-delay-max?         uint32
    |     +--ro one-way-delay-variation?   uint32
    |     +--ro two-way-delay-variation?   uint32
    |     +--ro utilized-bandwidth?        rt:bandwidth-ieee-float32
    +--ro vn-scaling-intent
       +--ro scale-in
       |  +--ro scale-in-operation-type?
       |  |       scaling-criteria-operation
       |  +--ro threshold-time?            uint32
       |  +--ro scale-in-condition* [performance-type]
       |     +--ro performance-type    identityref
       |     +--ro performance-data?   binary
       +--ro scale-down
          +--ro cooldown-time?              uint32
          +--ro scale-out-operation-type?
          |       scaling-criteria-operation
          +--ro scale-out-condition* [performance-type]
             +--ro performance-type    identityref
             +--ro performance-data?   binary
  augment /actn-vn:actn/actn-vn:vn/actn-vn:vn-list/actn-vn:vn-member-list:
    +--rw vn-telemetry
       +--rw grouping-op
          +--rw delay-op?                identityref
          +--rw delay-variation-op?      identityref
          +--rw utilized-bandwidth-op?   identityref
  augment /actn-vn:actn-state/actn-vn:vn/actn-vn:vn-list/actn-vn:vn-member-list:
    +--ro vn-telemetry
       +--ro grouping-op
```

```
       |  +--ro delay-op?                identityref
       |  +--ro delay-variation-op?      identityref
       |  +--ro utilized-bandwidth-op?   identityref
       +--ro data
          +--ro one-way-delay?             uint32
          +--ro two-way-delay?             uint32
          +--ro one-way-delay-min?         uint32
          +--ro one-way-delay-max?         uint32
          +--ro two-way-delay-min?         uint32
          +--ro two-way-delay-max?         uint32
          +--ro one-way-delay-variation?   uint32
          +--ro two-way-delay-variation?   uint32
          +--ro utilized-bandwidth?        rt:bandwidth-ieee-float32
```

6. Yang Data Model

ietf-te-kpi-telemetry model

The YANG code is as follows:

<CODE BEGINS> file "ietf-te-kpi-telemetry@2017-06-20.yang"

```
module ietf-te-kpi-telemetry {
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-kpi-telemetry";

    prefix "te-tel";

    import ietf-te {
        prefix "te";
    }

    import ietf-te-topology {
       prefix "tet";
    }

    organization
        "IETF Traffic Engineering Architecture and Signaling (TEAS)
        Working Group";

    contact
        "Editor: Young Lee <leeyoung@huawei.com>
         Editor: Dhruv Dhody <dhruv.ietf@gmail.com>
         Editor: Ricard Vilalta <ricard.vilalta@cttc.es>
         Editor: Satish Karunanithi <satish.karunanithi@gmail.com>";
```

```
    description
        "This module describes telemetry for teas tunnel model";

    revision 2017-06-20 {
        description
            "Initial revision. This YANG file defines
            the reusable base types for TE telemetry.";
        reference
            "Derived from earlier versions of base YANG files";
    }

    revision 2017-03-13 {
        description
            "Removed references to technology specific kpi.";
        reference
            "draft-lee-teas-actn-pm-telemetry-autonomics-00";
    }

    /*
     * Identities
     */

    identity telemetry-param-type {
        description
            "Base identity for telemetry param types";
    }

    identity one-way-delay {
        base telemetry-param-type;
        description
            "To specify average Delay in one (forward) direction";
    }
    identity two-way-delay {
        base telemetry-param-type;
        description
            "To specify average Delay in both (forward and reverse)
             directions";
    }

    identity one-way-delay-variation {
        base telemetry-param-type;
        description
            "To specify average Delay Variation in one (forward)
            direction";
    }

    identity two-way-delay-variation {
```

```
        base telemetry-param-type;
        description
            "To specify average Delay Variation in both (forward
            and reverse) directions";
    }

    identity utilized-bandwidth {
        base telemetry-param-type;
        description
            "To specify utilized bandwidth over the specified source
             and destination.";
    }

    /*
     * Enums
     */
    typedef scaling-criteria-operation {
        type enumeration {
            enum AND {
                description
                 "AND operation";
            }
            enum OR {
              description
               "OR operation";
            }
        }
        description
         "Operations to analize list of scaling criterias";
    }

    /*
     * Groupings
     */

    grouping telemetry-delay {
      description
      "Base telemetry delay parameters";

      leaf one-way-delay {
            type uint32;
            units "microseconds";
      description
          "To specify average Delay in one (forward) direction
                during the measurement interval";
      }
```

```
      leaf two-way-delay {
            type uint32;
            units "microseconds";
      description
          "To specify average Delay in both (forward and reverse)
               directions during the measurement interval";
      }

      leaf one-way-delay-min {
            type uint32;
            units "microseconds";
      description
          "To specify minimum Delay in one (forward) direction
           during the measurement interval";
      }
      leaf one-way-delay-max {
            type uint32;
            units "microseconds";
      description
          "To specify maximum Delay in one (forward) direction
           during the measurement interval";
      }

      leaf two-way-delay-min {
            type uint32;
            units "microseconds";
      description
          "To specify minimum Delay in both (forward and reverse)
               directions during the measurement interval";
      }

      leaf two-way-delay-max {
            type uint32;
            units "microseconds";
      description
          "To specify maximum Delay in both (forward and reverse)
               directions during the measurement interval";
      }

    }

    grouping telemetry-delay-variance {

    description
      "Base telemetry delay variance parameters";
    leaf one-way-delay-variation {
      type uint32;
```

```
      units "microseconds";
      description
        "To specify average Delay Variation in one (forward)
        direction during the measurement interval";
    }

    leaf two-way-delay-variation {
      type uint32;
      units "microseconds";
      description
        "To specify average Delay Variation in both
        (forward and reverse) directions during the
        measurement interval";
    }
  }

    grouping telemetry-bandwidth {
      description
          "Base telemetry bandwidth parameters";
        leaf utilized-bandwidth {
            type tet:te-bandwidth;
        description
          "To specify utilized bandwidth over the specified source
            and destination in bytes per seconds.";
            reference
                "RFC 3471";
        }
    }

    grouping scaling-criteria {
      description
            "Grouping for scaling criteria";
        leaf performance-type {
          type identityref {
              base telemetry-param-type;
            }
            description
              "Reference to the tunnel level telemetry type";
        }

        leaf performance-data {
          type binary;
          description
              "The encoding and meaning of this field is
              based on the performance-type";
        }
    }
```

```
    grouping scaling-intent {
      description
        "Basic scaling intent";

        container scale-in {
            description
            "Basic scaling-in intent";

            leaf scale-in-operation-type {
                type scaling-criteria-operation;
                default AND;
                description
                    "Operation to be applied to check between
                    scaling criterias to check if the scale in
                    threshold condition has been met.
                    Defaults to AND";
            }

            leaf threshold-time {
                type uint32;
                units "seconds";
                description
                    "The duration for which the criteria must
                    hold true";
            }

            list scale-in-condition {
              key "performance-type";
              description
                "Scaling conditions";
              uses scaling-criteria;
          }
        }
        container scale-down {
            description
            "Basic scaling-out intent";
            leaf cooldown-time {
                type uint32;
                units "seconds";
                description
                "The duration after a scaling-in/scaling-out action
                has been triggered, for which there will be no
                further operation";
            }
            leaf scale-out-operation-type {
                type scaling-criteria-operation;
```

```
                default OR;
                description
                    "Operation to be applied to check between
                    scaling criterias to check if the scale out
                    threshold condition has been met.
                    Defauls to OR";
            }
            list scale-out-condition {
              key "performance-type";
              description
                "Scaling conditions";
              uses scaling-criteria;
          }

        }

    }

    grouping telemetry-param {

        description
      "Base telemetry parameters";
    container data {
      description
        "The telemetry data";

          uses telemetry-delay;

          uses telemetry-delay-variance;

          uses telemetry-bandwidth;
      }

    }

    /*
     * Augments
     */
    augment "/te:te/te:tunnels/te:tunnel/te:config" {

        description
            "Augmentation parameters for config scaling-criteria
            TE tunnel topologies. Scale in/out criteria might be
            used for network autonomics in order the controller
            to react to a certain set of monitored params.";

        container te-scaling-intent {
```

```
          description
            "scaling intent";
            uses scaling-intent;

        }

    }

    augment "/te:te/te:tunnels/te:tunnel/te:state" {

        description
            "Augmentation parameters for state TE tunnel
            topologies.";

        container te-telemetry {
          description
            "telemetry params";
          uses telemetry-param;
        }
    }

}//module


<CODE ENDS>
```

ietf-actn-te-kpi-telemetry model

The YANG code is as follows:

```
<CODE BEGINS> file "ietf-actn-te-kpi-telemetry@2017-06-20.yang"

module ietf-actn-te-kpi-telemetry {

    namespace
      "urn:ietf:params:xml:ns:yang:ietf-actn-te-kpi-telemetry";

    prefix "actn-tel";

    import ietf-actn-vn {
        prefix "actn-vn";
    }

    import ietf-te-kpi-telemetry {
```

```
        prefix "te-kpi";
    }

    organization
        "IETF Traffic Engineering Architecture and Signaling (TEAS)
        Working Group";

    contact
        "Editor: Young Lee <leeyoung@huawei.com>
         Editor: Dhruv Dhody <dhruv.ietf@gmail.com>
         Editor: Ricard Vilalta <ricard.vilalta@cttc.es>
         Editor: Satish Karunanithi <satish.karunanithi@gmail.com>";

    description
        "This module describes telemetry for actn vn model";

    revision 2017-06-20 {
        description
            "Removed references to technology specific kpi.";
        reference
            "draft-lee-teas-actn-pm-telemetry-autonomics-01";
    }

    revision 2017-03-13 {
        description
            "Initial revision. This YANG file defines
            the ACTN VN telemetry.";
        reference
            "Derived from earlier versions of base YANG files";
    }

    /*
     * Identities
     */



    identity grouping-operation {
        description "Base identity for operations to analize list of monitored p
aram";
    }

    identity minimum-grouping-operation {
        base grouping-operation;
        description
            "Select the minimum param";
    }

    identity maximum-grouping-operation {
        base grouping-operation;
```

```
        description
            "Select the maximum param";
    }

    identity mean-grouping-operation {
        base grouping-operation;
        description
            "Select the MEAN of the params";
    }

    identity stddev-grouping-operation {
        base grouping-operation;
        description
            "Select the STD deviation of the params";
    }

    identity sum-grouping-operation {
        base grouping-operation;
        description
            "Select the sum of the params";
       reference
            "RFC 7823";
    }


    /*
     * Groupings
     */
    grouping vn-telemetry-param {
        description
            "telemetry-parameter for VN";
        uses te-kpi:telemetry-param;
    }
    grouping telemetry-grouping-op {
        description
            "Config how the VN telemetry should be applied";
        container grouping-op {
            description
                "The grouping operations";
            leaf delay-op {
                type identityref {
                  base grouping-operation;
                }
                default maximum-grouping-operation;
                description
                    "The operation that should be applied on the
                     VN-member telemetry to get the VN telemetry";
            }
```

```
            leaf delay-variation-op {
                type identityref {
                  base grouping-operation;
                }
                default maximum-grouping-operation;
                description
                    "The operation that should be applied on the
                     VN-member telemetry to get the VN telemetry";
            }

            leaf utilized-bandwidth-op {
                type identityref {
                  base grouping-operation;
                }
                default maximum-grouping-operation;
                description
                    "The operation that should be applied on the
                     VN-member telemetry to get the VN telemetry";
            }
        }

    }

    /*
     * Augments
     */
    augment "/actn-vn:actn/actn-vn:vn/actn-vn:vn-list" {

        description
            "Augmentation parameters for state TE VN topologies.";

        container vn-telemetry {
          description
            "VN telemetry configurations";
            uses telemetry-grouping-op;
        }
        container vn-scaling-intent {
            description
                "scaling intent";
            uses te-kpi:scaling-intent;
        }
    }
    augment "/actn-vn:actn-state/actn-vn:vn/actn-vn:vn-list" {

        description
            "Augmentation parameters for state TE VN topologies.";

        container vn-telemetry {
            description
```

```
                "VN telemetry params";
            uses telemetry-grouping-op;
            uses vn-telemetry-param;
        }
        container vn-scaling-intent {
            description
                "scaling intent";
            uses te-kpi:scaling-intent;
        }
    }

    /*
     * VN-member augment
     */
    augment "/actn-vn:actn/actn-vn:vn/actn-vn:vn-list/" +
            "actn-vn:vn-member-list" {
        description
            "Augmentation parameters for state TE vn member
            topologies.";
        container vn-telemetry {
            description
                "VN Member config";
            uses telemetry-grouping-op;
        }
    }
    augment "/actn-vn:actn-state/actn-vn:vn/actn-vn:vn-list/" +
            "actn-vn:vn-member-list" {
        description
            "Augmentation parameters for state TE vn member
            topologies.";
        container vn-telemetry {
          description
            "VN telemetry params";
            uses telemetry-grouping-op;
          uses vn-telemetry-param;
        }
    }
}
<CODE ENDS>
```

7. Security Considerations

The configuration, state, and action data defined in this document are designed to be accessed via a management protocol with a secure transport layer, such as NETCONF [RFC6241]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available

NETCONF protocol operations and content.

A number of configuration data nodes defined in this document are writable/deletable (i.e., "config true") These data nodes may be considered sensitive or vulnerable in some network environments.

8. IANA Considerations

TDB

9. Acknowledgements

10. References

Informative References

[RFC4110] R. Callon and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)", RFC 4110, July 2005.

[RFC6020] M. Bjorklund, Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[Service-YANG] Q. Wu, W. Liu and A. Farrel, "Service Models Explained", draft-wu-opsawg-service-model-explained, work in progress.

[Netmod-Yang-Model-Classification] D. Bogdanovic, B. Claise, and C. Moberg, "YANG Module Classification", draft-ietf-netmod-yang-model-classification, work in progress.

[Netconf] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241.

[Restconf] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf, work in progress.

Normative References

[ACTN-Frame] D. Cecarelli and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks", draft-ietf-teas-actn-framework, work in progress.

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

[TE-Tunnel] T. Saad (Editor), "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te, work in progress.

[ACTN-VN-YANG] Y. Lee (Editor), "A Yang Data Model for ACTN VN Operation", draft-lee-teas-actn-vn-yang, work in progress.

[L3SM-YANG] S. Litkowski, L.Tomotaki, and K. Ogaki, "YANG Data Model for L3VPN service delivery", draft-ietf-l3sm-l3vpn-service-model, work in progress.

[PCEP-Service-Aware] D. Dhody, et al., "Extensions to the Path Computation Element Communication Protocol (PCEP) to compute service aware Label Switched Path (LSP)", draft-ietf-pce-pcep-service-aware, work in progress.

[ACTN-PERF] 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-03, work in progress.

11. Contributors

Authors' Addresses

Young Lee
Huawei Technologies
5340 Legacy Drive Suite 173
Plano, TX 75024, USA

Email: leeyoung@huawei.com

Dhruv Dhody
Huawei Technology
Leela Palace
Bangalore, Karnataka 560008
India

Email: dhruv.dhody@huawei.com

Satish Karunanithi
Huawei Technology
Leela Palace
Bangalore, Karnataka 560008
India

Email: satish.karunanithi@gmail.com

Ricard Vilalta
Centre Tecnologic de Telecomunicacions de Catalunya (CTTC/CERCA)
Av. Carl Friedrich Gauss 7
08860 - Castelldefels
Barcelona (Spain)
Email: ricard.vilalta@cttc.es

Daniel King
Lancaster University

Email: d.king@lancaster.ac.uk

Daniele Ceccarelli
Ericsson
Torshamnsgatan,48
Stockholm, Sweden

Email: daniele.ceccarelli@ericsson.com

TEAS Working Group
Internet Draft
Intended Status: Standard Track

Y. Lee (Editor)
Dhruv Dhody
Huawei
D. Ceccarelli
Ericsson
Takuya Miyasaka
KDDI
Peter Park
KT
Bin Young Yoon
ETRI

Expires: September 9, 2017 March 9, 2017

A Yang Data Model for ACTN VN Operation

draft-lee-teas-actn-vn-yang-03

Abstract

This document provides a YANG data model for the Abstraction and Control of Traffic Engineered (TE) networks (ACTN) Virtual Network (VN) operation.

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire on September 9, 2017.

Table of Contents

1. Introduction

This document provides a YANG data model for the Abstraction and Control of Traffic Engineered (TE) networks (ACTN) Virtual Network (VN) operation that is going to be implemented for the Customer Network Controller (CNC)- Multi-Domain Service Coordinator (MSDC) interface (CMI).

The YANG model on the CMI is also known as customer service model in [Service-YANG]. The YANG model discussed in this document is used to operate customer-driven VNs during the VN computation, VN instantiation and its life-cycle operations stages.

Note that the YANG model presented in this draft has two aspects:

- VN pre-instantiation mode of operation (also known as VN compute);
- VN instantiation mode of operation.

The VN pre-instantiation mode of operation is concerned about service inquiry before making a formal request for VN instantiation. This operation is important for a customer to make sure the network can provide VN services it desires.

The VN instantiation mode of operation is concerned about instantiating VNs. In the VN instantiation mode, the CNC provides the VN service definition that includes VN members, VN service objective, VN service policy and preferences, etc. Upon receipt of a VN instantiation request, the MDSC (in coordination with PNCs) executes service request into network operation that include creating tunnels/paths and securing network resources/slices for VNs.

The YANG model discussed in this document basically provides the characteristics of VNs such as VN level parameters (e.g., VN ID, VN member, VN objective function, VN service preference, etc.), customer’s end point characteristics (e.g., Customer Interface Capability, Access Points Interface characteristics, etc.), and other relevant VN information that needs to be known to the MDSC to facilitate ACTN VN operation.

1.1. Terminology

Refer to [ACTN-Frame] and [RFC7926] for the key terms used in this document.

2. ACTN CMI context

The model presented in this document has the following ACTN context.

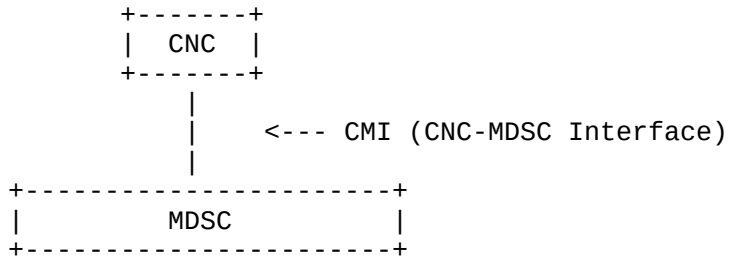


Figure 1. ACTN CMI

As defined in [ACTN-FW], a Virtual Network is a client view (typically a network slice) of the transport network. It can be presented by the provider as a set of physical and/or abstracted resources. Depending on the agreement between client and provider various VN operations and VN views are possible.

a) VN can be seen as an (or set of) e2e tunnel(s) from a customer point of view where an e2e tunnel is referred as a VN member. Each VN member (i.e., e2e tunnel) can then be formed by recursive aggregation of lower level paths at a provider level. Such end to end tunnels may comprise of customer end points, access links, intra domain paths and inter-domain link. In this view VN is thus a list of VN members.

b) VN can also be seen as a terms of topology comprising of physical and abstracted nodes and links. The nodes in this case include physical customer end points, border nodes, and internal nodes as well as abstracted nodes. Similarly the links includes physical access, inter-domain and intra-domain links as well as abstracted links. The abstracted nodes and links in this view can be pre-negotiated or created dynamically.

For both cases, the CNC can dynamically add VN elements. For case 1, the VN element is an end-to-end tunnel and for case 2, the VN element can be virtual nodes or virtual links.

In the subsequent discussion, the first form of VN will be discussed.

The following figure describes a VN that comprises three VN members forming a full mesh for the VN as an illustration.

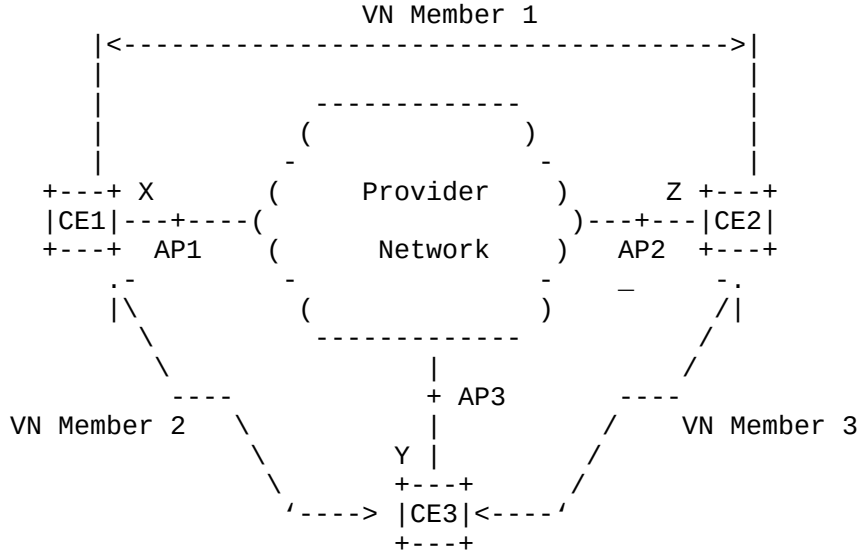


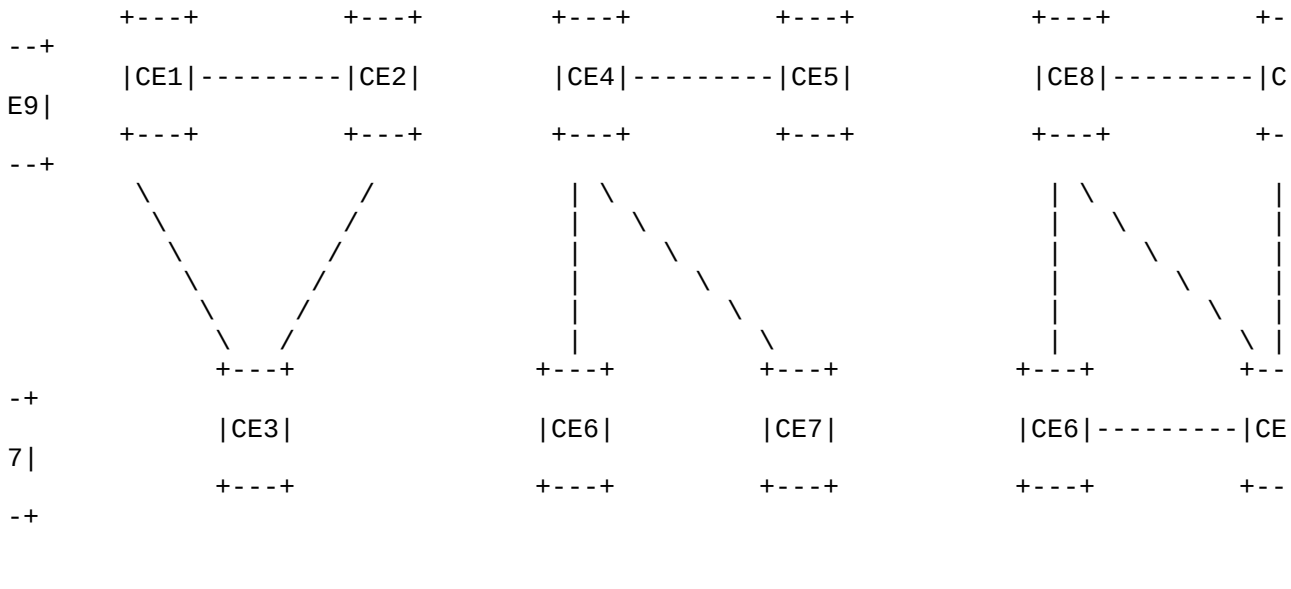
Figure 2. Full Mesh Example for a VN

In Figure 2, a VN has three members, namely, VN Member 1, VN member 2, and VN member 3. VN Member 1 is an end-to-end tunnel identified by CE1-AP1 (source) and CE2-AP2 (destination). Similarly, VN Member 2 by CE1-AP1 and CE3-AP3 and VN Member 3 by CE3-AP3 and CE2-AP2. This particular VN shown in Figure 2 is a full mesh connectivity across these three customer end-points.

The set of assumptions that applies to this document is the following:

- CNC is responsible for providing necessary Customer End-Points information to the MDSC via the CMI.
- The access links (between Customer Edge (CE) Devices and the Provider Edge (PE) Devices) are assumed to have been provisioned prior to the VN instantiation request. Access point identifiers have been configured and therefore are known in both the CNC and the MDSC.

It is also possible for the customer to create a VN which can be a hub and spoke or any other form of connectivity depending on its connectivity requirement. Each end-to-end tunnel may be unidirectional or bidirectional which is also depending on its connectivity requirements. The following figure shows some examples of a VN that can be represented in a different connectivity form depending on the customer's connectivity requirements.



(a) Full Mesh (b) Hub and Spoke (c) partial Mesh

Figure 3. Different Connectivity Forms of a VN

It is important to note that a VN can associate a multiple number of end-to-end tunnels (i.e., VN members) with one unique identifier. From a customer standpoint, this simplifies its VN operation significantly.

The MDSC interacts with the CNC for the VN operation. Once the customer VN is requested by the CNC to the MDSC, the MDSC shall be responsible for translating and mapping the VN request into specific network centric-models (e.g., TE-tunnels [TE-Tunnel], TE-topology [TE-TOPO], etc.) to coordinate the multi-domain network operations with PNCs.

2.1. Justification of the ACTN VN Model on the CMI.

1. It binds multiple e2e tunnels as a VN. VN is a unit for concurrency. TE-tunnel model deals each VN member as a separate entity, so it loses concurrent allocation of TE resources. TE-tunnel model is a sequential provisioning approach.

2. It is easier for some customers to work on VN level/Network slicing rather than individual TE tunnels.

3. ACTN VN supports multi-source and multi-destination which TE-tunnel model does not support. (See Section 2.2 for details)

4. ACTN VN supports VN compute (pre-instantiation mode) which TE-tunnel model does not.

5. There are certain advantages to keep a set of TE-tunnels as one VN unit for applying policy, reroute, protection, restoration, etc. rather than treating each TE-tunnel as individual unit.

2.2. Multi-sources and multi-destinations

In creating a virtual network, the list of sources or destinations or both may not be pre-determined by the customer. For instance, for a given source, there may be a list of multiple-destinations to

which the optimal destination may be chosen depending on the network resource situations. Likewise, for a given destination, there may also be multiple-sources from which the optimal source may be chosen. In some cases, there may be a pool of multiple sources and destinations from which the optimal source-destination may be chosen. The following YANG module is shown for describing source container and destination container. See details in Section 4.

```
   +--rw actn
          |  +--rw vn
          |     +--rw vn-list* [vn-id]
          |           ...
          |        |  +--rw src
          |        |  |  +--rw src?            -> /actn/ap/access-point-list/acce
ss-
   point-id
          |        |  |  +--rw src-vn-ap-id?   uint32
          |        |  |  +--rw multi-src?      boolean
          |        |  +--rw dest
          |        |     +--rw dest?            -> /actn/ap/access-point-list/acc
ess-
   point-id
          |        |     +--rw dest-vn-ap-id?   uint32
          |        |     +--rw multi-dest?      boolean

         +--ro actn-state
            +--ro vn
               +--ro vn-list* [vn-id]
                  |  +--ro src
                  |  |  +--ro src?            -> /actn/ap/access-point-list/acce
ss-
   point-id
                  |  |  +--ro src-vn-ap-id?   uint32
                  |  |  +--ro multi-src?      boolean
                  |  +--ro dest
                  |  |  +--ro dest?            -> /actn/ap/access-point-list/acc
ess-
   point-id
                  |  |  +--ro dest-vn-ap-id?   uint32
                  |  |  +--ro multi-dest?      boolean
                   ...
                  +--ro multi-src-dest
                  |  +--ro selected-vn-member?   -> /actn-state/vn/vn-list/vn-id
```

3. ACTN VN YANG Model (Tree Structure)

```
module: ietf-actn-vn
    +--rw actn
    |  +--rw ap
    |  |  +--rw access-point-list* [access-point-id]
    |  |     +--rw access-point-id      uint32
    |  |     +--rw access-point-name?   string
    |  |     +--rw src-tp-id?           binary
    |  |     +--rw dst-tp-id?           binary
    |  |     +--rw max-bandwidth?       tet:te-bandwidth
```

```
    |  |     +--rw avl-bandwidth?       tet:te-bandwidth
    |  +--rw vn
    |     +--rw vn-list* [vn-id]
    |        +--rw vn-id                 uint32
    |        +--rw vn-name?              string
    |        +--rw vn-member-list* [vn-member-id]
    |        |  +--rw vn-member-id    uint32
    |        |  +--rw src
    |        |  |  +--rw src?            -> /actn/ap/access-point-list/access-po
int-id
    |        |  |  +--rw src-vn-ap-id?   uint32
    |        |  |  +--rw multi-src?      boolean
    |        |  +--rw dest
    |        |     +--rw dest?            -> /actn/ap/access-point-list/access-p
oint-
id
    |        |     +--rw dest-vn-ap-id?   uint32
    |        |     +--rw multi-dest?      boolean
    |        +--rw objective-function?   pcep:objective-function
    |        +--rw metric* [metric-type]
    |        |  +--rw metric-type    identityref
    |        |  +--rw limit
    |        |  |  +--rw enabled?   boolean
    |        |  |  +--rw value?     uint32
    |        |  +--rw optimize
    |        |     +--rw enabled?   boolean
    |        |     +--rw value?     uint32
    |        +--rw bandwidth?            tet:te-bandwidth
    |        +--rw protection?           identityref
    |        +--rw local-reroute?        boolean
    |        +--rw push-allowed?         boolean
    |        +--rw incremental-update?   boolean
    |        +--rw admin-status?         identityref
    +--ro actn-state
       +--ro ap
       |  +--ro access-point-list* [access-point-id]
       |     +--ro access-point-id      uint32
       |     +--ro access-point-name?   string
       |     +--ro src-tp-id?           binary
       |     +--ro dst-tp-id?           binary
       |     +--ro max-bandwidth?       tet:te-bandwidth
       |     +--ro avl-bandwidth?       tet:te-bandwidth
       +--ro vn
          +--ro vn-list* [vn-id]
             +--ro vn-id                 uint32
             +--ro vn-name?              string
             +--ro vn-member-list* [vn-member-id]
             |  +--ro vn-member-id    uint32
             |  +--ro src
             |  |  +--ro src?            -> /actn/ap/access-point-list/access-po
int-id
             |  |  +--ro src-vn-ap-id?   uint32
```

```
            |  |  +--ro multi-src?      boolean
            |  +--ro dest
            |  |  +--ro dest?            -> /actn/ap/access-point-list/access-p
oint-
id
            |  |  +--ro dest-vn-ap-id?   uint32
            |  |  +--ro multi-dest?      boolean
            |  +--ro metric* [metric-type]
            |  |  +--ro metric-type    identityref
            |  |  +--ro limit
            |  |  |  +--ro enabled?   boolean
            |  |  |  +--ro value?     uint32
            |  |  +--ro optimize
            |  |     +--ro enabled?   boolean
            |  |     +--ro value?     uint32
            |  +--ro oper-status?    identityref
            |  +--ro tunnel-ref?     te:tunnel-ref
            +--ro multi-src-dest
            |  +--ro selected-vn-member?   -> /actn-state/vn/vn-list/vn-id
            +--ro vn-topology-ref
            |  +--ro network-ref?   -> /nw:networks/network/network-id
            +--ro objective-function?   pcep:objective-function
            +--ro metric* [metric-type]
            |  +--ro metric-type    identityref
            |  +--ro limit
            |  |  +--ro enabled?   boolean
            |  |  +--ro value?     uint32
            |  +--ro optimize
            |     +--ro enabled?   boolean
            |     +--ro value?     uint32
            +--ro bandwidth?            tet:te-bandwidth
            +--ro protection?           identityref
            +--ro local-reroute?        boolean
            +--ro push-allowed?         boolean
            +--ro incremental-update?   boolean
            +--ro admin-status?         identityref
            +--ro oper-status?          identityref

  rpcs:
    +---x vn-compute
       +---w input
       |  +---w vn-member-list* [vn-member-id]
       |  |  +---w vn-member-id    uint32
       |  |  +---w src
       |  |  |  +---w src?            -> /actn/ap/access-point-list/access-point
-id
       |  |  |  +---w src-vn-ap-id?   uint32
       |  |  |  +---w multi-src?      boolean
       |  |  +---w dest
       |  |     +---w dest?            -> /actn/ap/access-point-list/access-poin
t-id
       |  |     +---w dest-vn-ap-id?   uint32
```

```
       |  |     +---w multi-dest?      boolean
       |  +---w objective-function?   pcep:objective-function
       |  +---w metric* [metric-type]
       |  |  +---w metric-type    identityref
       |  |  +---w limit
       |  |  |  +---w enabled?   boolean
       |  |  |  +---w value?     uint32
       |  |  +---w optimize
       |  |     +---w enabled?   boolean
       |  |     +---w value?     uint32
       |  +---w bandwidth?            tet:te-bandwidth
       |  +---w protection?           identityref
       |  +---w local-reroute?        boolean
       |  +---w push-allowed?         boolean
       |  +---w incremental-update?   boolean
       +--ro output
          +--ro vn-member-list* [vn-member-id]
          |  +--ro vn-member-id    uint32
          |  +--ro src
          |  |  +--ro src?            -> /actn/ap/access-point-list/access-point
-id
          |  |  +--ro src-vn-ap-id?   uint32
          |  |  +--ro multi-src?      boolean
          |  +--ro dest
          |  |  +--ro dest?            -> /actn/ap/access-point-list/access-poin
t-id
          |  |  +--ro dest-vn-ap-id?   uint32
          |  |  +--ro multi-dest?      boolean
          |  +--ro metric* [metric-type]
          |  |  +--ro metric-type    identityref
          |  |  +--ro limit
          |  |  |  +--ro enabled?   boolean
          |  |  |  +--ro value?     uint32
          |  |  +--ro optimize
          |  |     +--ro enabled?   boolean
          |  |     +--ro value?     uint32
          |  +--ro oper-status?    identityref
          +--ro multi-src-dest
             +--ro selected-vn-member-id?   uint32
```

4. ACTN-VN YANG Code

The YANG code is as follows:

<CODE BEGINS> file "ietf-actn-vn@2017-03-09.yang"

```
module ietf-actn-vn {

       namespace "urn:ietf:params:xml:ns:yang:ietf-actn-vn";
```

```
        prefix "vn";

        /* Import TE generic types */
        import ietf-te-types {
            prefix "te-types";
        }

        import ietf-te-topology {
            prefix "tet";
        }

        import ietf-te {
            prefix "te";
        }

        import ietf-pcep {
           prefix "pcep";
        }

        organization
            "IETF Traffic Engineering Architecture and Signaling (TEAS)
            Working Group";

        contact
            "Editor: Young Lee <leeyoung@huawei.com>
             Editor: Dhruv Dhody <dhruv.ietf@gmail.com>";

        description
            "This module contains a YANG module for the ACTN VN. It
            describes a VN operation module that takes place in the
            context of the CNC-MDSC Interface (CMI) of the ACTN
            architecture where the CNC is the actor of a VN Instantiation
            /modification /deletion.";

        revision 2017-03-09 {
            description
                "initial version.";
            reference
                "TBD";
        }

        identity path-metric-delay {
           base te-types:path-metric-type;
           description
```

```
            "delay  path metric";
          }

          identity path-metric-delay-variation {
            base te-types:path-metric-type;
            description
              "delay-variation path metric";
          }

          identity path-metric-loss {
            base te-types:path-metric-type;
            description
              "loss path metric";
          }

         /*
          * Groupings
          */

         grouping access-point{
             description
                 "AP related information";
             leaf access-point-id {
                 type uint32;
                 description
                     "unique identifier for the referred
                     access point";
             }
             leaf access-point-name {
                 type string;
                 description
                     "ap name";
             }
             /*using direct tp-id for now as per ietf-te
               should check if reference is better*/
             leaf src-tp-id {
              type binary;
              description
                "TE tunnel source termination point identifier.";
           }
           leaf dst-tp-id {
              type binary;
              description
                "TE tunnel destination termination point identifier.";
           }
```

```
            leaf max-bandwidth {
                type tet:te-bandwidth;
                description
                    "max bandwidth of the AP";
            }
            leaf avl-bandwidth {
                type tet:te-bandwidth;
                description
                    "available bandwidth of the AP";
            }
            /*add details and any other properties of AP,
            not associated by a VN
            CE port, PE port etc.

            */
        }//access-point

        grouping vn-member {
            description
                "vn-member is described by this container";
            leaf vn-member-id {
                type uint32;
                description
                    "vn-member identifier";
            }
            container src
            {
                description
                    "the source of VN Member";
                leaf src {
                    type leafref {
                        path "/actn/ap/access-point-list/access-point-id";
                    }
                    description
                        "reference to source AP";
                }
                leaf src-vn-ap-id{
                    type uint32;
                    description

                        "vn-ap-id";
                }
                leaf multi-src {
                    type boolean;
                    description
```

```
                        "Is source part of multi-source, where
                        only one of the source is enabled";
                }
            }
            container dest
            {
                description
                    "the destination of VN Member";
                leaf dest {
                    type leafref {
                        path "/actn/ap/access-point-list/access-point-id";
                    }
                    description
                        "reference to destination AP";
                }
                leaf dest-vn-ap-id{
                    type uint32;
                    description
                        "vn-ap-id";
                }
                leaf multi-dest {
                    type boolean;
                    description
                        "Is destination part of multi-destination, where
                        only one of the destination is enabled";
                }
            }
        }//vn-member

        grouping policy {
            description
                "policy related to vn-member-id";
            leaf local-reroute {
                type boolean;
                description
                    "Policy to state if reroute
                    can be done locally";
            }
            leaf push-allowed {
                type boolean;
                description
                    "Policy to state if changes
                    can be pushed to the customer";
            }
            leaf incremental-update {
```

```
                type boolean;
                description
                    "Policy to allow only the
                    changes to be reported";
            }
        }//policy

        grouping metrics {
            description
                "metric related information";
            list metric{
                key "metric-type";
                description
                    "The list of metrics for VN";
                leaf metric-type {
                    type identityref {
                        base te-types:path-metric-type;
                    }
                    description
                        "The VN metric type.";
                }
                container limit {
                    description
                        "Limiting contraints";
                    leaf enabled{
                        type boolean;
                        description
                            "Limit contraint is enabled";
                    }
                    leaf value{
                        type uint32;
                        description
                            "The limit value";
                    }
                }
                container optimize{
                    description
                        "optimizing constraints";
                    leaf enabled{
                        type boolean;
                        description
                            "Metric to optimize";
                     }
                     leaf value{
                         type uint32;
```

```
                        description
                           "The computed value";
                    }
                }
            }

        }

        grouping service-metric {
            description
                "service-metric";
            leaf objective-function {
                type pcep:objective-function;
                    description
                        "operational state of the objective function";
            }

            uses metrics;

            leaf bandwidth {
                type tet:te-bandwidth;
                description
                    "bandwidth requested/required for
                    vn-member-id";
            }

            leaf protection {
                type identityref {
                    base te-types:lsp-prot-type;
                }
                description "protection type.";
            }

            uses policy;

        }//service-metric

        /*
         * Configuration data nodes
         */
        container actn {
            description
                "actn is described by this container";
            container ap {
                description
```

```
                "AP configurations";
             list access-point-list {
                key "access-point-id";
                description
                    "access-point identifier";
                uses access-point{
                    description
                        "access-point information";
                }
             }
            }
            container vn {
                description
                    "VN configurations";

                list vn-list {
                    key "vn-id";
                    description
                        "a virtual network is identified by a vn-id";
                    leaf vn-id {
                        type uint32;
                        description
                            "a unique vn identifier";
                    }
                    leaf vn-name {
                        type string;
                        description "vn name";
                    }
                    list vn-member-list{
                        key "vn-member-id";
                        description
                            "List of VN-members in a VN";
                        uses vn-member;
                    }
                    uses service-metric;

                    leaf admin-status {
                        type identityref {
                            base te-types:state-type;
                        }
                        default te-types:state-up;
                        description "VN administrative state.";
                    }

                }//vn-list
```

```
            }//vn
          }//actn

          /*
           * Operational data nodes
           */

          container actn-state{
              config false;

              description
                  "actn is described by this container";

              container ap {
                  description
                      "AP state";
                  list access-point-list {
                      key "access-point-id";
                      description
                          "access-point identifier";
                      uses access-point{
                          description
                              "access-point information";
                      }
               }
              }
              container vn {
                  description
                      "VN state";
                  list vn-list {
                      key "vn-id";
                      description
                          "a virtual network is identified by a vn-id";
                      leaf vn-id {
                          type uint32;
                          description
                              "a unique vn identifier";
                      }
                      leaf vn-name {
                          type string;
                          description "vn name";
                      }
                      list vn-member-list{
                          key "vn-member-id";
                          description
```

```
                "List of VN-members in a VN";
            uses vn-member;
            uses metrics;
            leaf oper-status {
                type identityref {
                    base te-types:state-type;
                }
                description

                    "VN-member operational state.";
            }
            leaf tunnel-ref {
                type te:tunnel-ref;
                description
                    "A reference to the TE tunnel
                    in the TE model";
            }
        }
        container multi-src-dest{
            description
                "The selected VN Member when multi-src
                and/or mult-destination is enabled.";
            leaf selected-vn-member{
                type leafref {
                    path "/actn-state/vn/vn-list/vn-id";
                }
                description
                    "The selected VN Member along the set
                    of source and destination configured
                    with multi-source and/or multi-destination";
            }
        }
        container vn-topology-ref{
            description
                "An optional reference to the TE Topology
                 Model where the abstract nodes and links
                 of the Topology can be found";
             uses tet:te-topology-ref;
        }

        uses service-metric;

        leaf admin-status {
            type identityref {
                base te-types:state-type;
```

```
                        }
                        description "VN administrative state.";
                    }
                    leaf oper-status {
                        type identityref {
                            base te-types:state-type;
                        }
                        description "VN operational state.";
                    }

                }//vn-list
            }//vn
        }//actn-state
        /*
        * Notifications - TBD
        */
        /*
        * RPC
        */
        rpc  vn-compute{
            description
                "The VN computation without actual
                instantiation";
            input {
                list vn-member-list{
                    key "vn-member-id";
                    description
                        "List of VN-members in a VN";
                    uses vn-member;
                }
                uses service-metric;
            }
            output {
                list vn-member-list{
                    key "vn-member-id";
                    description
                        "List of VN-members in a VN";
                    uses vn-member;
                    uses metrics;
                    leaf oper-status {
                        type identityref {
                            base te-types:state-type;
                        }
                        description
                            "VN-member operational state.";
```

```
                    }
                }
                container multi-src-dest{
                    description
                        "The selected VN Member when multi-src
                        and/or mult-destination is enabled.";
                    leaf selected-vn-member-id{
                        type uint32;
                        description
                            "The selected VN Member-id from the
                            input";
                    }
                }
            }
        }
    }
```

<CODE ENDS>

5. Security Considerations

TDB

6. IANA Considerations

TDB

7. Acknowledgments

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

8. References

8.1. Normative References

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

[TE-tunnel] T. Saad, et al., "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", work in progress: draft-ietf-teas-yang-te.

[Service-YANG] Q. Wu, W. Liu and A. Farrel, "Service Models Explained", draft-wu-opsawg-service-model-explained, work in progress.

8.2. Informative References

[RFC7926] A. Farrel (Ed.), "Problem Statement and Architecture for Information Exchange between Interconnected Traffic-Engineered Networks", RFC 7926, July 2016.

[ACTN-REQ] Lee, et al., "Requirements for Abstraction and Control of TE Networks", work in progress: draft-ietf-teas-actn-requirements.

[ACTN-FWK] D. Ceccarelli, Y. Lee [Editors], "Framework for Abstraction and Control of Traffic Engineered Networks", work in progress: draft-ceccarelli-teas-actn-framework.

9. Contributors

Contributor’s Addresses

Haomian Zheng
Huawei Technologies
Email: zhenghaomian@huawei.com

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

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

Authors' Addresses

Young Lee (ed.)
Huawei Technologies
Email: leeyoung@huawei.com

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

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

Takuya Miyasaka
KDDI
Email: ta-miyasaka@kddi.com

Peter Park
KT
Email: peter.park@kt.com

Bin Yeong Yoon
ETRI
Email: byyun@etri.re.kr

TEAS WG

Young Lee
Dhruv Dhody
Huawei

Internet Draft
Intended status: standard

Daniele Ceccarelli
Ericsson

Expires: September 2017

-

March 9, 2017

Traffic Engineering and Service Mapping Yang Model
draft-lee-teas-te-service-mapping-yang-00

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire on September 9, 2017.

Copyright Notice

Abstract

This document provides a YANG data model to map service model (e.g. L3SM) and Traffic Engineering model (e.g. TE Tunnel or ACTN VN model). This model is referred to as TE service Mapping Model. This model is applicable to the operation's need for a seamless control and management of their L3VPN with TE tunnel support.

Table of Contents

1. Introduction

Data models are a representation of objects that can be configured or monitored within a system. Within the IETF, YANG [RFC6020] is the language of choice for documenting data models, and YANG models have been produced to allow configuration or modeling of a variety of network devices, protocol instances, and network services. YANG data models have been classified in [Netmod-Yang-Model-Classification] and [Service-YANG].

[RFC4110] provides a framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs). [L3SM-YANG] provides a L3VPN service delivery YANG model for PE-based VPNs.

[ACTN-VN-YANG] describes how customers or end to end orchestrators can request and/or instantiate a generic virtual network service. [ACTN-Applicability] describes a connection between IETF YANG model classifications to ACTN interfaces. In particular, it describes the customer service model can be mapped into the CMI (CNC-MDSC Interface) of the ACTN architecture.

The YANG model on the ACTN CMI is known as customer service model in [Service-YANG]. The YANG model developed in this document describes how operator's end to end orchestrator interacts with the MDSC so that the MDSC then can coordinate the control and management of L3VPN MPLS TE tunnels that traverse both IP/MPLS and Transport networks.

While IP/MPLS PNC is responsible for provisioning the VPN service on the PE nodes, the MDSC can coordinate how to map the VPN services with TE tunnels. This is consistent with the two of the core functions of the MDSC specified in [ACTN-Frame]:

. Customer mapping/translation function: This function is to map customer requests/commands into network provisioning requests that can be sent to the Physical Network Controller (PNC) according to business policies provisioned statically or dynamically. Specifically, it provides mapping and translation of a customer's service request into a set of parameters that are specific to a network type and technology such that network configuration process is made possible.

. Virtual service coordination function: This function translates customer service-related information into virtual network service operations in order to seamlessly operate virtual networks while meeting a customer's service requirements. In the context of ACTN, service/virtual service coordination includes a number of service orchestration functions such as multi-destination load balancing, guarantees of service quality, bandwidth and throughput. It also includes notifications for service fault and performance degradation and so forth.

In some cases, under the confines of service policy, dynamic TE tunnel creation may need to be supported for the VPN service. This may occur when there are no suitable existing TE tunnels that can support VPN service requirements. Or the operator would like to dynamically create and bind tunnels to the VPN, which could not be shared for network slicing.

To summarize there are two mode of operations -

. VN/Tunnel Binding - Customer could use the VPN service model [L3SM-Yang] to communicate to the network operator to deliver a L3VPN service. Based on the sites, QoS parameters, VPN service topology, the network operator could create an ACTN VN [ACTN-VN-YANG]. Further the mapping yang model described in Section 5 of this document is used to set this mapping between the L3VPN service and the ACTN VN. Note that this could be done dynamically. The VN (and TE tunnels) could be bound to the L3VPN and not used for any other VPN.

. VN/Tunnel Selection - Customer could request an L3VPN service [L3SM-Yang], and with this model as input, the PNC configures the different network elements to deliver the service. Each network element would select a tunnel based on the configuration. With this mode, new tunnels (or VN) are not created for each VPN. Thus, the tunnels can be shared across multiple VPN. Further the mapping yang model described in Section 5 of this document is used to get the mapping between the L3VPN and the tunnels in use.

The YANG model described in this document provides an ACTN TE-service mapping model that enables a seamless service mapping across L3VPN, ACTN VN and TE-tunnel models at the controllers.

2. L3VPN Architecture in ACTN context

Figure 1 shows the architectural context of this document.

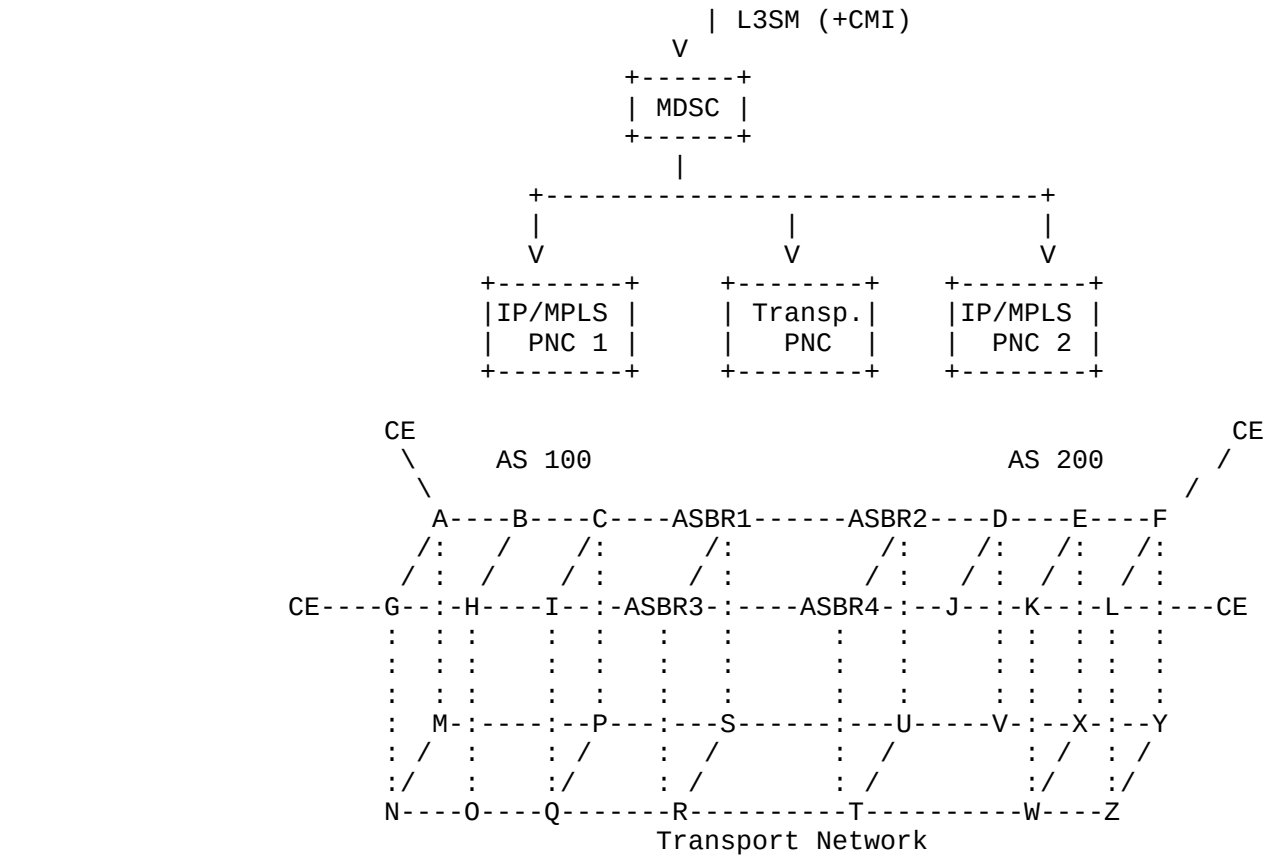


Figure 1: L3VPN Architecture from the IP+Optical Network Perspective

There are three main entities in the architecture.

. MDSC: This entity is responsible for coordinating a L3VPN service request (expressed in L3SM) with the IP PNC and the Transport PNC. One of the key responsibilities of the MDSC for TE services is to coordinate with both the IP PNC and the Transport PNC for the mapping of L3VPN Service Model and ACTN VN/TE-Tunnel models. With the VN/TE-tunnel binding case, the MDSC will need to coordinate with the Transport PNC to dynamically create the TE-tunnel(s) in the Transport network as needed. These tunnels are added as links in the IP Layer topology. The MDSC coordinates with IP PNC to

create the TE-tunnel(s) in the IP layer, as part of the ACTN VN creation.

. IP/MPLS PNC: This entity is responsible for device configuration to create PE-PE L3VPN tunnels for the VPN customer and for the configuration of the L3VPN VRF on the PE nodes. Each network element would select a tunnel based on the configuration.

. Transport PNC: This entity is responsible for device configuration for TE tunnels in the transport networks.

High-Level Control Flows

1. Customer asks for a L3VPN between CE1 and CE2 with TE constraints using L3SM model. The customer can provide tunnel creation policy where it allows dynamic VN/TE tunnel creation or not. Under this policy, dynamic VN/TE tunnels can be created when there are no proper VN/TE-tunnels that can support L3VPN tunnels or when there is a strict isolation requirement for the VPN service, e.g., no sharing with other tunnels is allowed.

2. The MDSC determines if it needs to create a new VN, and if that is the case, ACTN VN YANG [ACTN-VN-YANG] is used to configure a new VN based on this VPN and map the VPN service to ACTN VN. In case an existing tunnel is to be used, each device will select which tunnel to use and populates this mapping information.

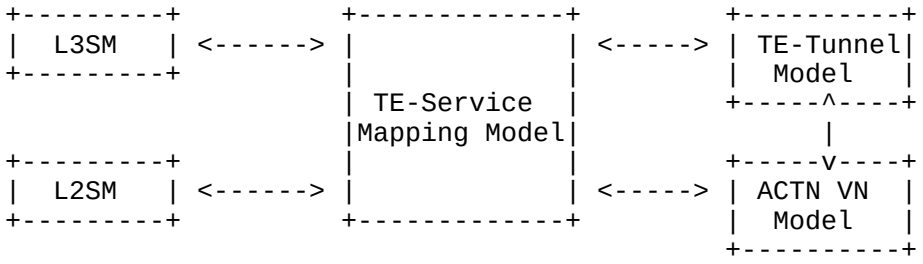
3. The MDSC interacts with both the IP/MPLS PNC and the Transport PNC to create a PE-PE tunnel in the IP network mapped to a TE tunnel in the transport network by providing the inter-layer access points and tunnel requirements. The specific service information are passed to the IP/MPLS PNC for the actual VPN configuration and activation.

    a. The Transport PNC creates the corresponding TE tunnel matching with the access point and egress point.
    b. The IP/MPLS PNC maps the VPN ID with the corresponding TE tunnel ID to bind these two IDs.

4. The IP/MPLS PNC creates/updates a VRF instance for this VPN customer. (This is not covered by the Yang Model presented in this draft).

3. TE-Service Mapping Model

The role of TE-service Mapping model is to create a binding relationship across L3SM and TE Tunnel model via generic ACTN VN Model. The ACTN VN YANG model is a generic virtual network service model that allows customers (internal or external) to create a VN that meets the customer's service objective with various constraints. The TE-service mapping model is needed to bind L3VPN specific service model with TE-specific parameters. This binding will facilitate a seamless service operation with underlay-TE network visibility. The TE-service model developed in this document can also be extended to support other services such as L2SM and so one.



. . .

4. YANG Data Tree

```
module: ietf-te-service-mapping
    +--rw te-service-mapping
    |  +--rw service-mapping
    |  |  +--rw mapping-list* [map-id]
    |  |     +--rw map-id            uint32
    |  |     +--rw map-type?         map-type
    |  |     +--rw (service)?
    |  |     |  +--:(l3vpn)
    |  |     |  |  +--rw l3vpn-ref?        -> /l3:l3vpn-svc/vpn-services/vpn-
service/vpn-id
    |  |     |  +--:(l2vpn)
    |  |     |     +--rw l2vpn-ref?        -> /l2:l2vpn-svc/vpn-services/vpn-
svc/vpn-id
```

```
       |  |     +--rw (te)?
       |  |        +--:(actn-vn)
       |  |        |  +--rw actn-vn-ref?      -> /vn:actn/vn/vn-list/vn-id
       |  |        +--:(te)
       |  |           +--rw te-tunnel-list*   te:tunnel-ref
       |  +--rw site-mapping
       |     +--rw mapping-list* [map-id]
       |        +--rw map-id         uint32
       |        +--rw (service)?
       |        |  +--:(l3vpn)
       |        |  |  +--rw l3vpn-ref?     -> /l3:l3vpn-svc/sites/site/site-id
       |        |  +--:(l2vpn)
       |        |     +--rw l2vpn-ref?     -> /l2:l2vpn-svc/sites/site/site-id
       |        +--rw (te)?
       |           +--:(actn-vn)
       |           |  +--rw actn-vn-ref?   -> /vn:actn/ap/access-point-list/acce
ss-
   point-id
       |           +--:(te)
       +--ro te-service-mapping-state
          +--ro service-mapping
          |  +--ro mapping-list* [map-id]
          |     +--ro map-id            uint32
          |     +--ro map-type?         map-type
          |     +--ro (service)?
          |     |  +--:(l3vpn)
          |     |  |  +--ro l3vpn-ref?        -> /l3:l3vpn-svc/vpn-services/vpn-
   service/vpn-id
          |     |  +--:(l2vpn)
          |     |     +--ro l2vpn-ref?        -> /l2:l2vpn-svc/vpn-services/vpn-
   svc/vpn-id
          |     +--ro (te)?
          |        +--:(actn-vn)
          |        |  +--ro actn-vn-ref?      -> /vn:actn/vn/vn-list/vn-id
          |        +--:(te)
          |           +--ro te-tunnel-list*   te:tunnel-ref
          +--ro site-mapping
             +--ro mapping-list* [map-id]
                +--ro map-id         uint32
                +--ro (service)?
                |  +--:(l3vpn)
                |  |  +--ro l3vpn-ref?     -> /l3:l3vpn-svc/sites/site/site-id
                |  +--:(l2vpn)
                |     +--ro l2vpn-ref?     -> /l2:l2vpn-svc/sites/site/site-id
                +--ro (te)?
                   +--:(actn-vn)
                   |  +--ro actn-vn-ref?   -> /vn:actn/ap/access-point-list/acce
ss-
   point-id
                   +--:(te)
```

5. Yang Data Model

The YANG code is as follows:

<CODE BEGINS> file "ietf-te-service-mapping@2017-03-09.yang"

```
module ietf-te-service-mapping {

    namespace "urn:ietf:params:xml:ns:yang:ietf-te-service-mapping";

    prefix "tm";

    import ietf-l3vpn-svc {
        prefix "l3";
    }

    import ietf-l2vpn-svc {
        prefix "l2";
    }

    import ietf-te {
        prefix "te";
    }

    import ietf-actn-vn {
        prefix "vn";
    }


    organization
        "IETF Traffic Engineering Architecture and Signaling (TEAS)
        Working Group";

    contact
        "Editor: Young Lee <leeyoung@huawei.com>
                 Dhruv Dhody <dhruv.ietf@gmail.com>";

    description
        "This module contains a YANG module for the mapping of
        service (e.g. L3VPN) to the TE tunnels or ACTN VN.";
```

```
        revision 2017-03-09 {
            description
                "initial version.";
            reference
                "TBD";
        }

        /*
         * Identities
         */
        identity service-type {
            description
                "Base identity from which specific service types are
                derived.";
        }

        identity l3vpn-service {
            base service-type;
            description
                "L3VPN service type.";
        }

        identity l2vpn-service {
            base service-type;
            description
                "L2VPN service type.";
        }


        /*
         * Enum
         */
        typedef map-type {
            type enumeration {
                enum "bind" {
                    description
                        "The VN/tunnels are binded to the service";
                }
                enum "select" {
                    description
                        "The VPN service select an existing tunnel";
```

```
                }
            }
            description
                "The map-type";
        }

        /*
         * Groupings
         */
        grouping service-ref{
            description
                "The reference to the service.";
             choice service {
                description
                    "The service";
                case l3vpn {
                    leaf l3vpn-ref {
                        type leafref {
                            path "/l3:l3vpn-svc/l3:vpn-services/"
                            + "l3:vpn-service/l3:vpn-id";
                        }
                        description
                            "The reference to L3VPN Service Yang Model";
                    }
                }
                case l2vpn {
                    leaf l2vpn-ref {
                        type leafref {
                            path "/l2:l2vpn-svc/l2:vpn-services/"
                            + "l2:vpn-svc/l2:vpn-id";
                        }
                        description
                            "The reference to L2VPN Service Yang Model";
                    }

                }

             }
        }

        grouping site-ref {
```

```
            description
                "The reference to the site.";
             choice service {
                description
                    "The service choice";
                case l3vpn {
                    leaf l3vpn-ref{
                        type leafref {
                            path "/l3:l3vpn-svc/l3:sites/l3:site/"
                            + "l3:site-id";
                        }
                        description
                            "The reference to L3VPN Service Yang Model";
                    }
                }
                case l2vpn {
                    leaf l2vpn-ref{
                        type leafref {
                            path "/l2:l2vpn-svc/l2:sites/l2:site/"
                            + "l2:site-id";
                        }
                        description
                            "The reference to L2VPN Service Yang Model";
                    }

                }

             }
        }

        grouping te-ref {
            description
                "The reference to TE.";
            choice te {
                description
                    "The TE";
                case actn-vn {
                    leaf actn-vn-ref {
                        type leafref {
                            path "/vn:actn/vn:vn/vn:vn-list/vn:vn-id";
                        }
```

```
                        description
                            "The reference to ACTN VN";
                    }
                }
                case te {
                    leaf-list te-tunnel-list {
                        type te:tunnel-ref;
                        description
                            "Reference to TE Tunnels";
                    }
                }

            }

        }

        grouping te-endpoint-ref {
            description
                "The reference to TE endpoints.";
            choice te {
                description
                    "The TE";
                case actn-vn {
                    leaf actn-vn-ref {
                        type leafref {
                            path "/vn:actn/vn:ap/vn:access-point-list"
                            + "/vn:access-point-id";
                        }
                        description
                            "The reference to ACTN VN";
                    }
                }
                case te {
                    /*should we refer to Te-topology or Te-tunnel's
   endpoint(?)*/
                }

            }

        }
```

```
       grouping service-mapping {
           description
               "Mapping between Services and TE";
           container service-mapping {
               description
                   "Mapping between Services and TE";

               list mapping-list {
                   key "map-id";
                   description
                       "Mapping identified via a map-id";
                   leaf map-id {
                       type uint32;
                       description
                           "a unique mapping identifier";
                   }
                   leaf map-type {
                       type map-type;
                       description
                           "Tunnel Bind or Tunnel Selection";
                   }
                   uses service-ref;

                   uses te-ref;
               }
           }
       }
       grouping site-mapping {
           description
               "Mapping between VPN access site and TE
               endpoints or AP";
           container site-mapping {
               description
                   "Mapping between VPN access site and TE
                   endpoints or AP";
               list mapping-list {
                   key "map-id";
                   description
                       "Mapping identified via a map-id";
                   leaf map-id {
                       type uint32;
```

```
                    description
                        "a unique mapping identifier";
                }
                uses site-ref;

                uses te-endpoint-ref;
            }

        }
    }



    /*
     * Configuration data nodes
     */
    container te-service-mapping {
        description
            "Mapping between Services and TE";


        uses service-mapping;

        uses site-mapping;
    }

    /*
     * Operational data nodes
     */

    container te-service-mapping-state{
        config false;

        description
            "Mapping between Services and TE";

        uses service-mapping;

        uses site-mapping;
    }
```

```
}

<CODE ENDS>
```

6. Security

This document is an informational draft. When the models mentioned in this draft are implemented, detailed security consideration will be given in such work.

How security fits into the whole architecture has the following components:

- the use of Restconf security between components

- the use of authentication and policy to govern which services can be requested by different parties.

- how security may be requested as an element of a service and mapped down to protocol security mechanisms as well as separation (slicing) of physical resources)

7. IANA Considerations

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

```
---------------------------------------------------------------------
URI: urn:ietf:params:xml:ns:yang:ietf-te-service-mapping
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
---------------------------------------------------------------------
```

This document registers the following YANG modules in the YANG Module

Names registry [RFC7950]:

```
---------------------------------------------------------------------
```

```
name:         ietf-service-mapping
namespace:    urn:ietf:params:xml:ns:yang:ietf-te-service-mapping
prefix:       rip
reference:    RFC XXXX (TDB)
-----------------------------------------------------------------
```

8. Acknowledgements

We thank Diego Caviglia and Igor Bryskin for useful discussions and motivation for this work.

9. References

9.1. Informative References

[RFC4110] R. Callon and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)", RFC 4110, July 2005.

[RFC6020] M. Bjorklund, Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[Service-YANG] Q. Wu, W. Liu and A. Farrel, "Service Models Explained", draft-wu-opsawg-service-model-explained, work in progress.

[Netmod-Yang-Model-Classification] D. Bogdanovic, B. Claise, and C. Moberg, "YANG Module Classification", draft-ietf-netmod-yang-model-classification, work in progress.

[Netconf] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241.

[Restconf] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf, work in progress.

[ACTN-Frame] D. Cecarelli and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks", draft-ietf-teas-actn-framework, work in progress.

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

[TE-Tunnel] T. Saad (Editor), "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te, work in progress.

[ACTN-VN-YANG] Y. Lee (Editor), "A Yang Data Model for ACTN VN Operation", draft-lee-teas-actn-vn-yang, work in progress.

[L3SM-YANG] S. Litkowski, L.Tomotaki, and K. Ogaki, "YANG Data Model for L3VPN service delivery", draft-ietf-l3sm-l3vpn-service-model, work in progress.

10. Contributors

Authors' Addresses

Young Lee
Huawei Technologies
5340 Legacy Drive
Plano, TX 75023, USA
Phone: (469)277-5838

Email: leeyoung@huawei.com

Dhruv Dhody
Huawei Technologies

Email: dhruv.ietf@gmail.com

Daniele Ceccarelli
Ericsson
Torshamnsgatan,48
Stockholm, Sweden

Email: daniele.ceccarelli@ericsson.com

TEAS Working Group
Internet-Draft
Intended status: Informational
Expires: August 21, 2017

H. Sitaraman, Ed.
V. Beeram
Juniper Networks
I. Minei
Google, Inc.
S. Sivabalan
Cisco Systems, Inc.
February 17, 2017

Recommendations for RSVP-TE and Segment Routing LSP co-existence
draft-sitaraman-sr-rsvp-coexistence-rec-02.txt

Abstract

Operators are looking to introduce services over Segment Routing (SR) LSPs in networks running Resource Reservation Protocol (RSVP-TE) LSPs. In some instances, operators are also migrating existing services from RSVP-TE to SR LSPs. For example, there might be certain services that are well suited for SR and need to co-exist with RSVP-TE in the same network. In other cases, services running on RSVP-TE might be migrated to run over SR. Such introduction or migration of traffic to SR might require co-existence with RSVP-TE in the same network for an extended period of time depending on the operator’s intent. The following document provides solution options for keeping the traffic engineering database (TED) consistent across the network, accounting for the different bandwidth utilization between SR and RSVP-TE.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 21, 2017.

Table of Contents

1. Introduction

Introduction of SR [I-D.ietf-spring-segment-routing] in the same network domain as RSVP-TE [RFC3209] presents the problem of accounting for SR traffic and making RSVP-TE aware of the actual available bandwidth on the network links. RSVP-TE is not aware of how much bandwidth is being consumed by SR services on the network links and hence both at computation time (for a distributed computation) and at signaling time RSVP-TE LSPs will incorrectly place loads. This is true where RSVP-TE paths are distributed or centrally computed without a common entity managing both SR and RSVP-TE computation for the entire network domain.

The problem space can be generalized as a dark bandwidth problem to cases where any other service exists in the network that runs in parallel across common links and whose bandwidth is not reflected in the available and reserved values in the TED. The general problem is management of dark bandwidth pools and can be generalized to cases where any other service exists in the network that runs in parallel across common links and whose bandwidth is not reflected in the available and reserved values in the TED. In most practical instances given the static nature of the traffic demands, limiting the available reservable bandwidth available to RSVP-TE has been an acceptable solution. However, in the case of SR traffic, there is assumed to be very dynamic traffic demands and there is considerable risk associated with stranding capacity or overbooking service traffic resulting in traffic drops.

The high level requirements or assumptions to consider are:

1. Placement of SR LSPs in the same domain as RSVP-TE LSPs MUST NOT introduce inaccuracies in the TED used by distributed or centralized path computation engines.

2. Engines that compute RSVP-TE paths MAY have no knowledge of the existence of the SR paths in the same domain.

3. Engines that compute RSVP-TE paths SHOULD NOT require a software upgrade or change to their path computation logic.

4. Protocol extensions SHOULD be avoided or be minimal as in many cases this co-existence of RSVP-TE and SR MAY be needed only during a transition phase.

5. Placement of SR LSPs in the same domain as RSVP-TE LSPs that are computed in a distributed fashion MUST NOT require migration to a central controller architecture for the RSVP-TE LSPs.

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 [RFC2119].

3. Solution options

3.1. Static partitioning of bandwidth

In this model, the static reservable bandwidth of an interface can be statically partitioned between SR and RSVP-TE and each can operate within that bandwidth allocation and SHOULD NOT preempt each other.

While it is possible to configure RSVP-TE to only reserve up to a certain maximum link bandwidth and manage the remaining link bandwidth for other services, this is a deployment where SR and RSVP-TE are separated in the same network (ships in the night) and can lead to suboptimal link bandwidth utilization not allowing each to consume more, if required and constraining the respective deployments.

The downside of this approach is the inability to use the reservable bandwidth effectively and inability to use bandwidth left unused by the other protocol.

3.2. Centralized management of available capacity

In this model, a central controller performs path placement for both RSVP-TE and SR LSPs. The controller manages and updates its own view of the in-use and the available capacity. As the controller is a single common entity managing the network it can have a unified and consistent view of the available capacity at all times.

A practical drawback of this model is that it requires the introduction of a central controller managing the RSVP-TE LSPs as a prerequisite to the deployment of any SR LSPs. Therefore, this approach is not practical for networks where distributed TE with RSVP-TE LSPs is already deployed, as it requires a redesign of the network and is not backwards compatible. This does not satisfy requirement 5.

Note that it is not enough for the controller to just maintain the unified view of the available capacity, it must also perform the path computation for the RSVP-TE LSPs, as the reservations for the SR LSPs are not reflected in the TED. This does not fit with assumption 2 mentioned earlier.

3.3. Flooding SR utilization in IGP

Using techniques in [RFC7810], [RFC7471] and [RFC7823], the SR utilization information can be flooded in IGP-TE and the RSVP-TE path computation engine (CSPF) can be changed to consider this information. This requires changes to the RSVP-TE path computation logic and would require upgrades in deployments where distributed computation is done across the network.

This does not fit with requirements 3 and 4 mentioned earlier.

3.4. Running SR over RSVP-TE

SR can run over dedicated RSVP-TE LSPs that carry only SR traffic. In this model, the LSPs can be one-hop or multi-hop and can provide bandwidth reservation for the SR traffic based on functionality such as auto-bandwidth. The model of deployment would be similar in nature to running LDP over RSVP-TE. This would allow the TED to stay consistent across the network and any other RSVP-TE LSPs will also be aware of the SR traffic reservations. In this approach, non-SR traffic MUST NOT take the SR-dedicated RSVP-TE LSPs, unless required by policy.

The drawback of this solution is that it requires SR to rely on RSVP-TE for deployment. Furthermore, the accounting accuracy/frequency of this method is dependent on performance of auto-bandwidth for RSVP-TE. Note that for this method to work, the SR-dedicated RSVP-TE LSPs must be set up with the best setup and hold priorities in the network.

3.5. TED consistency by reflecting SR traffic

The solution relies on dynamically measuring SR traffic utilization on each TE interface and reducing the bandwidth allowed for use by RSVP-TE. It is assumed that SR traffic receives precedence in terms of the placement on the path over RSVP traffic (that is, RSVP traffic can be preempted from the path in case of insufficient resources). This is logically equivalent to SR traffic having the best preemption priority in the network. Note that this does not necessarily mean that SR traffic has higher QoS priority, in fact, SR and RSVP traffic may be in the same QoS class. The following methodology can be used at every TE node for this solution:

- T: Traffic statistics collection time interval

- N: Traffic averaging calculation (adjustment) interval such that N = k * T, where k is a constant integer multiplier greater or equal to 1. Its purpose is to provide a smoothing function to the statistics collection.

- Maximum-Reservable-Bandwidth: The maximum available bandwidth for TE (this is the maximum available bandwidth on the interface, before any LSP reservations).

  If Differentiated-Service (Diffserv)-aware MPLS Traffic Engineering (DS-TE) [RFC4124] is enabled, the Maximum-Reservable-Bandwidth SHOULD be interpreted as the aggregate bandwidth constraint across all Class-Types independent of the Bandwidth Constraints model.

- RSVP-unreserved-bandwidth-at-priority-X: Maximum-Reservable-Bandwidth - sum of (existing reservations at priority X and all priorities better than X)

- SR traffic threshold percentage: The percentage difference of traffic demand that when exceeded can result in a change to the RSVP-TE Maximum-Reservable-Bandwidth

- IGP-TE update threshold: Specifies the frequency at which IGP-TE updates should be triggered based on TE bandwidth updates on a link

- M: An optional multiplier that can be applied to the SR traffic average. This multiplier provides the ability to grow or shrink the bandwidth used by SR

At every interval T, each node SHOULD collect the SR traffic statistics for each of its TE interfaces. Further, at every interval N, given a configured SR traffic threshold percentage and a set of collected SR traffic statistics samples across the interval N, the SR traffic average (or any other traffic metric depending on the algorithm used) over this period is calculated.

If the difference between the new calculated SR traffic average and the current SR traffic average (that was computed in the prior adjustment) is at least SR traffic threshold percentage, then two values MUST be updated:

- New Maximum-Reservable-Bandwidth = Current Maximum-Reservable-Bandwidth - (SR traffic average * M)

- New RSVP-unreserved-bandwidth-at-priority-X = New Maximum-Reservable-Bandwidth - sum of (existing reservations at priority X and all priorities better than X)

A DS-TE LSR that advertises Bandwidth Constraints TLV should update the bandwidth constraints for class-types based on operator policy. For example, when Russian Dolls Model (RDM) [RFC4127] is in use, then only BC0 may be updated. Whereas, when Maximum Allocation Model (MAM) [RFC4125] is in use, then all BCs may be updated equally such that the total value updated is equal to the newly calculated SR traffic average.

Note that the computation of the new RSVP-unreserved-bandwidth-at-priority-X MAY result in RSVP-TE LSPs being hard or soft preempted. Such preemption will be based on relative priority (e.g. low to high) between RSVP-TE LSPs. It is RECOMMENDED that the IGP-TE update threshold SHOULD be lower in order to flood unreserved bandwidth

updates often. From an operational point of view, an implementation SHOULD be able to expose both the configured and the actual values of the Maximum-Reservable-Bandwidth.

If LSP preemption is not acceptable, then the RSVP-TE Maximum-Reservable-Bandwidth cannot be reduced below what is currently reserved by RSVP-TE on that interface. This may result in bandwidth not being available for SR traffic. Thus, it is required that any external controller managing SR LSPs SHOULD be able to detect this situation (for example by subscribing to TED updates [RFC7752]) and SHOULD take action to reroute existing SR paths.

Generically, SR traffic (or any non-RSVP-TE traffic) should have its own priority allocated from the available priorities. This would allow SR to preempt other traffic according to the preemption priority order.

In this solution, the logic to retrieve the statistics, calculating averages and taking action to change the Maximum-Reservable-Bandwidth is an implementation choice, and all changes are local in nature. However, note that this is a new network trigger for RSVP-TE preemption and thus is a consideration for the operator.

The above solution offers the advantage of not introducing new network-wide mechanisms especially during scenarios of migrating to SR in an existing RSVP-TE network and reusing existing protocol mechanisms.

4. Acknowledgements

The authors would like to thank Steve Ulrich for his detailed review and comments.

5. Contributors

The following individuals contributed to this document:

Chandra Ramachandran
Juniper Networks
Email: csekar@juniper.net

Raveendra Torvi
Juniper Networks
Email: rtorvi@juniper.net

Sudharsana Venkataraman
Juniper Networks
Email: sudharsana@juniper.net

6. IANA Considerations

This draft does not have any request for IANA.

7. Security Considerations

No new security issues are introduced in this document beyond is already part of RSVP-TE and Segment routing architectures.

8. References

8.1. Normative References

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-11 (work in progress), February 2017.

[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>.

[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>.

8.2. Informative References

[RFC4124] Le Faucheur, F., Ed., "Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering", RFC 4124, DOI 10.17487/RFC4124, June 2005, <http://www.rfc-editor.org/info/rfc4124>.

[RFC4125] Le Faucheur, F. and W. Lai, "Maximum Allocation Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering", RFC 4125, DOI 10.17487/RFC4125, June 2005, <http://www.rfc-editor.org/info/rfc4125>.

[RFC4127] Le Faucheur, F., Ed., "Russian Dolls Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering", RFC 4127, DOI 10.17487/RFC4127, June 2005, <http://www.rfc-editor.org/info/rfc4127>.

[RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <http://www.rfc-editor.org/info/rfc7471>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <http://www.rfc-editor.org/info/rfc7752>.

[RFC7810] Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 7810, DOI 10.17487/RFC7810, May 2016, <http://www.rfc-editor.org/info/rfc7810>.

[RFC7823] Atlas, A., Drake, J., Giacalone, S., and S. Previdi, "Performance-Based Path Selection for Explicitly Routed Label Switched Paths (LSPs) Using TE Metric Extensions", RFC 7823, DOI 10.17487/RFC7823, May 2016, <http://www.rfc-editor.org/info/rfc7823>.

Authors' Addresses

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

Email: hsitaraman@juniper.net

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

Email: vbeeram@juniper.net

Ina Minei
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: inaminei@google.com

Siva Sivabalan
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, Ontario K2K 3E8
Canada

Email: msiva@cisco.com

TEAS Working Group
Internet Draft

A.Wang
China Telecom
Quintin Zhao
Boris Khasanov
Huawei Technologies
Penghui Mi
Tencent Company
Raghavendra Mallya
Juniper Networks
Shaofu Peng
ZTE Corporation

Intended status: Standard Track
March 13, 2017
Expires: September 12, 2017

PCE in Native IP Network
draft-wang-teas-pce-native-ip-03.txt

Status of this Memo

Abstract

This document defines the scenario and solution for traffic engineering within Native IP network, using Dual/Multi-BGP session strategy and PCE-based central control architecture. The proposed central mode control solution conforms to the concept that defined in draft [I-D.draft-ietf-teas-pce-control-function]. And together with draft [I-D.draft-ietf-teas-pcecc-use-cases], the solution portfolio for traffic engineering in MPLS and Native IP network is almost completed.

Table of Contents

1. Introduction

Currently, PCE based traffic assurance requires the underlying network devices support MPLS and the network must deploy multiple LSPs to assure the end-to-end traffic performance. LDP/RSVP-TE or Segment Routing should be enabled within the network to establish various MPLS paths. Such solution will certainly work but they does not cover the needs in legacy Native IP network, which demands less signaling protocol and less complex traffic steering policy.

Within Native IP network, the solution for traffic engineering is generally hop-by-hop differentiate treatment. To achieve the end2end QoS performance assurance, one can only deploy some dedicated links statically, but such solution is not feasible in the service provider network, because the complexity of underlying network and the variation of application traffic from time to time.

In summary, the requirements for traffic engineering in Native IP network are the following:
1) No complex MPLS signaling procedure.
2) End to End traffic assurance, determined QoS behavior.
3) Flexible deployment and automation control.

This document defines the solution for traffic engineering within Native IP network, using Dual/Multi-BGP session strategy and PCE-based central control architecture, to meet the above requirements in dynamical and central control mode. Future PCEP protocol extensions to transfer the key parameters between PCE and the underlying network devices(PCC) are provided in draft [draft-wang-pcep-extension-native-IP]

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 [RFC2119].

3. Dual-BGP solution for simple topology.

This section introduces the dual-BGP solution for simple topology that illustrated in Fig.1, which is comprised by SW1, SW2, R1, R2. There are multiple physical links between R1 and R2. Let’s assume traffic between IP11 and IP21 is normal traffic, traffic between IP12

and IP22 is priority traffic that should be treated differently.

Only Native IGP/BGP protocol is deployed between R1 and R2. The traffic between each address pair may change timely and the corresponding source/destination addresses of the traffic may also change dynamically.

The key idea of the Dual-BGP solution for this simple topology is the following:

1) Build two BGP sessions between R1 and R2, via the different loopback address lo0, lo1 on these routers.
2) Send different prefixes via the two BGP sessions. (For example, IP11/IP21 via the BGP pair 1 and IP12/IP22 via the BGP pair 2).
3) Set the explicit peer route on R1 and R2 respectively for BGP next hop of lo0, lo1 to different physical link address between R1 and R2.

So, the traffic between the IP11 and IP21, and the traffic between IP12 and IP22 will go through different physical links between R1 and R2, each type of traffic occupy the different dedicated physical links.

If there is more traffic between IP12 and IP22 that needs to be assured , one can add more physical links on R1 and R2 to reach the loopback address lo1(also the next hop for BGP Peer pair2). In this cases the prefixes that advertised by two BGP peer need not be changed.

If, for example, there is traffic from another address pair that needs to be assured (for example IP13/IP23), but the total volume of assured traffic does not exceed the capacity of the previous appointed physical links, then one need only to advertise the newly added source/destination prefixes via the BGP peer pair2, then the traffic between IP13/IP23 will go through the assigned dedicated physical links as the traffic between IP12/IP22.

Such decouple philosophy gives the network operator more flexible control ability on the network traffic, get the determined QoS assurance effect to meet the application’s requirement. No complex MPLS signal procedures is introduced, the router need only support native IP protocol.

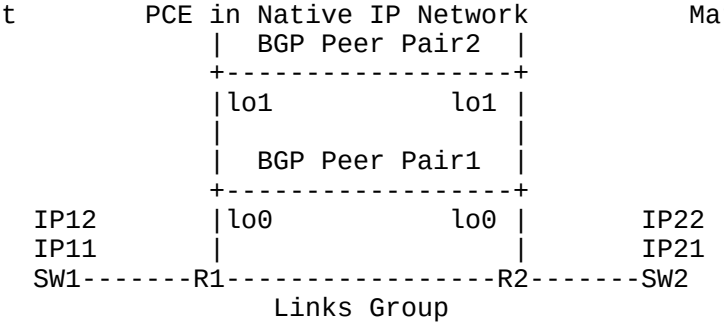


Fig.1 Design Philosophy for Dual-BGP Solution

4. Dual-BGP in large Scale Topology

When the assured traffic spans across one large scale network, as that illustrated in Fig.2, the dual BGP sessions cannot be established hop by hop especially for the iBGP within one AS. For such scenario, we should consider to use the Route Reflector (RR) to achieve the similar Dual-BGP effect, that is to say, select one router which performs the role of RR (for example R3 in Fig.2 - Dual-BGP Solution using Route Reflector for large scale network), every other edge router will establish two BGP peer sessions with the RR, using their different loopback addresses respectively(the inner router will establish one BGP session with RR). The other two steps for traffic differentiation are same as one described in the Dual-BGP simple topology usage case.

For the example shown in Fig.2, if we select the R1-R2-R4-R7 as the dedicated path, then we should set the explicit peer routes on these routers respectively, pointing to the BGP next hop (loopback addresses of R1 and R7, which are used to send the prefix of the assured traffic) to the actual address of the physical link

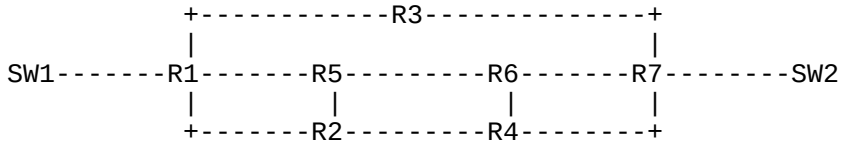


Fig.2 Dual-BGP solution for large scale network

5. Multi-BGP for Extended Traffic Differentiation

In general situation, several additional traffic differentiation criteria exist, including:
o Traffic that requires low latency links and is not sensitive to packet loss
o Traffic that requires low packet loss but can endure higher latency
o Traffic that requires lowest jitter path
o Traffic that requires high bandwidth links

These different traffic requirements can be summarized in the following table:

Flow No.	Latency	Packet Loss	Jitter
1	Low	Normal	Don’t care
2	Normal	Low	Dont’t care
3	Normal	Normal	Low

Table 1. Traffic Requirement Criteria

For Flow No.1, we can select the shortest distance path to carry the traffic; for Flow No.2, we can select the idle links to form its end to end path; for Flow No.3, we can let all the traffic pass one single path, no ECMP distribution on the parallel links is required.

It is difficult and almost impossible to provide an end-to-end (E2E) path with latency, latency variation, packet loss, and bandwidth utilization constraints to meet the above requirements in large scale IP-based network via the traditional distributed routing protocol, but these requirements can be solved using the PCE-based architecture since the PCE has the overall network view, can collect real network topology and network performance information about the underlying network, select the appropriate path to meet the various network performance requirements of different traffic type.

6. PCE based solution for Multi-BGP strategy deployment.

With the advent of SDN concepts towards pure IP networks, it is

possible to deploy the PCE related technology into the underlying native IP network, to accomplish the central and dynamic control of network traffic according to the application's various requirements.

The procedure to implement the dynamic deployment of Multi-BGP strategy is the following:

1) PCE gets topology and link utilization information from the underlying network, calculate the appropriate link path upon application's requirements.
2) PCE sends the key parameters to edge/RR routers(R1, R7 and R3 in Fig.3) to build multi-BGP peer relations and advertise different prefixes via them.
3) PCE sends the route information to the routers (R1,R2,R4,R7 in Fig.3) on forwarding path via PCEP, to build the path to the BGP next-hop of the advertised prefixes.
4) If the assured traffic prefixes were changed but the total volume of assured traffic does not exceed the physical capacity of the previous end-to-end path, then PCE needs only change the related information on edge routers (R1,R7 in Fig.3).
5) If volume of the assured traffic exceeds the capacity of previous calculated path, PCE must recalculate the appropriate path to accommodate the exceeding traffic via some new end-to-end physical link. After that PCE needs to update on-path routers to build such path hop by hop.

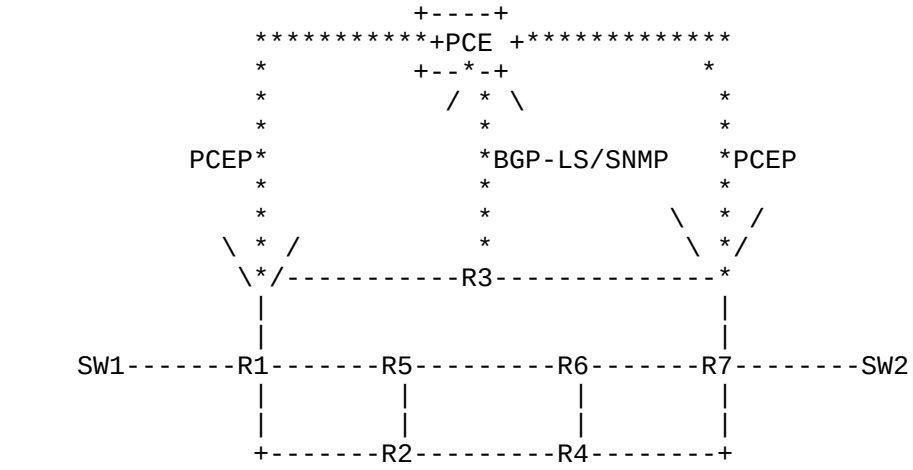


Fig.3 PCE based solution for Multi-BGP deployment

7. PCEP extension for key parameters delivery.

We need to extend the PCEP protocol to transfer the following key parameters:
1) BGP peer address and advertised prefixes.
2) Explicit route information to BGP next hop of advertised prefixes.

Once the router receives such information, it should establish the BGP session with the peer appointed in the PCEP message, advertise the prefixes that contained in the corresponding PCEP message, and build the end to end dedicated path hop by hop. Details of communications between PCEP and BGP subsystems in router's control plane are out of scope of this draft and will be described in separate draft.[draft-wang-pce-extension for native IP]

The reason why we selected PCEP as the southbound protocol instead of OpenFlow, is that PCEP is very suitable for the changes in control plane of the network devices, there OpenFlow dramatically changes the forwarding plane. We also think that the level of centralization that requires by OpenFlow is hardly achievable in many today's SP networks so hybrid BGP+PCEP approach looks much more interesting.

8. Deployment Consideration

This solution requires the parallel work of 2 subsystems in router's control plane: PCE (PCEP) and BGP as well as coordination between them, so it might require additional planning work before deployment.

8.1 Scalability

In current solution, PCE need only to influence the edge routers for the prefixes differentiation via the multi-BGP deployment. The route information for these prefixes within the on-path routers were distributed via the traditional BGP protocol. Unlike the solution from BGP Flowspec, the on-path router need only keep the specific policy routes to the BGP next-hop of the differentiate prefixes, not the specific routes to the prefixes themselves. This can lessen the burden from the table size of policy based routes for the on-path routers, and has more scalability when comparing with the solution from BGP flowspec or Openflow.

8.2 High Availability

Current solution is based on the traditional distributed IP protocol, then if the central control PCE failed, the forwarding plane will not be impacted, as the BGP session between all devices will not flap, and the forwarding table will remain the same. If one node on the optimal path is failed, the assurance traffic will fall over to the best-effort forwarding path. One can even design several assurance paths to load balance/hot standby the assurance traffic to meet the path failure situation, as done in MPLS FRR.
From PCE/SDN-controller HA side we will rely on existing HA solutions of SDN controllers such as clustering.

8.3 Incremental deployment

Not every router within the network support will support the PCEP extension that defined in [draft-wang-pce-extension-native-IP] simultaneously. For such situations, router on the edge of sub domain can be upgraded first, and then the traffic can be assured between different sub domains. Within each sub domain, the traffic will be forwarded along the best-effort path. Service provider can selectively upgrade the routers on each sub-domain in sequence.

8.4 Deployment within Pure IGP network

For some small underlying networks where the routers support only the pure IGP protocol, we can use EVPN/VxLAN technology and similar procedures that described within this draft to differentiate the forwarding paths for different applications:

1) PCE instructs the IGP edge router (ABR) build different BGP sessions.

2) PCE instructs the IGP edge router (ABR) redistribute external prefixes via different BGP sessions under the EVPN address family, and then different external prefixes will be associated with different VTEP addresses.

3) PCE calculates the optimal path and instruct the on-path routers to build the explicit peer routes to the different VTEP addresses (also the different loopback addresses on ABR).

The traffic will then be forwarded via the VxLAN encapsulation, the route path of them will be determined by the outer tunnel address, which is calculated and programmed by PCE.

The detail of deployment scenario and the corresponding PCEP extension will be exploited further later.

9. Security Considerations

TBD

10. IANA Considerations

TBD

11. Conclusions

TBD

12. References

12.1. Normative References

[RFC4655] Farrel, A., Vasseur, J.-P., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006,<http://www.rfc-editor.org/info/rfc4655>.

[RFC5440]Vasseur, JP., Ed., and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009, <http://www.rfc-editor.org/info/rfc5440>.

12.2. Informative References

[I-D.draft-ietf-teas-pce-control-function]

A.Farrel, Q.Zhao et al. "An Architecture for use of PCE and PCEP in a Network with Central Control"

https://datatracker.ietf.org/doc/draft-ietf-teas-pce-central-control/ September, 2016

[I-D. draft-ietf-teas-pcecc-use-cases]

Quintin Zhao, Robin Li, Boris Khasanov et al. "The Use Cases for Using PCE as the Central Controller(PCECC) of LSPs

https://tools.ietf.org/html/draft-ietf-teas-pcecc-use-cases-00

March,2017

[draft-wang-pcep-extension for native IP]

Aijun Wang, Boris Khasanov et al. "PCEP Extension for Native IP Network" https://datatracker.ietf.org/doc/draft-wang-pce-extension-native-ip/

13. Acknowledgments

The authors would like to thank George Swallow, Xia Chen, Jeff Tantsura, Daniele Ceccarelli and Dhruv Dhody for their valuable comments and suggestions.

The authors would also like to thank Lou Berger, Adrian Farrel, King Daniel for their suggestions to put forward this draft.

Authors' Addresses

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing,China

Email: wangaj.bri@chinatelecom.cn

Quintin Zhao
Huawei Technologies
125 Nagog Technology Park
Acton, MA 01719
US

EMail: quintin.zhao@huawei.com

Boris Khasanov
Huawei Technologies
Moskovskiy Prospekt 97A
St.Petersburg 196084
Russia

EMail: khasanov.boris@huawei.com

Penghui Mi
Tencent
Tencent Building, Kejizhongyi Avenue,
Hi-techPark, Nanshan District,Shenzhen 518057, P.R.China

Email kevinmi@tencent.com

Raghavendra Mallya
Juniper Networks
1133 Innovation Way
Sunnyvale, California 94089 USA

Email: rmallya@juniper.net

Shaofu Peng
ZTE Corporation
No.68 Zijinghua Road,Yuhuatai District
Nanjing 210012
China

Email: peng.shaofu@zte.com.cn

TEAS WG

Young Lee
Haomian Zheng
Huawei

Internet Draft
Intended status: Informational

Daniel Ceccarrelli

Expires: September 2017
Ericsson

Bin Yeong Yoon
ETRI

Oscar Gonzalez de Dios
Telefonica

Jong Yoon Shin
SKT

Sergio Belotti
Nokia

March 13, 2017

Applicability of YANG models for Abstraction and Control of Traffic Engineered Networks

draft-zhang-teas-actn-yang-04

Status of this Memo

This Internet-Draft will expire on September 13, 2017.

Abstract

Abstraction and Control of TE Networks (ACTN) refers to the set of virtual network operations needed to orchestrate, control and manage large-scale multi-domain TE networks, so as to facilitate network programmability, automation, efficient resource sharing, and end-to-end virtual service aware connectivity and network function virtualization services.

This document explains how the different types of YANG models defined in the Operations and Management Area and in the Routing Area are applicable to the ACTN framework. This document also shows how the ACTN architecture can be satisfied using classes of data model that have already been defined, and discusses the applicability of specific data models that are under development. It also highlights where new data models may need to be developed.

Table of Contents

1. Introduction

Abstraction and Control of TE Networks (ACTN) describes a method for operating a Traffic Engineered (TE) network (such as an MPLS-TE network or a layer 1 transport network) to provide connectivity and virtual network services for customers of the TE network. The services provided can be tuned to meet the requirements (such as traffic patterns, quality, and reliability) of the applications hosted by the customers. More details about ACTN can be found in Section 2.

Data models are a representation of objects that can be configured or monitored within a system. Within the IETF, YANG [RFC6020] is the language of choice for documenting data models, and YANG models have been produced to allow configuration or modelling of a variety of network devices, protocol instances, and network services. YANG data models have been classified in [Netmod-Yang-Model-Classification] and [Service-YANG].

This document shows how the ACTN architecture can be satisfied using classes of data model that have already been defined, and discusses the applicability of specific data models that are under development. It also highlights where new data models may need to be developed.

2. Abstraction and Control of TE Networks (ACTN) Architecture

[ACTN-Requirements] describes the high-level ACTN requirements. [ACTN-Frame] describes the architecture model for ACTN including the entities (Customer Network Controller (CNC), Multi-domain Service Coordinator (MDSC), and Physical Network Controller (PNC)) and their interfaces.

Figure 1 depicts a high-level control and interface architecture for ACTN and is a reproduction of Figure 7 from [ACTN-Frame]. A number of key ACTN interfaces exist for deployment and operation of ACTN-based networks. These are highlighted in Figure 1 (ACTN Interfaces) below:

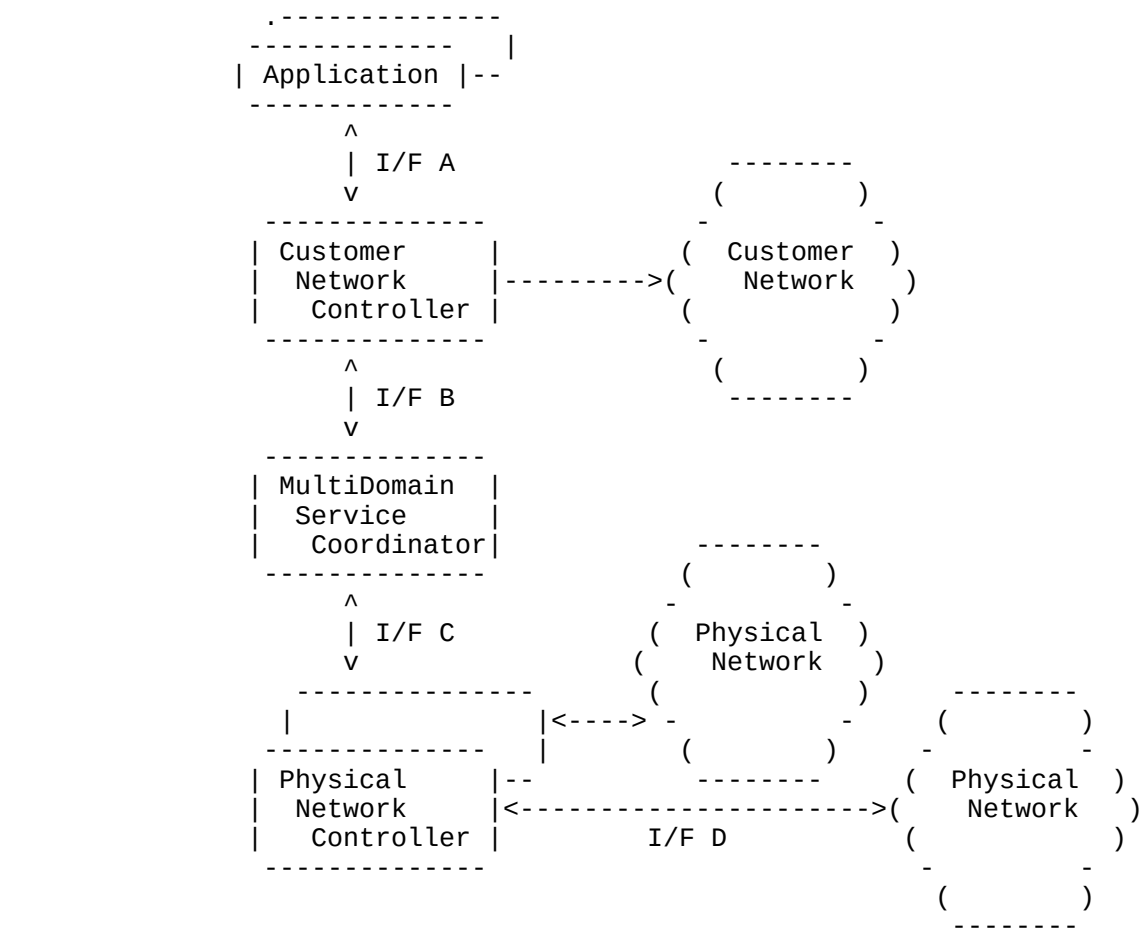


Figure 1 : ACTN Interfaces

The interfaces and functions are described below (without modifying the definitions) in [ACTN-Frame]:

. Interface A: This is out of scope of this draft.

. Interface B: The CNC-MDSC Interface (CMI) is an interface between a Customer Network Controller and a Multi Domain Service Controller. The interface will communicate the service request or application demand. A request will include specific service properties, for example, services type, bandwidth and constraint information. These constraints SHOULD be measurable by MDSC and therefore visible to CNC via CMI. The CNC can also

request the creation of the virtual network based on underlying physical resources to provide network services for the applications. The CNC can provide the end-point information/characteristics, traffic matrix specifying specific customer constraints. The MDSC 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 allows the MDSC to communicate requests to create/delete connectivity or to modify bandwidth reservations in the physical network. In multi-domain environments, each PNC is responsible for a separate domain. The MDSC needs to establish multiple MPIs, one for each PNC and perform coordination between them to provide cross-domain connectivity.

. Interface D: The provisioning interface for creating forwarding state in the physical network, requested via the Physical Network Controller. Interface D is not in the scope of ACTN, however, it is included in this document so that it can be compared to models in [Service-Yang].

3. Service Models

[Service-YANG] introduces a reference architecture to explain the nature and usage of service YANG models in the context of service orchestration. Figure 2 below depicts this relationship and is a reproduction of Figure 2 from [Service-YANG]. Four models depicted in Figure 2 are defined as follows:

. Customer Service Model: A customer service model is used to describe a service as offer or delivered to a customer by a network operator.
. Service Delivery Model: A service delivery model is used by a network operator to define and configure how a service is provided by the network.
. Network Configuration Model: A network configuration model is used by a network orchestrator to provide network-level configuration model to a controller.
. Device Configuration Model: A device configuration model is used by a controller to configure physical network elements.

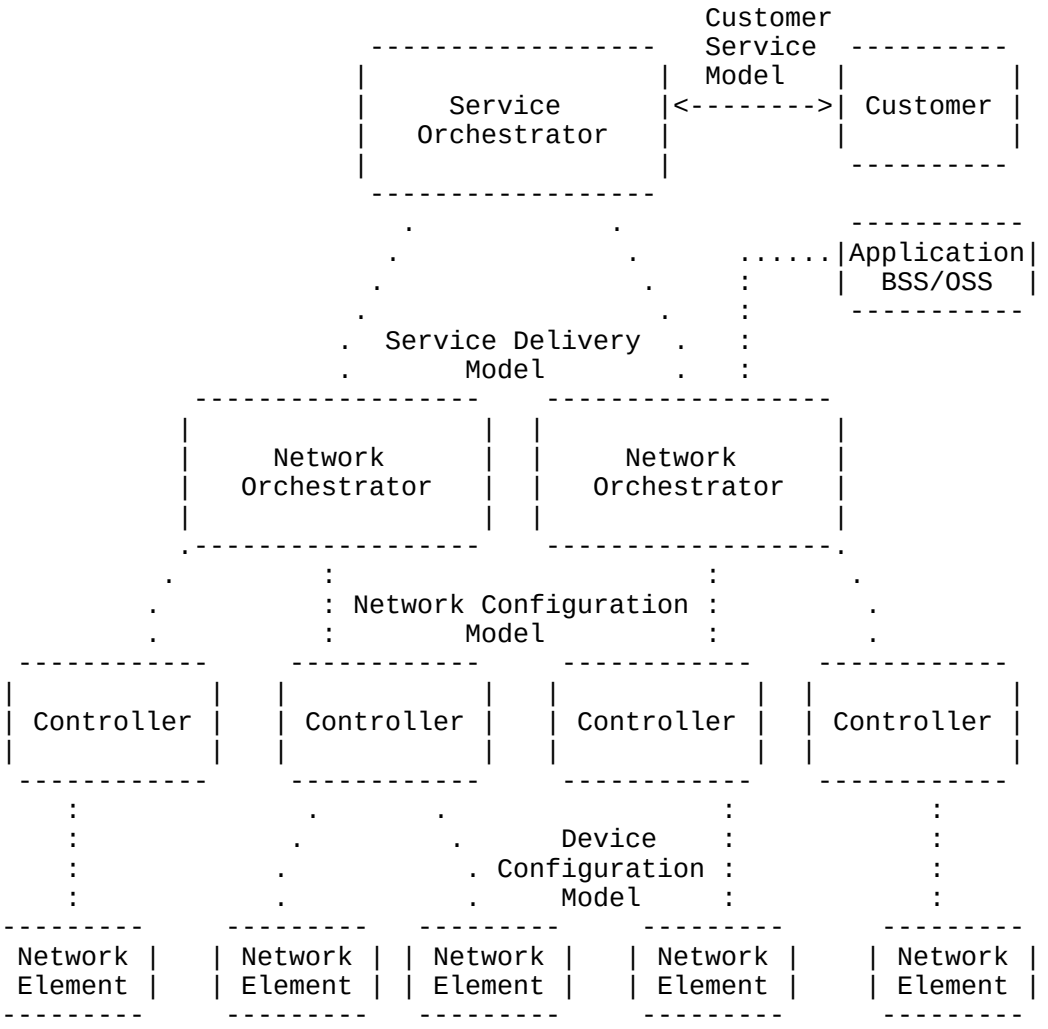


Figure 2: An SDN Architecture with a Service Orchestrator

4. Service Model Mapping to ACTN

YANG models coupled with the RESTCONF/NETCONF protocol [Netconf][Restconf] are solutions for the ACTN framework. This section explains which types of YANG models apply to each of the ACTN interfaces.

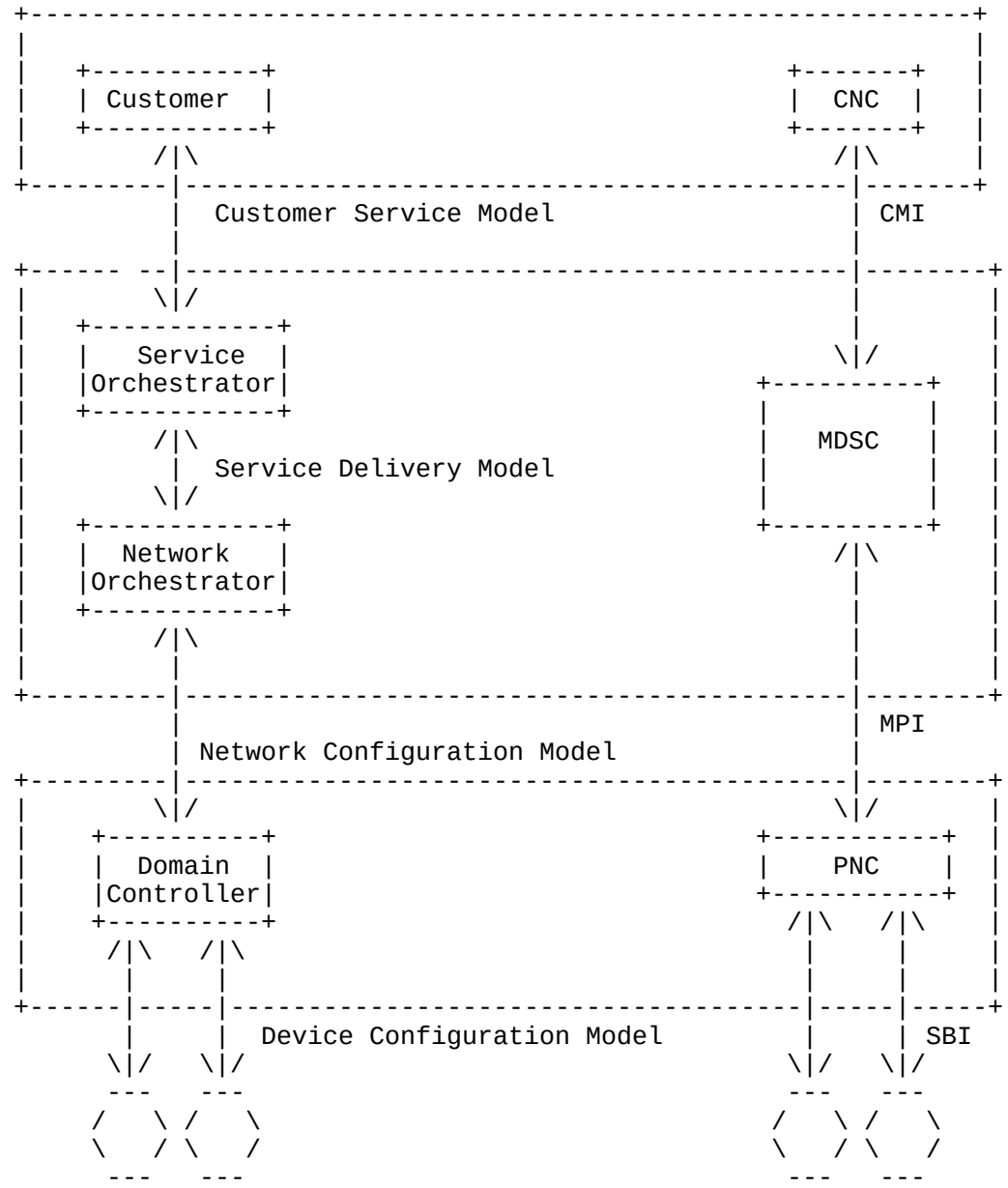


Figure 3 : Mapping between Service Model and ACTN Architecture

As shown in Figure 3, the architecture described in [Service-YANG] can be mapped to the ACTN architecture well. A point worth noting is that here the functions of the MDSC can include:

A. Customer mapping/translation

This function is to map customer intent-like commands received on CMI into network provisioning requests, and therefore sends the request message to the Physical Network Controller (PNC) via MPI according to business OSS/NMS provisioned static or dynamic policy. Specifically, it provides mapping and translation of customer's service request into a set of parameters that are specific to a network type and technology such that network configuration process is made possible.

B. Service/Virtual service coordination

This function translates customer service-related information into the virtual network service operations in order to seamlessly operate virtual networks while meeting customer's service requirements. In the context of ACTN, service/virtual service coordination includes a number of service orchestration functions such as multi-destination load balancing, guarantees of service quality, bandwidth and throughput and notification for service fault and performance degradation and so forth.

C. Multi domain coordination

This function oversees the specific aspects of the different domains and builds a single abstracted end-to-end network topology in order to coordinate end-to-end path computation and path/service provisioning. Domain sequence path calculation/determination is also a part of this function.

D. Virtualization/Abstraction

This function provides an abstracted view of the underlying network resources towards customer, being it the client or a higher level controller entity. It includes network path computation based on customer service connectivity request constraints, 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 network objective functions, and to the customer traffic profile.

The first two of these functions (A and B) are related to service orchestration while function C is related to network orchestration and function D is related to both service and network orchestration.

4.1. Customer Service Models in the ACTN Architecture (CMI)

Customer Service Models, which are used between a customer and a service orchestrator as in [Service-YANG], should be used between the CNC and MDSC (e.g., CMI) serving as providing a simple intent-like model/interface.

Among the key functions of Customer Service Models on the CMI is the service request. A request will include specific service properties, including: service type and its characteristics, bandwidth, constraint information, and end-point characteristics.

The following table provides a list of functions needed to build the CMI. They are mapped with Customer Service Models.

Function	Yang Model
Transport Service Request	[Transport-Service-Model]
VN Service Request & Instantiation	[ACTN-VN-YANG]
VN Path Computation Request	[ACTN-VN-YANG]*
VN Performance Monitoring Telemetry	[ACTN-PM-Telemetry]**

*VN Path computation request in the CMI context means network path computation request based on customer service connectivity request constraints prior to the instantiation of a VN creation.

**ietf-actn-te-kpi-telemetry model describes performance telemetry for ACTN VN model. This module also allows autonomic traffic engineering scaling intent configuration mechanism on the VN level. Scale in/out criteria might be used for network autonomics in order the controller to react to a certain set of variations in monitored parameters. Moreover, this module also provides mechanism to define aggregated telemetry parameters as a grouping of underlying VN level telemetry parameters.

4.2. Service Delivery Models in ACTN Architecture

The Service Delivery Models (as shown in Figure 3) where the service orchestration and the network orchestration could be implemented as

separate components as seen in [Service-YANG]. This is also known as Network Service Models. On the other hand, from an ACTN architecture point of view, the service delivery model between the service orchestrator and the network orchestrator is an internal interface between sub-components of the MDSC in a single MDSC model.

In the MDSC hierarchical model where there are multiple MDSCs, the interface between the top MDSC and the bottom MDSC (which is known as the MMI) can be mapped to service delivery models. See Section 4.5 for detailed discussions.

4.3. Network Configuration Models in ACTN Architecture (MPI)

The Network Configuration Models is used between the network orchestrator and the controller in [Service-YANG]. In ACTN, this model is used primarily between a MDSC and a PNC. The Network Configuration Model can be also used for the foundation of more advanced models, like hierarchical MDSCs (see Section 4.5)

The Network Configuration Model captures the parameters which are network wide information.

The following table provides a list of functions needed to build the MPI. They are mapped with Network Configuration Yang Models. Note that various Yang models are work in progress.

Function	Yang Model
Configuration Scheduling	[Schedule]
Path computation	[PATH_COMPUTATION-API]*
Path Provisioning	[TE-Tunnel]**
Topology Abstraction	[TE-topology]
Tunnel PM Telemetry	[ACTN-PM-Telemetry]***
Service Provisioning	TBD****
OTN Topology Abstraction	[OTN-YANG]
WSON Topology Abstraction	[WSON-YANG]
Flexi-grid Topology Abstraction	[Flexi-YANG]
ODU Tunnel Model	[ODU-Tunnel]
Other Tech specific Tunnel Model	TBD

* Related draft is presenting use cases for path computation API, and Yang related model is foreseen to be added.

** Note that path provisioning function is provided by ietf-te module in [TE-Tunnel].

** ietf-actn-te-kpi-telemetry model describes performance telemetry for TE-tunnel model. This module also allows autonomic traffic engineering scaling intent configuration mechanism on the TE-tunnel level. Various conditions can be set for auto-scaling based on the telemetry data.

**** This function needs to be investigated further. This can be a part of [TE-Tunnel] which is to be determined. Service provisioning is an optional function that builds on top the path provisioning one.

Path provisioning and Topology abstraction functions are mandatory in any case, while Path Computation may be mandatory or optional depending on the type of topology abstraction used. Details of this topic are discussed in [ACTN-Abstraction].

Telemetry may also be an optional function.

4.4. Device Models in ACTN Architecture (SBI)

For the device YANG models are used for per-device configuration purpose, they can be used between the PNC and the physical network/devices. Note that SBI is not in the scope of ACTN. This section is provided to give some examples of YANG-based Device Models. An example of Device Models is ietf-te-device yang module defined in [TE-tunnel].

4.5. Advanced Models

There may be some variation of interface C whereby there may be MDSC-MDSC interface (MMI) in case where there may be a recursive hierarchy among the MDSCs. This is a variation of ACTN architecture. See Figure 4 for this.

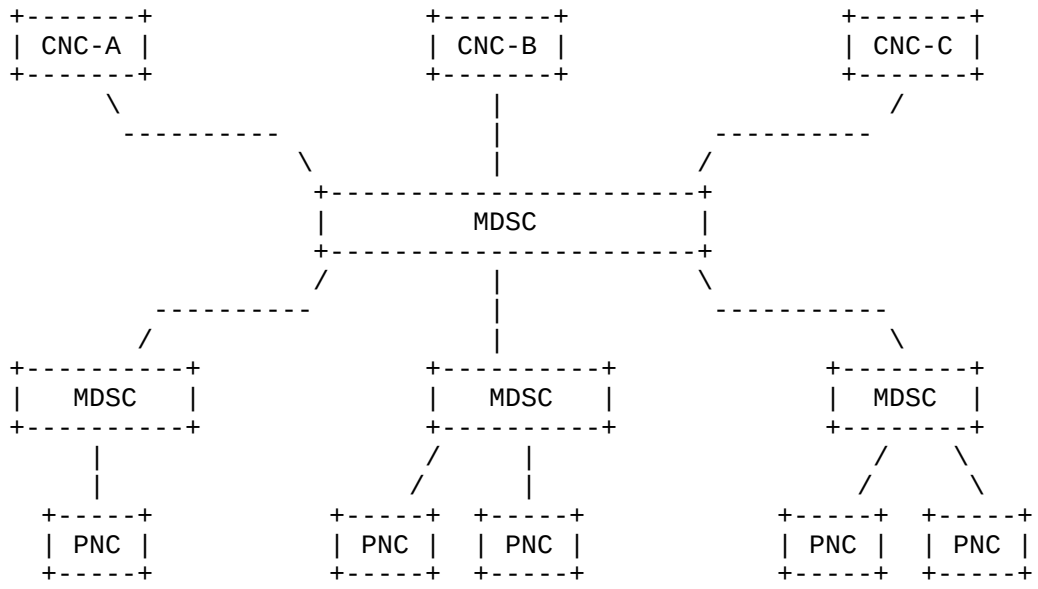


Figure 4: MDSC Controller Hierarchy

The MDSC-MDSC interface can be viewed as a recursive network configuration model which is similar to the MDSC-PNC interface.

On the other hand, in some implementations, the MMI can be viewed as the interface that fulfills the Service Delivery Model (See Figure 3). In this case, the top MDSC assumes service orchestration functions while the lower MDSC network orchestration functions.

5. Examples of Using Different Types of YANG Models

5.1. Simple Connectivity Examples

The data model in [Transport-Service-Model] provides an intent-like connectivity service model which can be used in connection-oriented networks.

It would be used as follows in the ACTN architecture:

. A CNC uses this service model to specify the two client nodes that are to be connected, and also indicates the amount of traffic (i.e., the bandwidth required) and payload type. What

may be additionally specified is the SLA that describes the required quality and resilience of the service.

. The MDSC uses the information in the request to pick the right network (domain) and also to select the provider edge nodes corresponding to the customer edge nodes.

  If there are multiple domains, then the MDSC needs to coordinate across domains to set up network tunnels to deliver a service. Thus coordination includes, but is not limited to, picking the right domain sequence to deliver a service. Before it can perform such functions, it needs to get the topology information from each PNC, using topology YANG models such as [te-topology]. The topology reported from PNC to MDSC can either be abstract or non-abstract.

  Additionally, an MDSC can initiate the creation of a tunnel (or tunnel segment) in order to fulfill the service request from CNC based on path computation upon the overall topology information it synthesized from different PNCs. The based model that can cater this purpose is the te-tunnel model specified in [te-tunnel].

. Then, the PNC needs to decide the explicit route of such a tunnel or tunnel segment (in case of multiple domains), and create such a tunnel using protocols such as PCEP and RSVP-TE or using per-hop configuration.

5.2. VN service example

The service model defined in [ACTN-VN-YANG] describes a virtual network (VN) as a service which is a set of multiple connectivity services:

. A CNC will request VN to the MDSC by specifying a list of VN members. Each VN member specifies either a single connectivity service, or a source with multiple potential destination points in the case that the precise destination sites are to be determined by MDSC.

  o In the first case, the procedure is the same as the connectivity service, except that in this case, there is a list of connections requested.

  o In the second case, where the CNC requests the MDSC to select the right destination out of a list of candidates, the MDSC needs to choose the best candidate and reply with

the chosen destination for a given VN member. After this is selected, the connectivity request setup procedure is the same as in the connectivity-as-a-service example.

After the VN is set up, a successful reply message is sent from MDSC to CNC, indicating the VN is ready. This message can also be achieved by using the model defined in [ACTN-VN-YANG].

5.3. Data Center-Interconnection Example

This section describes more concretely how existing YANG models described in Section 4 map to an ACTN data center interconnection use case. Figure 5 shows a use-case which shows service policy-driven Data Center selection and is a reproduction of Figure A.1 from [ACTN-Info].

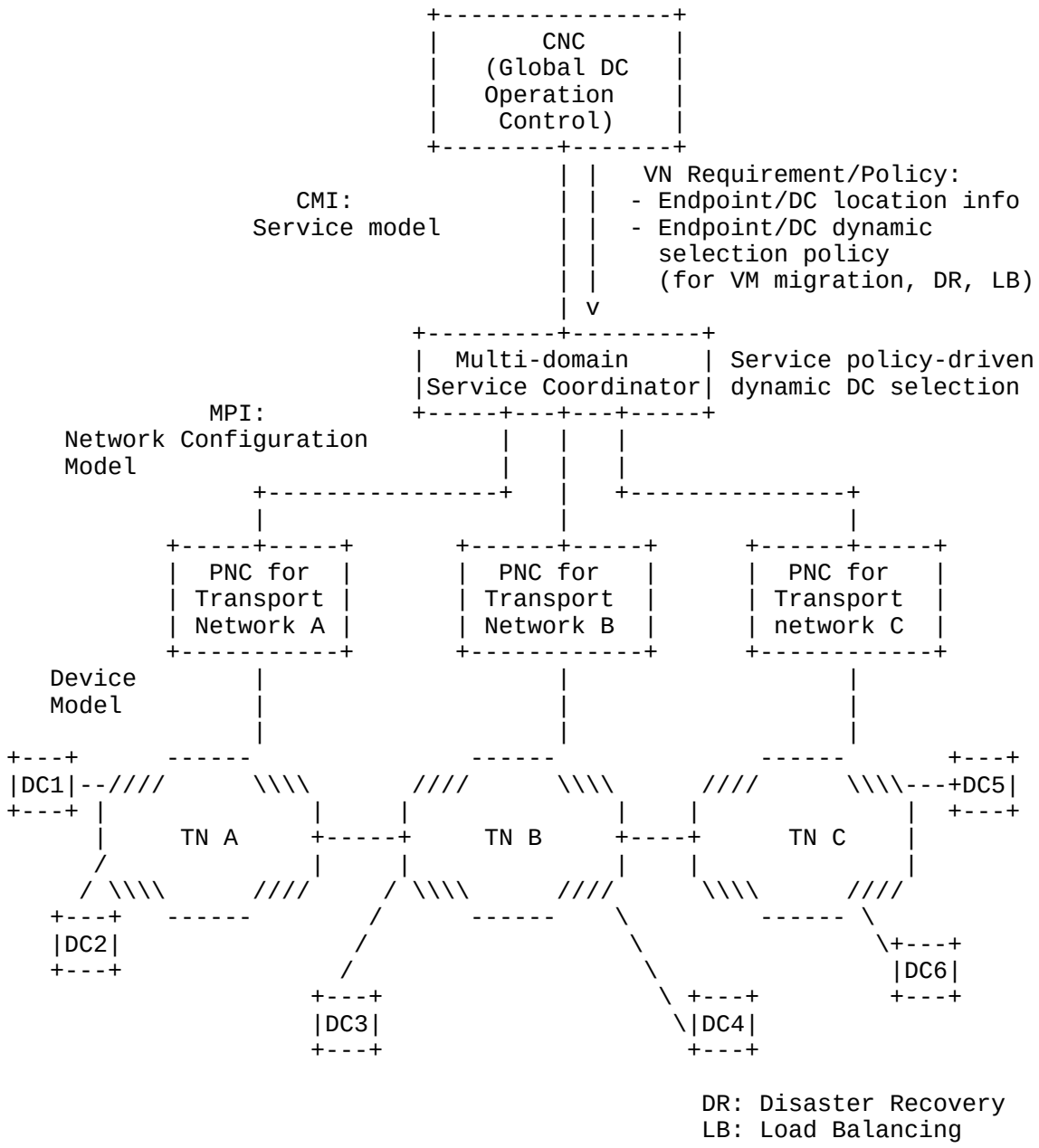


Figure 5: Service Policy-driven Data Center Selection

Figure 5 shows how VN policies from the CNC (Global Data Center Operation) 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 policy plays an important role for virtual network operation. Policy can be static or dynamic. Dynamic policy for data center selection may be placed as a result of utilization of data center resources supporting VMs. The MDSC would then incorporate this information to meet the objective of this application.

5.3.1. CMI (CNC-MDSC Interface)

[ACTN-VN-YANG] is used to express the definition of a VN, its VN creation request, the service objectives (metrics, QoS parameters, etc.), dynamic service policy when VM needs to be moved from one Data Center to another Data Center, etc. This service model is used between the CNC and the MDSC (CMI). The CNC in this use-case is an external entity that wants to create a VN and operates on the VN.

5.3.2. MPI (MDSC-PNC Interface)

The Network Configuration Model is used between the MDSC and the PNCs. Based on the Customer Service Model's request, the MDSC will need to translate the service model into the network configuration model to instantiate a set of multi-domain connections between the prescribed sources and the destinations. The MDSC will also need to dynamically interact with the CNC for dynamic policy changes initiated by the CNC. Upon the determination of the multi-domain connections, the MDSC will need to use the network configuration model such as [TE-Tunnel] to interact with each PNC involved on the path. [TE-Topology] is used to for the purpose of underlying domain network abstraction from the PNC to the MDSC.

5.3.3. PDI (PNC-Device interface)

The Device Model can be used between the PNC and its underlying devices that are controlled by the PNC. The PNC will need to trigger signaling using any mechanisms it employees (e.g. [RSVP-TE-YANG]) to provision its domain path segment. There can be a plethora of choices how to control/manage its domain network. The PNC is responsible to abstract its domain network resources and update it

to the MDSC. Note that this interface is not in the scope of ACTN. This section is provided just for an illustration purpose.

6. Security

This document is an informational draft. When the models mentioned in this draft are implemented, detailed security consideration will be given in such work.

How security fits into the whole architecture has the following components:

- the use of Restconf security between components

- the use of authentication and policy to govern which services can be requested by different parties.

- how security may be requested as an element of a service and mapped down to protocol security mechanisms as well as separation (slicing) of physical resources)

7. Acknowledgements

We thank Adrian Farrel for providing useful comments and suggestions for this draft.

8. References

8.1. Informative References

[Service-YANG] Q. Wu, W. Liu and A. Farrel, "Service Models Explained", draft-wu-opsawg-service-model-explained, work in progress.

[Netmod-Yang-Model-Classification] D. Bogdanovic, B. Claise, and C. Moberg, "YANG Module Classification", draft-ietf-netmod-yang-model-classification, work in progress.

[Netconf] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241.

[Restconf] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf, work in progress.

[ACTN-Requirements] Y. Lee, et al., "ACTN Requirements for Abstraction and Control of TE Networks", draft-ietf-teas-actn-requirements, work in progress.

[ACTN-Frame] D. Cecarelli and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks", draft-ietf-teas-actn-framework, work in progress.

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

[TE-Tunnel] T. Saad (Editor), "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te, work in progress.

[ACTN-VN-YANG] Y. Lee (Editor), "A Yang Data Model for ACTN VN Operation", draft-lee-teas-actn-vn-yang, work in progress.

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

[Transport-Service-Model] X. Zhang (Editor), "A Service YANG Model for Connection-oriented Transport Networks", draft-zhang-teas-transport-service-model, work in progress.

[PATH-COMPUTATION-API] I.Busi/S.Belotti et al. "Path Computation API", draft-busibel-ccamp-path-computation-api-00.txt, work in progress

[RSVP-TE-YANG] T. Saad (Editor), "A YANG Data Model for Resource Reservation Protocol (RSVP)", draft-ietf-teas-yang-rsvp, work in progress.

[Schedule] X. Liu, et. al., "A YANG Data Model for Configuration Scheduling", draft-liu-netmod-yang-schedule, work in progress.

[ACTN-Abstraction] Y. Lee, D. Dhody, and D. Ceccarelli, "ACTN Abstraction Methods", draft-lee-tease-actn-abstraction, work in progress.

[OTN-YANG] X. Zhang, A. Sharma, and X. Liu, "A YANG Data Model for Optical Transport Network Topology", draft-zhang-ccamp-l1-topo-yang, work in progress.

[WSON-YANG] Y. Lee, et. al., "A Yang Data Model for WSON Optical Networks", draft-ietf-ccamp-wson-yang, work in progress.

[Flexi-YANG] J.E. Lopez de Vergara, et. al., "YANG data model for Flexi-Grid Optical Networks", draft-vergara-ccamp-flexigrid-yang, work in progress.[ODU-Tunnel] Sharma, R. Rao, and X. Zhang, "OTN Service YANG Model", draft-sharma-ccamp-otn-service-model, work in progress.

[ACTN-PM-Telemetry]

9. Contributors

Contributor's Addresses

Dhruv Dhody
Huawei Technologies

Email: dhruv.ietf@gmail.com

Xian Zhang
Huawei Technologies

Email: zhang.xian@huawei.com

Authors' Addresses

Young Lee
Huawei Technologies
5340 Legacy Drive
Plano, TX 75023, USA
Phone: (469)277-5838

Email: leeyoung@huawei.com

Haomian Zheng
Huawei Technologies

Email: zhenghaomian@huawei.com

Daniele Ceccarelli
Ericsson
Torshamnsgatan,48
Stockholm, Sweden

Email: daniele.ceccarelli@ericsson.com

Bin Yeong Yoon
ETRI

Email: byyun@etri.re.kr

Oscar Gonzalez de Dios
Telefonica

Email: oscar.gonzalezdedios@telefonica.com

Jong Yoon Shin
SKT

Email: jongyoon.shin@sk.com

Sergio Belotti
Nokia

Email: sergio.belotti@nokia.com