Stream: Internet Engineering Task Force (IETF)
RFC: 9731
Category: Standards Track
Published: January 2025
ISSN: 2070-1721
Authors: Y. Lee, Ed. *Samsung Electronics*; D. Dhody, Ed. *Huawei*; D. Ceccarelli *Cisco*; I. Bryskin *Individual*; B. Yoon *ETRI*

RFC 9731
A YANG Data Model for Virtual Network (VN) Operations

Abstract

A Virtual Network (VN) is a network provided by a service provider to a customer for the customer to use in any way it wants as though it were a physical network. This document provides a YANG data model generally applicable to any mode of VN operations. This includes VN operations as per the Abstraction and Control of TE Networks (ACTN) framework.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9731.

Copyright Notice

Table of Contents

1. Introduction

Abstraction and Control of TE Networks (ACTN) describes a set of management and control functions used to operate one or more Traffic Engineered (TE) networks to construct a Virtual Network (VN). A VN is represented to customers and is built from the abstractions of the underlying TE networks [RFC8453]. This document provides a YANG data model [RFC7950] generally applicable to any mode of VN operation. ACTN is the primary example of the usage of the VN YANG model, but VN is not limited to it.

The VN model defined in this document is applicable in a generic sense as an independent model in and of itself. It can also work together with other customer service models such as the L3VPN Service Model (L3SM) [RFC8299], the L2VPN Service Model (L2SM) [RFC8466], and the L1 Connectivity Service Model (L1CSM) [L1CSM-YANG] to provide complete life-cycle service management and operations.

The YANG model discussed in this document basically provides the following:

- Characteristics of Access Points (APs) that describe customer's endpoint characteristics;
- Characteristics of Virtual Network Access Points (VNAPs) that describe how an AP is partitioned for multiple VNs sharing the AP and its reference to a Link Termination Point (LTP) of the Provider Edge (PE) node;
- Characteristics of Virtual Networks (VNs) that describe the customer's VN in terms of multiple VN Members comprising a VN, multi-source and/or multi-destination characteristics of the VN Member, the VN's reference to TE-topology's Abstract Node;

An abstract TE topology is a topology that contains abstract topological elements (nodes, links) created and customized based on customer preference [RFC8795]. The actual VN instantiation and computation is performed with Connectivity Matrices of the TE-Topology Model [RFC8795], which provides a TE network topology abstraction and management operation. As per [RFC8795], a TE node connectivity matrix is the TE node's switching limitations in the form of valid switching combinations of the TE node's LTPs and potential TE paths. The VN representation relies on a single abstract TE node with a connectivity matrix. The VN can be abstracted as a set of edge-to-edge links (a Type 1 VN). Each link is the VN member that is

mapped to the connectivity matrix entry (Section 2.1). The VN can also be abstracted as a topology of virtual nodes and virtual links (a Type 2 VN). Alongside the mapping of VN members to a connectivity matrix entry, an underlay path can also be specified (Section 2.2).

Once the TE-topology Model is used in triggering VN instantiation over the networks, the TE-tunnel Model [YANG-TE] will inevitably interact with the TE-Topology model when setting up actual tunnels and Label Switched Paths (LSPs) under the tunnels.

Sections 2 and 3 provide a discussion of how the VN YANG model is applicable to the ACTN context where a Virtual Network Service (VNS) operation is implemented for the interface of the Customer Network Controller (CNC) and the Multi-Domain Service Coordinator (MDSC).

The YANG model for the CNC-MDSC Interface (CMI) is also known as the "customer service model" in [RFC8309]. The YANG model discussed in this document is used to operate customer-driven VNs during the VN instantiation and computation as well as its life-cycle service management and operations.

The VN operational state is included in the same tree as the configuration consistent with Network Management Datastore Architecture (NMDA) [RFC8342].

1.1. Terminology

This document borrows the following terms from [RFC8453]:

VN: Virtual Network

AP: Access Point

VNAP: VN Access Point

ACTN: Abstraction and Control of TE Networks

CNC: Customer Network Controller

MDSC: Multi-Domain Service Coordinator

CMI: CNC-MDSC Interface

This document borrows the following terms from [RFC8795]:

LTP: Link Termination Point

Connectivity Matrix

This document borrows the terminology in Section 1.1 of [RFC7926].

This document uses the term 'Service Model' as described in [RFC8309].

1.2. Tree Diagram

A simplified graphical representation of the data model is used in Section 5 of this document. The meaning of the symbols in these diagrams is defined in [RFC8340].

1.3. Prefixes in Data Node Names

In this document, the 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
vn	ietf-vn	RFC 9731
yang	ietf-yang-types	[RFC6991]
nw	ietf-network	[RFC8345]
nt	ietf-network-topology	[RFC8345]
te-types	ietf-te-types	[RFC8776]
tet	ietf-te-topology	[RFC8795]

Table 1: Prefixes and Corresponding YANG Modules

2. Use Case of VN YANG Model in the ACTN Context

In this section, ACTN is being used to illustrate the general usage of the VN YANG model. The model presented in this section has the following ACTN context.

Figure 1: ACTN CMI

Both ACTN VN YANG and TE-topology models are used over the CMI to establish a VN over TE networks, as shown in Figure 1.

2.1. Type 1 VN

As defined in [RFC8453], a Virtual Network is a customer view of the TE network. To recapitulate VN types from [RFC8453], a Type 1 VN is defined as follows:

> The VN can be seen as a set of edge-to-edge abstract links (a Type 1 VN). Each abstract link is referred to as a VN member and is formed as an end-to-end tunnel across the underlying networks. Such tunnels may be constructed by recursive slicing or abstraction of paths in the underlying networks and can encompass edge points of the customer's network, access links, intra-domain paths, and inter-domain links.

If we were to create a VN where we have four VN-members as follows:

```
VN-member 1       L1-L4
VN-member 2       L1-L7
VN-member 3       L2-L4
VN-member 4       L3-L8
```

Figure 2

Where L1, L2, L3, L4, L7, and L8 correspond to a Customer Endpoint (or AP).

This VN can be modeled as one abstract node representation as follows in Figure 3:

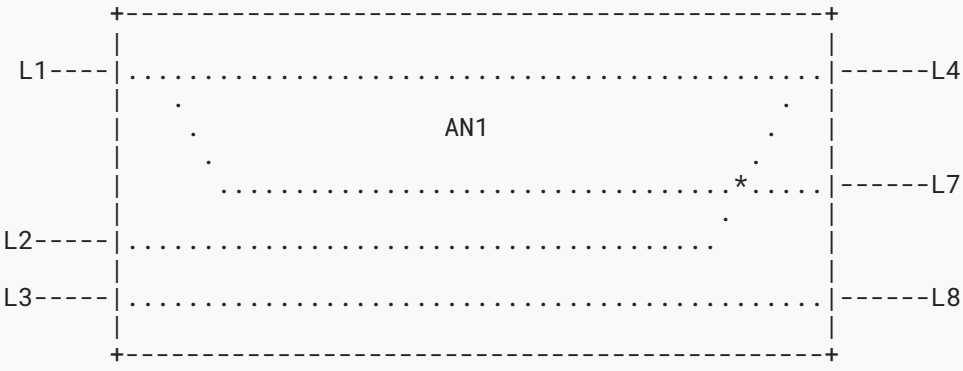

Figure 3: Abstract Node (One Node Topology)

Modeling a VN as one abstract node is the easiest way for customers to express their end-to-end connectivity as shown in Figure 3.

2.2. Type 2 VN

For some VN members, the customers are allowed to configure the intended path. To achieve this, alongside the single node abstract topology, an underlay topology is also needed. The underlay topology could be native TE topology or an abstract TE topology. The intended path is set based on the nodes and links of the underlay topology. A Type 1 VN can be seen as a higher abstraction of a Type 2 VN (which along with a single node abstract topology, an underlay topology and the intended path are specified). These topologies could be mutually agreed upon between the CNC and the MDSC prior to VN creation or they could be created as part of VN instantiation.

If a Type 2 VN is desired for some or all of the VN members of a Type 1 VN (see the example in Section 2.1), the TE-topology model can provide the following abstract topologies (a single node topology AN1 and an underlay topology (with nodes S1 to S11 and corresponding links)).

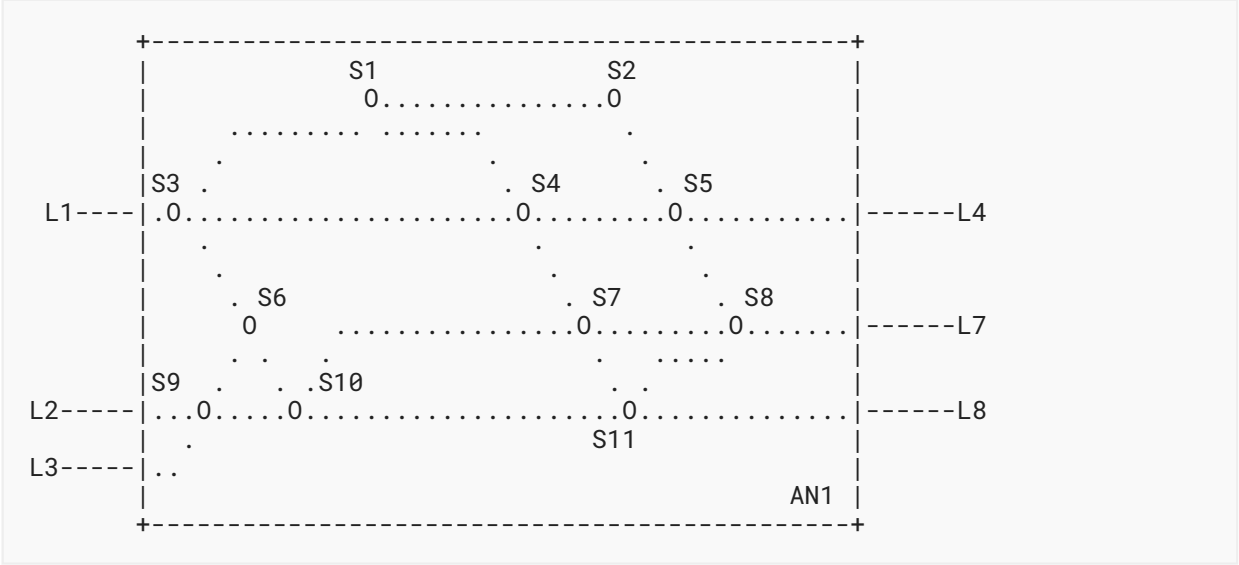

Figure 4: Type 2 Topology

As shown in Figure 4, the abstract node is AN1 and an underlay topology is depicted with nodes and links (S1 to S11).

As an example, if VN-member 1 (L1-L4) is chosen to configure its own path over Type 2 topology, it can select, say, a path that consists of the explicit abstract path {S3,S4,S5} based on the underlay topology and its service requirement. This capability is enacted via TE-topology configuration by the customer.

3. High-Level Control Flows with Examples

3.1. Type 1 VN Illustration

If this VN is Type 1, the following diagram shows the message flow between CNC and MDSC to instantiate this VN using VN and TE-Topology Models.

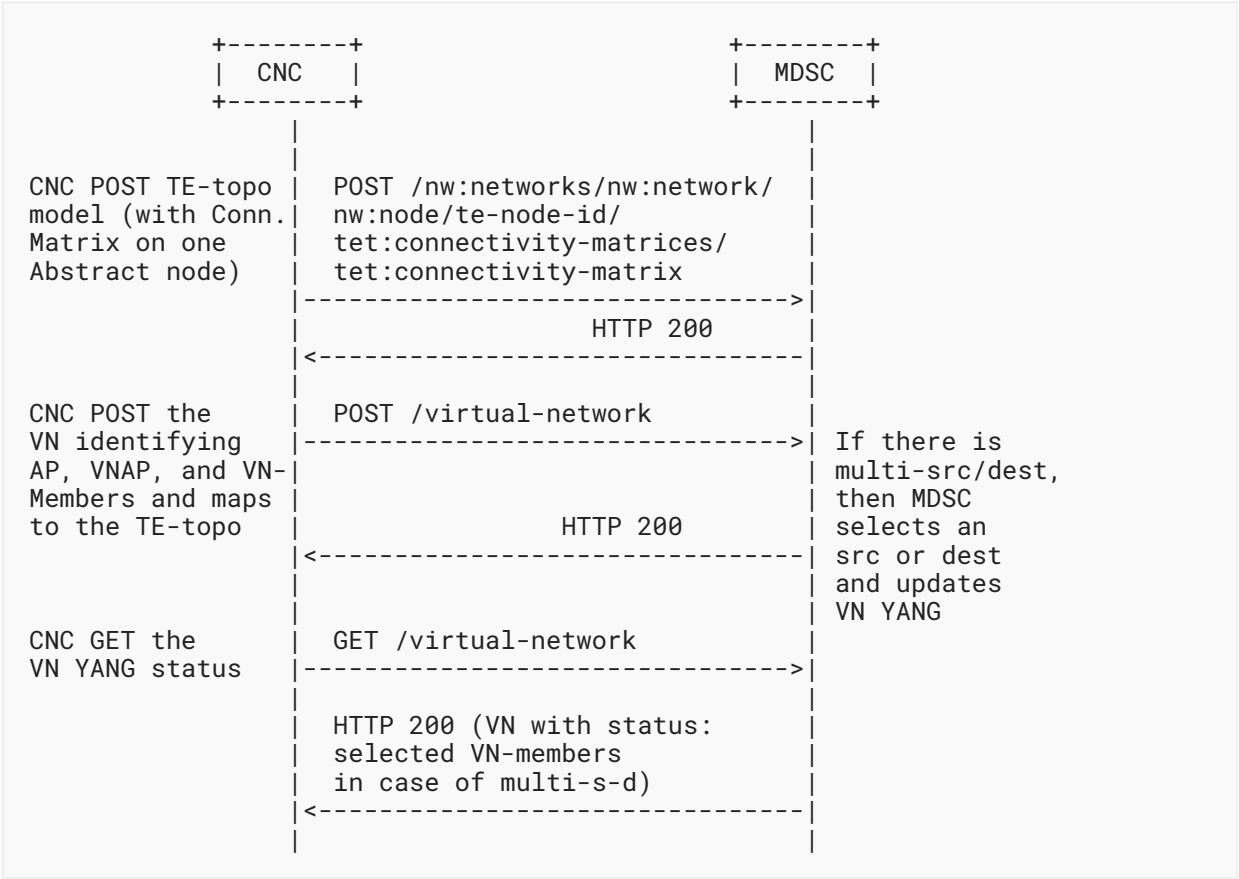

Figure 5: Type 1 VN Illustration

3.2. Type 2 VN Illustration

For some VN members, the customer may want to "configure" an explicit path that connects its two endpoints. Let us consider the following example:

```
VN-member 1     L1-L4 (via S3, S4, and S5)
VN-member 2     L1-L7 (via S3, S4, S7, and S8)
VN-member 3     L2-L7 (via S9, S10, and S11)
VN-member 4     L3-L8 (via S9, S10, and S11)
```

Figure 6

There are two options depending on whether CNC or MDSC creates the single abstract node topology.

Case 1:

If the CNC creates the single-abstract-node topology, the message flow between the CNC and MDSC to instantiate this VN using a VN and TE-Topology Model would be as shown in the following diagram:

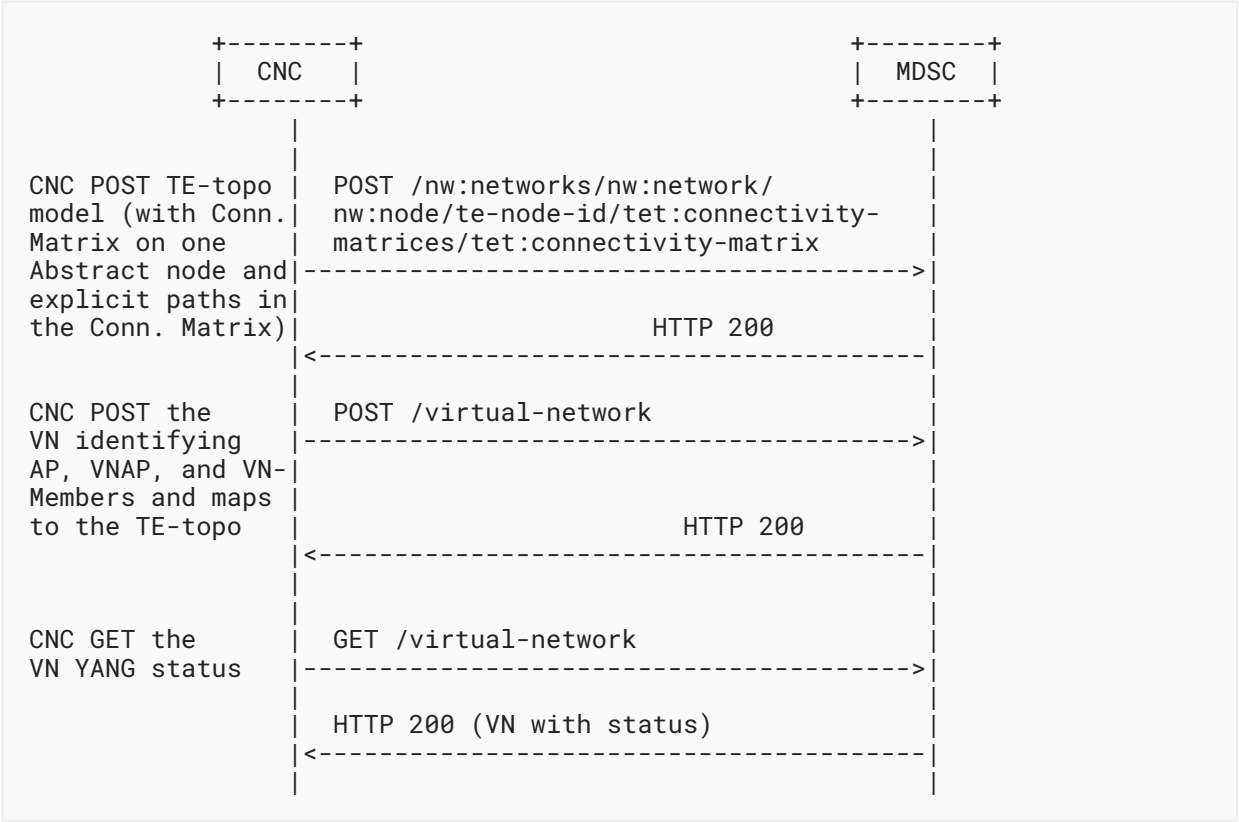

Figure 7: Type 2 VN Illustration: Case 1

Case 2:

On the other hand, if MDSC create the single-abstract-node topology based on VN YANG posted by the CNC, the following diagram shows the message flow between CNC and MDSC to instantiate this VN using VN and TE-Topology Models.

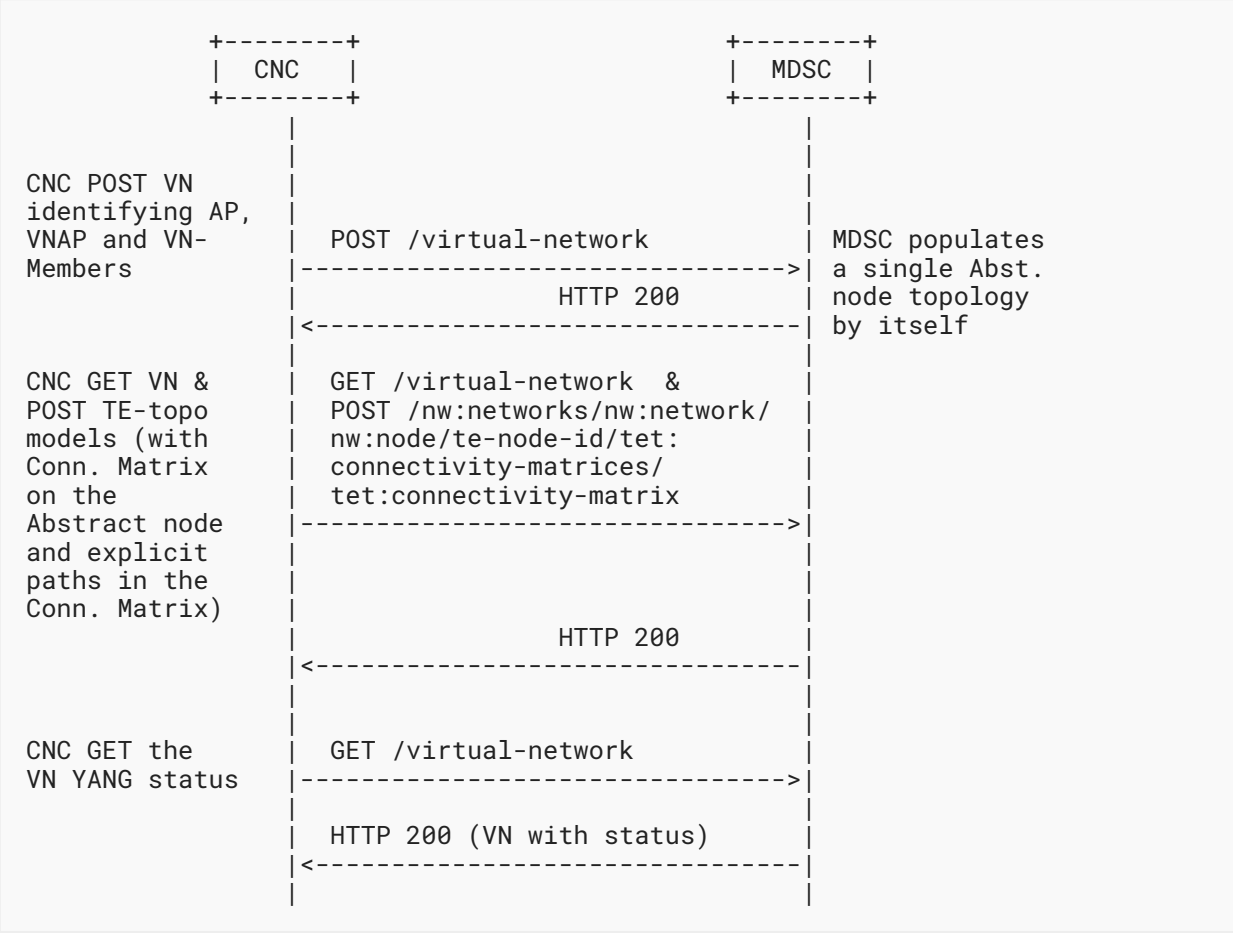

Figure 8: Type 2 VN Illustration: Case 2

Note that the underlay topology (which is referred to by the single-abstract-node topology) could be a Native/White topology or a Grey topology ([RFC8453]) that is further customized based on the requirements of the customer and configured at the MDSC.

Appendix B provides JSON examples for both the VN model and the TE-topology Connectivity Matrix sub-model to illustrate how a VN can be created by the CNC making use of the VN module as well as the TE-topology Connectivity Matrix module.

3.2.1. VN and AP Usage

The customer access information may be known at the time of VN creation. A shared logical AP identifier is used between the customer and the operator to identify the access link between Customer Edge (CE) and Provider Edge (PE). This is described in Section 6 of [RFC8453].

In some VN operations, the customer access may not be known at the initial VN creation. The VN operation allows the creation of a VN with only a PE identifier as well. The customer access information could be added later.

To achieve this, the 'ap' container has a leaf for the 'pe' node that allows the AP to be created with PE information. The vn-member (and vn) could use APs that initially only have PE information.

4. VN Model Usage

4.1. Customer View of VN

The VN-YANG model allows the definition of a customer view and allows the customer to communicate using the VN constructs as described in [RFC8454]. It allows the grouping of edge-to-edge links (i.e., VN members) under a common umbrella of VN. This allows the customer to instantiate and view the VN as one entity, making it easier for some customers to work on VN without worrying about the details of the provider-based YANG models.

This is similar to the benefits offered by a separate YANG model for customer services described in [RFC8309], which states that service models do not make any assumptions about how a service is actually engineered and delivered for a customer.

4.2. Auto-creation of VN by MDSC

The VN could be configured at the MDSC explicitly by the CNC using the VN YANG model. In some other cases, the VN is not explicitly configured but is instead created automatically by the MDSC based on the customer service model and local policy; even in these cases, the VN YANG model can be used by the CNC to learn details of the underlying VN, created to meet the requirements of the customer service model.

4.3. Innovative Services

4.3.1. VN Compute

The VN Model supports VN compute (pre-instantiation mode) to view the full VN as a single entity before instantiation; achieving this via path computation or "compute only" tunnel setup ([YANG-TE]) does not provide the same functionality.

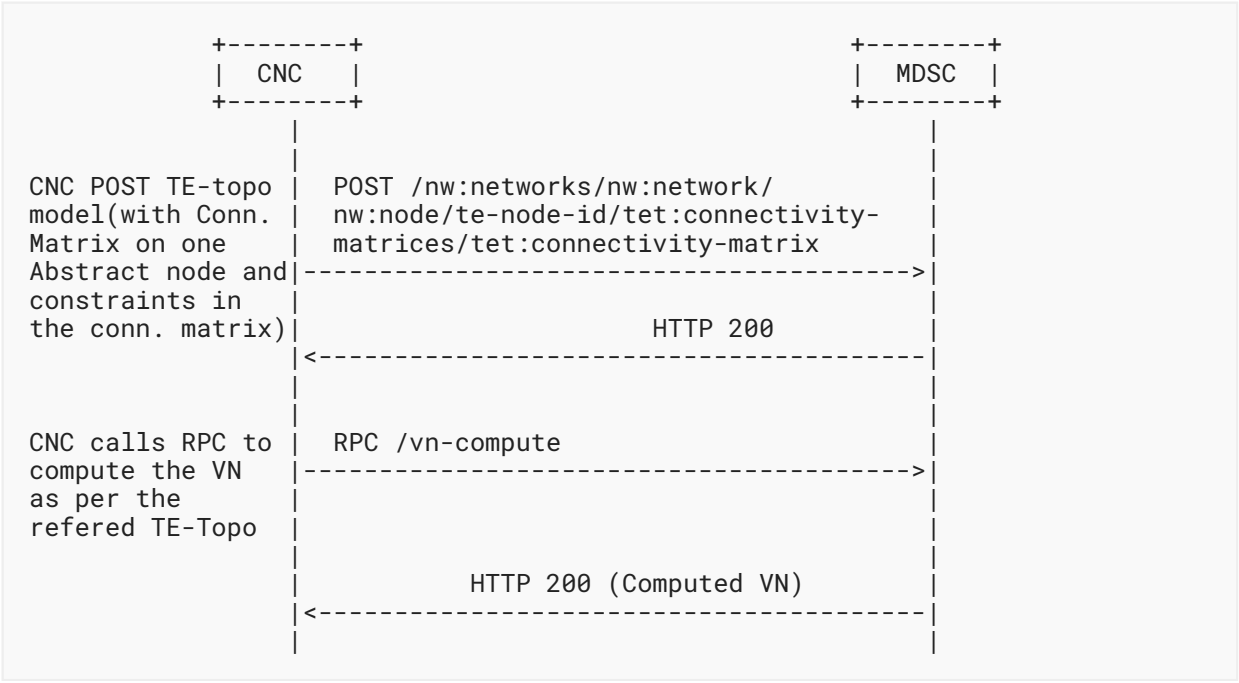

Figure 9: VN Compute

The VN compute RPC allows the optional inclusion of the constraints and the optimization criteria at the VN as well as at the individual VN-member level. Thus, the RPC can be used independently to get the computed VN result without creating an abstract topology first.

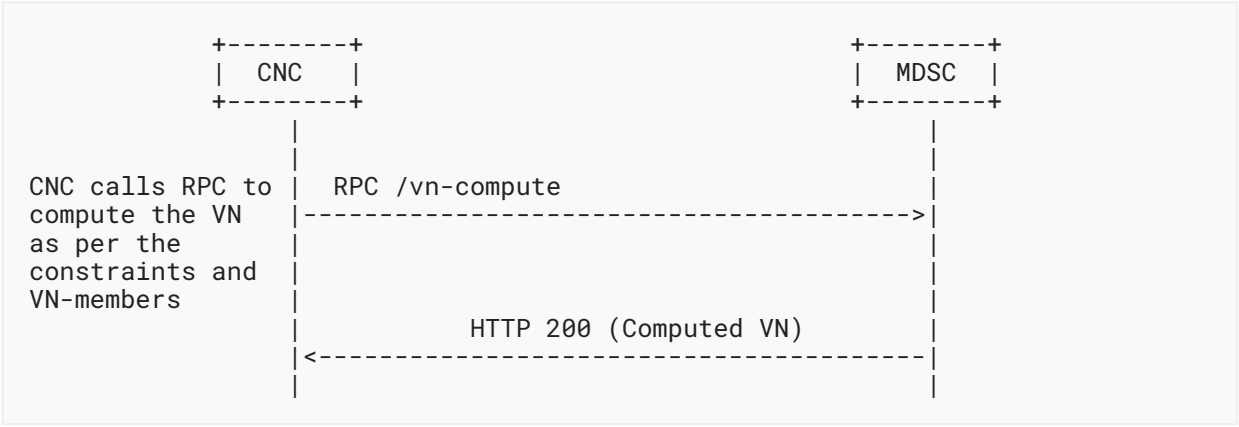

Figure 10: VN Compute

In either case, the output includes a reference to the single node abstract topology with each VN-member including a reference to the connectivity-matrix-id where the path properties could be found.

To achieve this, the VN-compute RPC reuses the following common groupings:

- te-types:generic-path-constraints: This is used optionally in the RPC input at the VN and/or VN-member level. The VN-member level overrides the VN-level data. This also overrides any constraints in the referenced abstract node in the TE topology.
- te-types:generic-path-optimization: This is used optionally in the RPC input at the VN and/or VN-member level. The VN-member level overrides the VN-level data. This also overrides any optimization in the referenced abstract node in the TE topology.
- vn-member: This identifies the VN member in both RPC input and output.
- vn-policy: This is used optionally in the RPC input to apply any VN-level policies.

When MDSC receives this RPC, it computes the VN based on the input provided in the RPC. This computation does not create a VN or reserve any resources in the system, it simply computes the resulting VN based on information at the MDSC or in coordination with the CNC. A single-node-abstract topology is used to convey the result of each VN member as a reference to the connectivity-matrix-id. In case of an error, the error information is included.

```
rpcs:
  +---x vn-compute
     +---w input
     |  +---w te-topology-identifier
     |  |  +---w provider-id?   te-global-id
     |  |  +---w client-id?     te-global-id
     |  |  +---w topology-id?   te-topology-id
     |  +---w abstract-node?
     |  |       -> /nw:networks/network/node/tet:te-node-id
     |  +---w path-constraints
     |  |  +---w te-bandwidth
     |  |  |  +---w (technology)?
     |  |  |        ...
     |  |  +---w link-protection?          identityref
     |  |  +---w setup-priority?           uint8
     |  |  +---w hold-priority?            uint8
     |  |  +---w signaling-type?           identityref
     |  |  +---w path-metric-bounds
     |  |  |  +---w path-metric-bound* [metric-type]
     |  |  |        ...
     |  |  +---w path-affinities-values
     |  |  |  +---w path-affinities-value* [usage]
     |  |  |        ...
     |  |  +---w path-affinity-names
     |  |  |  +---w path-affinity-name* [usage]
     |  |  |        ...
     |  |  +---w path-srlgs-lists
     |  |  |  +---w path-srlgs-list* [usage]
     |  |  |        ...
     |  |  +---w path-srlgs-names
     |  |  |  +---w path-srlgs-name* [usage]
     |  |  |        ...
     |  |  +---w disjointness?            te-path-disjointness
     |  +---w cos?                     te-types:te-ds-class
     |  +---w optimizations
     |  |  +---w (algorithm)?
```

```
      |  |     +--:(metric) {path-optimization-metric}?
      |  |     |     ...
      |  |     +--:(objective-function)
      |  |              {path-optimization-objective-function}?
      |  |           ...
      |  +---w vn-member-list* [id]
      |  |  +---w id                          vnm-id
      |  |  +---w src
      |  |  |  +---w ap?          -> /access-point/ap/id
      |  |  |  +---w vn-ap-id?
      |  |  |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
      |  |  |  +---w multi-src?   boolean {multi-src-dest}?
      |  |  +---w dest
      |  |  |  +---w ap?           -> /access-point/ap/id
      |  |  |  +---w vn-ap-id?
      |  |  |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
      |  |  |  +---w multi-dest?   boolean {multi-src-dest}?
      |  |  +---w connectivity-matrix-id?   leafref
      |  |  +---w underlay
      |  |  +---w path-constraints
      |  |  |  +---w te-bandwidth
      |  |  |  |     ...
      |  |  |  +---w link-protection?          identityref
      |  |  |  +---w setup-priority?           uint8
      |  |  |  +---w hold-priority?            uint8
      |  |  |  +---w signaling-type?           identityref
      |  |  |  +---w path-metric-bounds
      |  |  |  |     ...
      |  |  |  +---w path-affinities-values
      |  |  |  |     ...
      |  |  |  +---w path-affinity-names
      |  |  |  |     ...
      |  |  |  +---w path-srlgs-lists
      |  |  |  |     ...
      |  |  |  +---w path-srlgs-names
      |  |  |  |     ...
      |  |  |  +---w disjointness?            te-path-disjointness
      |  |  +---w cos?                      te-types:te-ds-class
      |  |  +---w optimizations
      |  |     +---w (algorithm)?
      |  |           ...
      |  +---w vn-level-diversity?       te-types:te-path-disjointness
      +--ro output
         +--ro te-topology-identifier
         |  +--ro provider-id?   te-global-id
         |  +--ro client-id?     te-global-id
         |  +--ro topology-id?   te-topology-id
         +--ro abstract-node?
         |       -> /nw:networks/network/node/tet:te-node-id
         +--ro vn-member-list* [id]
            +--ro id                          vnm-id
            +--ro src
            |  +--ro ap?          -> /access-point/ap/id
            |  +--ro vn-ap-id?
            |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
            |  +--ro multi-src?   boolean {multi-src-dest}?
            +--ro dest
            |  +--ro ap?           -> /access-point/ap/id
```

```
           |  +--ro vn-ap-id?
           |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
           |  +--ro multi-dest?   boolean {multi-src-dest}?
           +--ro connectivity-matrix-id?   leafref
           +--ro underlay
           +--ro if-selected?              boolean {multi-src-dest}?
           +--ro compute-status?           vn-compute-status
           +--ro error-info
              +--ro error-description?   string
              +--ro error-timestamp?     yang:date-and-time
              +--ro error-reason?        identityref
```

4.3.2. Multiple Sources and Multiple Destinations

In creating a virtual network, the list of sources or destinations or both may not be predetermined 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 tree shows how to model multiple sources and multiple destinations.

```
module: ietf-vn
  +--rw virtual-network
     +--rw vn* [id]
        +--rw id                          vn-id
        +--rw te-topology-identifier
        |  +--rw provider-id?   te-global-id
        |  +--rw client-id?     te-global-id
        |  +--rw topology-id?   te-topology-id
        +--rw abstract-node?
        |       -> /nw:networks/network/node/tet:te-node-id
        +--rw vn-member* [id]
        |  +--rw id                         vnm-id
        |  +--rw src
        |  |  +--rw ap?          -> /access-point/ap/id
        |  |  +--rw vn-ap-id?
        |  |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
        |  |  +--rw multi-src?   boolean {multi-src-dest}?
        |  +--rw dest
        |  |  +--rw ap?           -> /access-point/ap/id
        |  |  +--rw vn-ap-id?
        |  |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
        |  |  +--rw multi-dest?   boolean {multi-src-dest}?
        |  +--rw connectivity-matrix-id?   leafref
        |  +--rw underlay
        |  +--ro oper-status?              te-types:te-oper-status
        |  +--ro if-selected?              boolean {multi-src-dest}?
        +--rw admin-status?             te-types:te-admin-status
        +--ro oper-status?              te-types:te-oper-status
        +--rw vn-level-diversity?       te-types:te-path-disjointness
```

4.4. Others

The VN YANG model can easily be augmented to support the mapping of VN to the services such as L3SM and L2SM as described in [TE-SERVICE-MAPPING].

The VN YANG model can be extended to support telemetry, performance monitoring, and network autonomics as described in [TEAS-ACTN-PM].

Note that the YANG model is tightly coupled with the TE Topology model [RFC8795]. Any underlay technology not supported by [RFC8795] is also not supported by this model. The model does include an empty container called "underlay" that can be augmented. For example the Segment Routing (SR) Policy [RFC9256] information can be augmented for the SR underlay by a future model.

Apart from the te-types:generic-path-constraints and te-types:generic-path-optimization, an additional leaf called "cos" for the class of service [RFC4124] is added to represent the Class-Type of traffic to be used as one of the path constraints.

4.5. Summary

This section summarizes the features of the VN YANG model.

- Maintenance of APs and VNAPs along with the VN
- VN construct to group of edge-to-edge links
- Ability to support various VN and VNS Types
  - VN Type 1: Customer configures the VN as a set of VN Members. No other details need to be set by the customer, making for a simplified operation for the customer.
  - VN Type 2: Along with VN Members, the customer could also provide an abstract topology, this topology is provided by the Abstract TE Topology YANG Model.
    - Note that the VN Type is not explicitly identified in the VN Yang model, as the VN Model is exactly the same for both VN Type 1 and VN Type 2. The VN type can be implicitly known based on the referenced TE topology and whether the connectivity matrix includes the underlay path (Type 2) or not (Type 1).
- VN Compute (pre-instantiate)
- Multi-Source / Multi-Destination

5. VN YANG Model (Tree Structure)

```
module: ietf-vn
  +--rw access-point
  |  +--rw ap* [id]
  |     +--rw id                ap-id
  |     +--rw pe?
  |     |       -> /nw:networks/network/node/tet:te-node-id
```

```
   |     +--rw max-bandwidth?   te-types:te-bandwidth
   |     +--rw avl-bandwidth?   te-types:te-bandwidth
   |     +--rw vn-ap* [id]
   |        +--rw id               ap-id
   |        +--rw vn?              -> /virtual-network/vn/id
   |        +--rw abstract-node?   -> /nw:networks/network/node/node-id
   |        +--rw ltp?             leafref
   |        +--ro max-bandwidth?   te-types:te-bandwidth
   +--rw virtual-network
      +--rw vn* [id]
         +--rw id                        vn-id
         +--rw te-topology-identifier
         |  +--rw provider-id?   te-global-id
         |  +--rw client-id?     te-global-id
         |  +--rw topology-id?   te-topology-id
         +--rw abstract-node?
         |       -> /nw:networks/network/node/tet:te-node-id
         +--rw vn-member* [id]
         |  +--rw id                        vnm-id
         |  +--rw src
         |  |  +--rw ap?          -> /access-point/ap/id
         |  |  +--rw vn-ap-id?
         |  |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
         |  |  +--rw multi-src?   boolean {multi-src-dest}?
         |  +--rw dest
         |  |  +--rw ap?           -> /access-point/ap/id
         |  |  +--rw vn-ap-id?
         |  |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
         |  |  +--rw multi-dest?   boolean {multi-src-dest}?
         |  +--rw connectivity-matrix-id?   leafref
         |  +--rw underlay
         |  +--ro oper-status?              te-types:te-oper-status
         |  +--ro if-selected?              boolean {multi-src-dest}?
         +--rw admin-status?             te-types:te-admin-status
         +--ro oper-status?              te-types:te-oper-status
         +--rw vn-level-diversity?       te-types:te-path-disjointness

   rpcs:
     +---x vn-compute
        +---w input
        |  +---w te-topology-identifier
        |  |  +---w provider-id?   te-global-id
        |  |  +---w client-id?     te-global-id
        |  |  +---w topology-id?   te-topology-id
        |  +---w abstract-node?
        |  |       -> /nw:networks/network/node/tet:te-node-id
        |  +---w path-constraints
        |  |  +---w te-bandwidth
        |  |  |  +---w (technology)?
        |  |  |        ...
        |  |  +---w link-protection?          identityref
        |  |  +---w setup-priority?           uint8
        |  |  +---w hold-priority?            uint8
        |  |  +---w signaling-type?           identityref
        |  |  +---w path-metric-bounds
        |  |  |  +---w path-metric-bound* [metric-type]
        |  |  |        ...
        |  |  +---w path-affinities-values
```

```
        |  |  |  +---w path-affinities-value* [usage]
        |  |  |        ...
        |  |  +---w path-affinity-names
        |  |  |  +---w path-affinity-name* [usage]
        |  |  |        ...
        |  |  +---w path-srlgs-lists
        |  |  |  +---w path-srlgs-list* [usage]
        |  |  |        ...
        |  |  +---w path-srlgs-names
        |  |  |  +---w path-srlgs-name* [usage]
        |  |  |        ...
        |  |  +---w disjointness?            te-path-disjointness
        |  +---w cos?                      te-types:te-ds-class
        |  +---w optimizations
        |  |  +---w (algorithm)?
        |  |     +--:(metric) {path-optimization-metric}?
        |  |     |     ...
        |  |     +--:(objective-function)
        |  |              {path-optimization-objective-function}?
        |  |           ...
        |  +---w vn-member-list* [id]
        |  |  +---w id                        vnm-id
        |  |  +---w src
        |  |  |  +---w ap?          -> /access-point/ap/id
        |  |  |  +---w vn-ap-id?
        |  |  |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
        |  |  |  +---w multi-src?   boolean {multi-src-dest}?
        |  |  +---w dest
        |  |  |  +---w ap?           -> /access-point/ap/id
        |  |  |  +---w vn-ap-id?
        |  |  |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
        |  |  |  +---w multi-dest?   boolean {multi-src-dest}?
        |  |  +---w connectivity-matrix-id?   leafref
        |  |  +---w underlay
        |  |  +---w path-constraints
        |  |  |  +---w te-bandwidth
        |  |  |  |     ...
        |  |  |  +---w link-protection?          identityref
        |  |  |  +---w setup-priority?           uint8
        |  |  |  +---w hold-priority?            uint8
        |  |  |  +---w signaling-type?           identityref
        |  |  |  +---w path-metric-bounds
        |  |  |  |     ...
        |  |  |  +---w path-affinities-values
        |  |  |  |     ...
        |  |  |  +---w path-affinity-names
        |  |  |  |     ...
        |  |  |  +---w path-srlgs-lists
        |  |  |  |     ...
        |  |  |  +---w path-srlgs-names
        |  |  |  |     ...
        |  |  |  +---w disjointness?            te-path-disjointness
        |  |  +---w cos?                      te-types:te-ds-class
        |  |  +---w optimizations
        |  |     +---w (algorithm)?
        |  |           ...
        |  +---w vn-level-diversity?       te-types:te-path-disjointness
        +--ro output
```

```
          +--ro te-topology-identifier
          |  +--ro provider-id?   te-global-id
          |  +--ro client-id?     te-global-id
          |  +--ro topology-id?   te-topology-id
          +--ro abstract-node?
          |       -> /nw:networks/network/node/tet:te-node-id
          +--ro vn-member-list* [id]
             +--ro id                          vnm-id
             +--ro src
             |  +--ro ap?          -> /access-point/ap/id
             |  +--ro vn-ap-id?
             |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
             |  +--ro multi-src?   boolean {multi-src-dest}?
             +--ro dest
             |  +--ro ap?           -> /access-point/ap/id
             |  +--ro vn-ap-id?
             |  |       -> /access-point/ap[id=current()/../ap]/vn-ap/id
             |  +--ro multi-dest?   boolean {multi-src-dest}?
             +--ro connectivity-matrix-id?   leafref
             +--ro underlay
             +--ro if-selected?              boolean {multi-src-dest}?
             +--ro compute-status?           vn-compute-status
             +--ro error-info
                +--ro error-description?   string
                +--ro error-timestamp?     yang:date-and-time
                +--ro error-reason?        identityref
```

6. VN YANG Model

The VN YANG model is as follows:

```
<CODE BEGINS> file "ietf-vn@2025-01-27.yang"

module ietf-vn {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-vn";
  prefix vn;

  /* Import common YANG types */

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  /* Import network */

  import ietf-network {
    prefix nw;
    reference
      "RFC 8345: A YANG Data Model for Network Topologies";
  }

  /* Import network topology */
```

```
import ietf-network-topology {
  prefix nt;
  reference
    "RFC 8345: A YANG Data Model for Network Topologies";
}

/* Import TE Common types */

import ietf-te-types {
  prefix te-types;
  reference
    "RFC 8776: Common YANG Data Types for Traffic Engineering";
}

/* Import TE Topology */

import ietf-te-topology {
  prefix tet;
  reference
    "RFC 8795: YANG Data Model for Traffic Engineering (TE)
     Topologies";
}

organization
  "IETF Traffic Engineering Architecture and Signaling (TEAS)
   Working Group";
contact
  "WG Web:  <https://datatracker.ietf.org/wg/teas/>
   WG List:  <mailto:teas@ietf.org>

   Editor:  Young Lee <younglee.tx@gmail.com>
   Editor:  Dhruv Dhody <dhruv.ietf@gmail.com>";
description
  "This module contains a YANG module for the Virtual Network
   (VN).  It describes a VN operation module that can take place
   in the context of the Customer Network Controller (CNC) -
   Multi-Domain Service Coordinator (MDSC) interface (CMI) of
   the Abstraction and Control of TE Networks (ACTN)
   architecture where the CNC is the actor of a VN
   instantiation/modification/deletion as per RFC 8453.

   This module uses the following abbreviations:
   - VN: Virtual Network
   - AP: Access Point
   - VNAP: Virtual Network Access Point
   - LTP: Link Termination Point
   - PE: Provider Edge
   - COS: Class of Service

   Further, src and dest are used for source and
   destination, respectively.

   Copyright (c) 2025 IETF Trust and the persons identified as
   authors of the code.  All rights reserved.

   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject to
```

```
      the license terms contained in, the Revised BSD License set
      forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (https://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC 9731; see the
      RFC itself for full legal notices.";

  revision 2025-01-27 {
    description
      "The initial version.";
    reference
      "RFC 9731: A YANG Data Model for Virtual Network (VN)
       Operations";
  }

  /* Features */

  feature multi-src-dest {
    description
      "Support for selection of one src or dest
       among multiple.";
    reference
      "RFC 8453: Framework for Abstraction and Control of TE
       Networks (ACTN)";
  }

  /* Typedef */

  typedef vn-id {
    type string {
      length "1..max";
    }
    description
      "A type definition for a Virtual Network (VN)
       identifier.";
  }

  typedef ap-id {
    type string {
      length "1..max";
    }
    description
      "A type definition for an Access Point (AP) identifier.";
  }

  typedef vnm-id {
    type string {
      length "1..max";
    }
    description
      "A type definition for a VN member identifier.";
  }

  typedef vn-compute-status {
    type te-types:te-common-status;
    description
      "A type definition for representing the VN compute status.
```

```
        Note that all statuses apart from up and down are considered
        to be unknown.";
    }

    /* identities */

    identity vn-computation-error-reason {
      description
        "Base identity for VN computation error reasons.";
    }

    identity vn-computation-error-not-ready {
      base vn-computation-error-reason;
      description
        "VN computation has failed because the MDSC is not
         ready.";
    }

    identity vn-computation-error-no-cnc {
      base vn-computation-error-reason;
      description
        "VN computation has failed because one or more dependent
         CNCs are unavailable.";
    }

    identity vn-computation-error-no-resource {
      base vn-computation-error-reason;
      description
        "VN computation has failed because there is no
         available resource in one or more domains.";
    }

    identity vn-computation-error-path-not-found {
      base vn-computation-error-reason;
      description
        "VN computation failed as no path found.";
    }

    identity vn-computation-ap-unknown {
      base vn-computation-error-reason;
      description
        "VN computation failed as the source or destination Access
         Point (AP) not known.";
    }

    /* Groupings */

    grouping vn-member {
      description
        "The vn-member is described by this grouping.";
      leaf id {
        type vnm-id;
        description
          "A vn-member identifier.";
      }
      container src {
        description
          "The source of VN Member.";
```

```
      leaf ap {
        type leafref {
          path "/access-point/ap/id";
        }
        description
          "A reference to the source AP.";
      }
      leaf vn-ap-id {
        type leafref {
          path "/access-point/ap[id=current()/../ap]/vn-ap"
             + "/id";
        }
        description
          "A reference to the source VNAP.";
      }
      leaf multi-src {
        if-feature "multi-src-dest";
        type boolean;
        default "false";
        description
          "Is the source part of a multi-source, where
           only one of the sources is enabled?";
      }
    }
    container dest {
      description
        "The destination of the VN Member.";
      leaf ap {
        type leafref {
          path "/access-point/ap/id";
        }
        description
          "A reference to the destination AP.";
      }
      leaf vn-ap-id {
        type leafref {
          path "/access-point/ap[id=current()/../ap]/"
             + "vn-ap/id";
        }
        description
          "A reference to the dest VNAP.";
      }
      leaf multi-dest {
        if-feature "multi-src-dest";
        type boolean;
        default "false";
        description
          "Is the destination part of a multi-destination,
           where only one of the destinations is enabled.";
      }
    }
    leaf connectivity-matrix-id {
      type leafref {
        path "/nw:networks/nw:network/nw:node/tet:te/"
           + "tet:te-node-attributes/"
           + "tet:connectivity-matrices/"
           + "tet:connectivity-matrix/tet:id";
      }
```

```
      description
        "A reference to the connectivity-matrix.";
      reference
        "RFC 8795: YANG Data Model for Traffic Engineering (TE)
         Topologies";
    }
    container underlay {
      description
        "An empty container that can be augmented with underlay
         technology information not supported by RFC 8795 (for
         example - Segment Routing (SR).";
    }
    reference
      "RFC 8454: Information Model for Abstraction and Control of TE
       Networks (ACTN)";
  }

  grouping vn-policy {
    description
      "policy for VN-level diversity";
    leaf vn-level-diversity {
      type te-types:te-path-disjointness;
      description
        "The type of disjointness on the VN level (i.e., across all
         VN members).";
    }
  }

  /* Configuration data nodes */

  container access-point {
    description
      "AP configurations.";
    list ap {
      key "id";
      description
        "The access-point identifier.";
      leaf id {
        type ap-id;
        description
          "An AP identifier unique within the scope of the entity
           that controls the VN.";
      }
      leaf pe {
        type leafref {
          path "/nw:networks/nw:network/nw:node/tet:te-node-id";
        }
        description
          "A reference to the PE node in the native TE Topology.";
      }
      leaf max-bandwidth {
        type te-types:te-bandwidth;
        description
          "The max bandwidth of the AP.";
      }
      leaf avl-bandwidth {
        type te-types:te-bandwidth;
        description
```

```
          "The available bandwidth of the AP.";
      }
      list vn-ap {
        key "id";
        leaf id {
          type ap-id;
          description
            "A unique identifier for the VNAP.";
        }
        leaf vn {
          type leafref {
            path "/virtual-network/vn/id";
          }
          description
            "A reference to the VN.";
        }
        leaf abstract-node {
          type leafref {
            path "/nw:networks/nw:network/nw:node/nw:node-id";
          }
          must '/nw:networks/nw:network/nw:node[nw:node-id='
             + 'current()/../abstract-node]/tet:te-node-id' {
            description
              "The associated network for the abstract-node
               must be TE enabled.";
          }
          description
            "A reference to the abstract node that represents
             the VN.";
        }
        leaf ltp {
          type leafref {
            path "/nw:networks/nw:network/nw:node[nw:node-id="
               + "current()/../abstract-node]/nt:termination-point/"
               + "tet:te-tp-id";
          }
          description
            "A reference to the Link Termination Point (LTP)
             in the abstract-node, i.e., the LTP should be in
             the abstract layer and not the underlying layer.";
          reference
            "RFC 8795: YANG Data Model for Traffic Engineering (TE)
             Topologies";
        }
        leaf max-bandwidth {
          type te-types:te-bandwidth;
          config false;
          description
            "The max bandwidth of the VNAP.";
        }
        description
          "List of VNAPs in this AP.";
      }
    }
    reference
      "RFC 8453: Framework for Abstraction and Control of TE
       Networks (ACTN), Section 6";
  }
```

```
  container virtual-network {
    description
      "VN configurations.";
    list vn {
      key "id";
      description
        "A virtual network is identified by a vn-id.";
      leaf id {
        type vn-id;
        description
          "An identifier unique within the scope of the entity
           that controls the VN.";
      }
      uses te-types:te-topology-identifier;
      leaf abstract-node {
        type leafref {
          path "/nw:networks/nw:network/nw:node/tet:te-node-id";
        }
        description
          "A reference to the abstract node in TE Topology.";
      }
      list vn-member {
        key "id";
        description
          "List of vn-members in a VN.";
        uses vn-member;
        leaf oper-status {
          type te-types:te-oper-status;
          config false;
          description
            "The vn-member operational state.";
        }
        leaf if-selected {
          if-feature "multi-src-dest";
          type boolean;
          default "false";
          config false;
          description
            "Is the vn-member selected among the multi-src
             or multi-dest options?";
        }
      }
      leaf admin-status {
        type te-types:te-admin-status;
        default "up";
        description
          "VN administrative state.";
      }
      leaf oper-status {
        type te-types:te-oper-status;
        config false;
        description
          "VN operational state.";
      }
      uses vn-policy;
    }
    reference
      "RFC 8453: Framework for Abstraction and Control of TE
```

```
      Networks (ACTN)";
  }

  /* RPC */

  rpc vn-compute {
    description
      "The VN computation without actual instantiation.  This is
       used by the CNC to get the VN results without actually
       creating it in the network.

       The input could include a reference to the single-node
       -abstract topology.  It could optionally also include
       constraints and optimization criteria.  The computation
       is done based on the list of VN-members.

       The output includes a reference to the single-node
       -abstract topology with each VN-member including a
       reference to the connectivity-matrix-id where the
       path properties could be found.  Error information is
       also included.";
    input {
      uses te-types:te-topology-identifier;
      leaf abstract-node {
        type leafref {
          path "/nw:networks/nw:network/nw:node/tet:te-node-id";
        }
        description
          "A reference to the abstract node in TE Topology.";
      }
      uses te-types:generic-path-constraints;
      leaf cos {
        type te-types:te-ds-class;
        description
          "The class of service (COS).";
      }
      uses te-types:generic-path-optimization;
      list vn-member-list {
        key "id";
        description
          "List of VN-members in a VN.";
        uses vn-member;
        uses te-types:generic-path-constraints;
        leaf cos {
          type te-types:te-ds-class;
          description
            "The class of service.";
          reference
            "RFC 4124: Protocol Extensions for Support of
             Diffserv-aware MPLS Traffic Engineering,
             Section 4.3.1";
        }
        uses te-types:generic-path-optimization;
      }
      uses vn-policy;
    }
    output {
      uses te-types:te-topology-identifier;
```

```
      leaf abstract-node {
        type leafref {
          path "/nw:networks/nw:network/nw:node/tet:te-node-id";
        }
        description
          "A reference to the abstract node in TE Topology.";
      }
      list vn-member-list {
        key "id";
        description
          "List of VN-members in a VN.";
        uses vn-member;
        leaf if-selected {
          if-feature "multi-src-dest";
          type boolean;
          default "false";
          description
            "Is the vn-member selected among the multi-src or
             multi-dest options?";
          reference
            "RFC 8453: Framework for Abstraction and Control of TE
             Networks (ACTN), Section 7";
        }
        leaf compute-status {
          type vn-compute-status;
          description
            "The VN-member compute state.";
        }
        container error-info {
          description
            "Error information related to the VN member.";
          leaf error-description {
            type string {
              length "1..max";
            }
            description
              "Textual representation of the error that occurred
               during VN compute.";
          }
          leaf error-timestamp {
            type yang:date-and-time;
            description
              "Timestamp of the attempt.";
          }
          leaf error-reason {
            type identityref {
              base vn-computation-error-reason;
            }
            description
              "Reason for the VN computation error.";
          }
        }
      }
    }
  }
}
```

```
<CODE ENDS>
```

7. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

The model presented in this document is used in the interface between the CNC and MDSC, which is referred to as CNC-MDSC Interface (CMI). Security risks, such as malicious attack and rogue elements attempting to connect to the various ACTN components, are possible. Furthermore, some ACTN components (e.g., MDSC) represent a single point of failure and threat vector. Also, there is a need to manage policy conflicts and eavesdropping on communication between different ACTN components.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., "config true", which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- ap: This list includes a set of sensitive data that influences how the APs in the VN service are attached. By accessing the following data nodes, an attacker may be able to manipulate the VN.
  - id
  - pe
  - max-bandwidth
  - avl-bandwidth
- vn-ap: This list includes a set of sensitive data that influences how the VN service is delivered. By accessing the following data nodes, an attacker may be able to manipulate the VN.
  - id
  - vn
  - abstract-node
  - ltp
  - max-bandwidth

- vn: This list includes a set of sensitive data that influences how the VN service is delivered. By accessing the following data nodes, an attacker may be able to manipulate the VN.
  - id
  - te-topology-identifier
  - abstract-node
- vn-member: This list includes a set of sensitive data that influences how the VN member in the VN service is delivered. By accessing the following data nodes, an attacker may be able to manipulate the VN member.
  - id
  - src/ap
  - src/vn-ap-id
  - dest/ap
  - dest/vn-ap-id
  - connectivity-matrix-id

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- oper-status: This leaf can reveal the current operational state of the VN.
- if-selected: This leaf can reveal which vn-member is selected among the various multi-src/dest options.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

- vn-compute: This RPC triggers the VN computation at the MDSC, which can reveal the VN information.

8. IANA Considerations

IANA has made the following allocation for a URI in the "ns" registry within the "IETF XML Registry" registry group [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-vn
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

IANA has made the following allocation for the VN YANG module (see Section 5in the "YANG Module Names" registry [RFC6020]:

name: ietf-vn
namespace: urn:ietf:params:xml:ns:yang:ietf-vn
prefix: vn
reference: RFC 9731

9. References

9.1. Normative References

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

[RFC4124] Le Faucheur, F., Ed., "Protocol Extensions for Support of Diffserv-aware MPLS Traffic Engineering", RFC 4124, DOI 10.17487/RFC4124, June 2005, <https://www.rfc-editor.org/info/rfc4124>.

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

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

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

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

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

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/info/rfc8345>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8776] Saad, T., Gandhi, R., Liu, X., Beeram, V., and I. Bryskin, "Common YANG Data Types for Traffic Engineering", RFC 8776, DOI 10.17487/RFC8776, June 2020, <https://www.rfc-editor.org/info/rfc8776>.

[RFC8795] Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Gonzalez de Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", RFC 8795, DOI 10.17487/RFC8795, August 2020, <https://www.rfc-editor.org/info/rfc8795>.

9.2. Informative References

[L1CSM-YANG] Lee, Y., Lee, K., Zheng, H., Gonzalez de Dios, O., and D. Ceccarelli, "A YANG Data Model for L1 Connectivity Service Model (L1CSM)", Work in Progress, Internet-Draft, draft-ietf-ccamp-l1csm-yang-26, 11 April 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-ccamp-l1csm-yang-26>.

[RFC7926] Farrel, A., Ed., Drake, J., Bitar, N., Swallow, G., Ceccarelli, D., and X. Zhang, "Problem Statement and Architecture for Information Exchange between Interconnected Traffic-Engineered Networks", BCP 206, RFC 7926, DOI 10.17487/RFC7926, July 2016, <https://www.rfc-editor.org/info/rfc7926>.

[RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <https://www.rfc-editor.org/info/rfc8299>.

[RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <https://www.rfc-editor.org/info/rfc8309>.

[RFC8453] Ceccarelli, D., Ed. and Y. Lee, Ed., "Framework for Abstraction and Control of TE Networks (ACTN)", RFC 8453, DOI 10.17487/RFC8453, August 2018, <https://www.rfc-editor.org/info/rfc8453>.

[RFC8454] Lee, Y., Belotti, S., Dhody, D., Ceccarelli, D., and B. Yoon, "Information Model for Abstraction and Control of TE Networks (ACTN)", RFC 8454, DOI 10.17487/RFC8454, September 2018, <https://www.rfc-editor.org/info/rfc8454>.

[RFC8466] Wen, B., Fioccola, G., Ed., Xie, C., and L. Jalil, "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", RFC 8466, DOI 10.17487/RFC8466, October 2018, <https://www.rfc-editor.org/info/rfc8466>.

[RFC9256] Filsfils, C., Talaulikar, K., Ed., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", RFC 9256, DOI 10.17487/RFC9256, July 2022, <https://www.rfc-editor.org/info/rfc9256>.

[TE-SERVICE-MAPPING] Lee, Y., Dhody, D., Fioccola, G., Wu, Q., Ceccarelli, D., and J. Tantsura, "Traffic Engineering (TE) and Service Mapping YANG Data Model", Work in Progress, Internet-Draft, draft-ietf-teas-te-service-mapping-yang-16, 20 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-te-service-mapping-yang-16>.

[TEAS-ACTN-PM] Lee, Y., Dhody, D., Vilalta, R., King, D., and D. Ceccarelli, "YANG models for Virtual Network (VN)/TE Performance Monitoring Telemetry and Scaling Intent Autonomics", Work in Progress, Internet-Draft, draft-ietf-teas-actn-pm-telemetry-autonomics-14, 19 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-actn-pm-telemetry-autonomics-14>.

[YANG-TE] Saad, T., Gandhi, R., Liu, X., Beeram, V. P., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels, Label Switched Paths and Interfaces", Work in Progress, Internet-Draft, draft-ietf-teas-yang-te-37, 9 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-yang-te-37>.

Appendix A. Performance Constraints

At the creation of a VN, it is natural to provide VN-level constraints and optimization criteria. It should be noted that the VN YANG module described in this document relies on the TE-Topology Model in [RFC8795] by using a reference to an abstract node to achieve this. Further, the connectivity-matrix structure is used to assign the constraints and optimization criteria including delay, jitter, etc. [RFC8776] defines some of the metric-types; future documents are meant to augment it.

Note that the VN compute allows the inclusion of the constraints and the optimization criteria directly in the RPC to allow it to be used independently.

Appendix B. JSON Example

B.1. VN JSON

This section provides JSON examples of how the VN YANG model and TE topology model are used together to instantiate a VN.

The example in this section includes the following VNs:

- VN1 (Type 1): This VN maps to the single node topology abstract1 and consists of VN Members 104 (L1 to L4), 107 (L1 to L7), 204 (L2 to L4), 308 (L3 to L8), and 108 (L1 to L8).
- VN2 (Type 2): This VN maps to the single node topology abstract2; this topology has an underlay topology (called underlay). This VN has a single VN member 105 (L1 to L5) and an underlay path (S4 and S7) has been set in the connectivity matrix of the abstract2 topology;
- VN3 (Type 1): This VN has a multi-source and multi-destination feature enabled. VN Member 106 (L1 to L6) and 107 (L1 to L7) showcase multi-dest and VN Member 108 (L1 to L8) and 308

(L3 to L8) showcase the multi-src feature. The selected VN-member is known via the field "if-selected" and the corresponding connectivity-matrix-id.

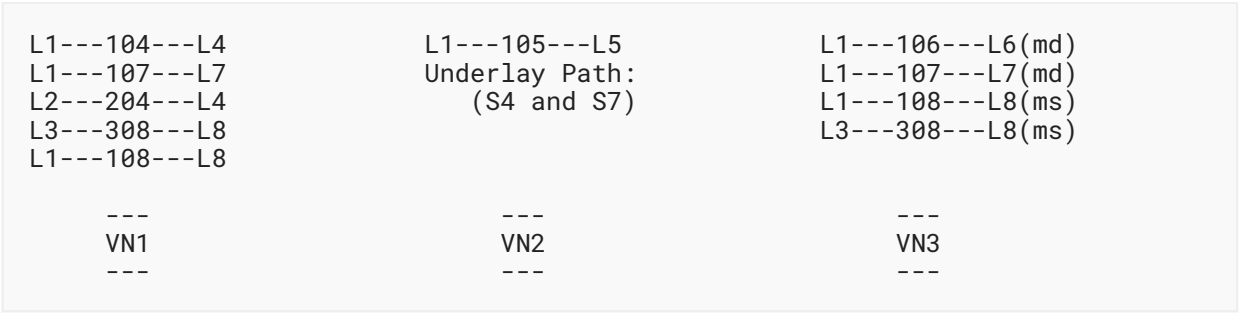

Figure 11

Note that the VN YANG model also includes the AP and VNAP, which shows various VNs using the same AP.

```
{
  "ietf-vn:access-point": {
    "ap": [
      {
        "id": "101",
        "vn-ap": [
          {
            "id": "10101",
            "vn": "1",
            "abstract-node": "192.0.2.1",
            "ltp": "203.0.113.11"
          },
          {
            "id": "10102",
            "vn": "2",
            "abstract-node": "192.0.2.2",
            "ltp": "203.0.113.12"
          },
          {
            "id": "10103",
            "vn": "3",
            "abstract-node": "192.0.2.3",
            "ltp": "203.0.113.13"
          }
        ]
      },
      {
        "id": "202",
        "vn-ap": [
          {
            "id": "20201",
            "vn": "1",
            "abstract-node": "192.0.2.1",
            "ltp": "203.0.113.21"
          }
```

```
        ]
      },
      {
        "id": "303",
        "vn-ap": [
          {
            "id": "30301",
            "vn": "1",
            "abstract-node": "192.0.2.1",
            "ltp": "203.0.113.31"
          },
          {
            "id": "30303",
            "vn": "3",
            "abstract-node": "192.0.2.3",
            "ltp": "203.0.113.33"
          }
        ]
      },
      {
        "id": "404",
        "vn-ap": [
          {
            "id": "40401",
            "vn": "1",
            "abstract-node": "192.0.2.1",
            "ltp": "203.0.113.41"
          }
        ]
      },
      {
        "id": "505",
        "vn-ap": [
          {
            "id": "50502",
            "vn": "2",
            "abstract-node": "192.0.2.2",
            "ltp": "203.0.113.52"
          }
        ]
      },
      {
        "id": "606",
        "vn-ap": [
          {
            "id": "60603",
            "vn": "3",
            "abstract-node": "192.0.2.3",
            "ltp": "203.0.113.63"
          }
        ]
      },
      {
        "id": "707",
        "vn-ap": [
          {
            "id": "70701",
            "vn": "1",
```

```
            "abstract-node": "192.0.2.1",
            "ltp": "203.0.113.71"
          },
          {
            "id": "70703",
            "vn": "3",
            "abstract-node": "192.0.2.3",
            "ltp": "203.0.113.73"
          }
        ]
      },
      {
        "id": "808",
        "vn-ap": [
          {
            "id": "80801",
            "vn": "1",
            "abstract-node": "192.0.2.1",
            "ltp": "203.0.113.81"
          },
          {
            "id": "80803",
            "vn": "3",
            "abstract-node": "192.0.2.3",
            "ltp": "203.0.113.83"
          }
        ]
      }
    ]
  },
  "ietf-vn:virtual-network": {
    "vn": [
      {
        "id": "1",
        "te-topology-identifier": {
          "topology-id": "abstract1"
        },
        "abstract-node": "192.0.2.1",
        "vn-member": [
          {
            "id": "104",
            "src": {
              "ap": "101",
              "vn-ap-id": "10101"
            },
            "dest": {
              "ap": "404",
              "vn-ap-id": "40401"
            },
            "connectivity-matrix-id": 10104
          },
          {
            "id": "107",
            "src": {
              "ap": "101",
              "vn-ap-id": "10101"
            },
            "dest": {
```

```
              "ap": "707",
              "vn-ap-id": "70701"
            },
            "connectivity-matrix-id": 10107
          },
          {
            "id": "204",
            "src": {
              "ap": "202",
              "vn-ap-id": "20201"
            },
            "dest": {
              "ap": "404",
              "vn-ap-id": "40401"
            },
            "connectivity-matrix-id": 10204
          },
          {
            "id": "308",
            "src": {
              "ap": "303",
              "vn-ap-id": "30301"
            },
            "dest": {
              "ap": "808",
              "vn-ap-id": "80801"
            },
            "connectivity-matrix-id": 10308
          },
          {
            "id": "108",
            "src": {
              "ap": "101",
              "vn-ap-id": "10101"
            },
            "dest": {
              "ap": "808",
              "vn-ap-id": "80801"
            },
            "connectivity-matrix-id": 10108
          }
        ]
      },
      {
        "id": "2",
        "te-topology-identifier": {
          "topology-id": "abstract2"
        },
        "abstract-node": "192.0.2.2",
        "vn-member": [
          {
            "id": "105",
            "src": {
              "ap": "101",
              "vn-ap-id": "10102"
            },
            "dest": {
              "ap": "505",
```

```
              "vn-ap-id": "50502"
            },
            "connectivity-matrix-id": 20105
          }
        ]
      },
      {
        "id": "3",
        "te-topology-identifier": {
          "topology-id": "abstract3"
        },
        "abstract-node": "192.0.2.3",
        "vn-member": [
          {
            "id": "106",
            "src": {
              "ap": "101",
              "vn-ap-id": "10103"
            },
            "dest": {
              "ap": "606",
              "vn-ap-id": "60603",
              "multi-dest": true
            },
            "connectivity-matrix-id": 30106,
            "if-selected": false
          },
          {
            "id": "107",
            "src": {
              "ap": "101",
              "vn-ap-id": "10103"
            },
            "dest": {
              "ap": "707",
              "vn-ap-id": "70703",
              "multi-dest": true
            },
            "connectivity-matrix-id": 30107,
            "if-selected": true
          },
          {
            "id": "108",
            "src": {
              "ap": "101",
              "vn-ap-id": "10103",
              "multi-src": true
            },
            "dest": {
              "ap": "808",
              "vn-ap-id": "80803",
            },
            "connectivity-matrix-id": 30108,
            "if-selected": false
          },
          {
            "id": "308",
            "src": {
```

```
              "ap": "303",
              "vn-ap-id": "30303",
              "multi-src": true
            },
            "dest": {
              "ap": "808",
              "vn-ap-id": "80803"
            },
            "connectivity-matrix-id": 30308,
            "if-selected": true
          }
        ]
      }
    ]
  }
}
```

B.2. TE-Topology JSON

This section provides JSON examples of the various TE topology instances.

The example in this section includes the following TE Topologies:

- abstract1: a single node TE topology referenced by VN1. We also show how disjointness (node, link, Shared Risk Link Group (SRLG)) is supported in the example on the connectivity matrices.
- abstract2: a single node TE topology referenced by VN2 with an underlay path.
- underlay: the topology with multiple nodes (in the underlay path of abstract2). For brevity, the example includes only the node: other parameters are not included.
- abstract3: a single node TE topology referenced by VN3.

```
{
  "ietf-network:networks": {
    "network": [
      {
        "network-types": {
          "ietf-te-topology:te-topology": {}
        },
        "network-id": "example:abstract1",
        "ietf-te-topology:te-topology-identifier": {
          "provider-id": 0,
          "client-id": 0,
          "topology-id": "example:abstract1"
        },
        "node": [
          {
            "node-id": "example:192.0.2.1",
            "ietf-network-topology:termination-point": [
              {
                "tp-id": "example:1-0-1",
                "ietf-te-topology:te-tp-id": "203.0.113.11"
              },
              {
```

```
                "tp-id": "example:1-0-2",
                "ietf-te-topology:te-tp-id": "203.0.113.21"
              },
              {
                "tp-id": "example:1-0-3",
                "ietf-te-topology:te-tp-id": "203.0.113.31"
              },
              {
                "tp-id": "example:1-0-4",
                "ietf-te-topology:te-tp-id": "203.0.113.41"
              },
              {
                "tp-id": "example:1-0-7",
                "ietf-te-topology:te-tp-id": "203.0.113.71"
              },
              {
                "tp-id": "example:1-0-8",
                "ietf-te-topology:te-tp-id": "203.0.113.81"
              }
            ],
            "ietf-te-topology:te-node-id": "192.0.2.1",
            "ietf-te-topology:te": {
              "te-node-attributes": {
                "domain-id": 1,
                "is-abstract": [
                  null
                ],
                "connectivity-matrices": {
                  "is-allowed": true,
                  "path-constraints": {
                    "te-bandwidth": {
                      "generic": "0x1p10"
                    },
                    "disjointness": "node link srlg"
                  },
                  "connectivity-matrix": [
                    {
                      "id": 10104,
                      "from": {
                        "tp-ref": "example:1-0-1"
                      },
                      "to": {
                        "tp-ref": "example:1-0-4"
                      }
                    },
                    {
                      "id": 10107,
                      "from": {
                        "tp-ref": "example:1-0-1"
                      },
                      "to": {
                        "tp-ref": "example:1-0-7"
                      }
                    },
                    {
                      "id": 10204,
                      "from": {
                        "tp-ref": "example:1-0-2"
```

```
                    },
                    "to": {
                      "tp-ref": "example:1-0-4"
                    }
                  },
                  {
                    "id": 10308,
                    "from": {
                      "tp-ref": "example:1-0-3"
                    },
                    "to": {
                      "tp-ref": "example:1-0-8"
                    }
                  },
                  {
                    "id": 10108,
                    "from": {
                      "tp-ref": "example:1-0-1"
                    },
                    "to": {
                      "tp-ref": "example:1-0-8"
                    }
                  }
                ]
              }
            }
          }
        }
      ]
    },
    {
      "network-types": {
        "ietf-te-topology:te-topology": {}
      },
      "network-id": "example:abstract2",
      "ietf-te-topology:te-topology-identifier": {
        "provider-id": 0,
        "client-id": 0,
        "topology-id": "example:abstract2"
      },
      "node": [
        {
          "node-id": "example:192.0.2.2",
          "ietf-network-topology:termination-point": [
            {
              "tp-id": "example:2-0-1",
              "ietf-te-topology:te-tp-id": "203.0.113.12"
            },
            {
              "tp-id": "example:2-0-5",
              "ietf-te-topology:te-tp-id": "203.0.113.52"
            }
          ],
          "ietf-te-topology:te-node-id": "192.0.2.2",
          "ietf-te-topology:te": {
            "te-node-attributes": {
              "domain-id": 1,
              "is-abstract": [
```

```
                  null
                ],
                "connectivity-matrices": {
                  "is-allowed": true,
                  "underlay": {
                    "enabled": true
                  },
                  "path-constraints": {
                    "te-bandwidth": {
                      "generic": "0x1p10"
                    }
                  },
                  "optimizations": {
                    "objective-function": {
                      "objective-function-type":
                       "ietf-te-types:of-maximize-residual-bandwidth"
                    }
                  },
                  "ietf-te-topology:connectivity-matrix": [
                    {
                      "id": 20105,
                      "from": {
                        "tp-ref": "example:2-0-1"
                      },
                      "to": {
                        "tp-ref": "example:2-0-5"
                      },
                      "underlay": {
                        "enabled": true,
                        "primary-path": {
                          "network-ref": "example:underlay",
                          "path-element": [
                            {
                              "path-element-id": 1,
                              "numbered-node-hop": {
                                "node-id": "198.51.100.44",
                                "hop-type": "strict"
                              }
                            },
                            {
                              "path-element-id": 2,
                              "numbered-node-hop": {
                                "node-id": "198.51.100.77",
                                "hop-type": "strict"
                              }
                            }
                          ]
                        }
                      }
                    }
                  ]
                }
              }
            }
          }
        ]
      },
      {
```

```
        "network-types": {
          "ietf-te-topology:te-topology": {}
        },
        "network-id": "example:underlay",
        "ietf-te-topology:te-topology-identifier": {
          "provider-id": 0,
          "client-id": 0,
          "topology-id": "example:underlay"
        },
        "node": [
          {
            "node-id": "example:198.51.100.11",
            "ietf-te-topology:te-node-id": "198.51.100.11"
          },
          {
            "node-id": "example:198.51.100.22",
            "ietf-te-topology:te-node-id": "198.51.100.22"
          },
          {
            "node-id": "example:198.51.100.33",
            "ietf-te-topology:te-node-id": "198.51.100.33"
          },
          {
            "node-id": "example:198.51.100.44",
            "ietf-te-topology:te-node-id": "198.51.100.44"
          },
          {
            "node-id": "example:198.51.100.55",
            "ietf-te-topology:te-node-id": "198.51.100.55"
          },
          {
            "node-id": "example:198.51.100.66",
            "ietf-te-topology:te-node-id": "198.51.100.66"
          },
          {
            "node-id": "example:198.51.100.77",
            "ietf-te-topology:te-node-id": "198.51.100.77"
          },
          {
            "node-id": "example:198.51.100.88",
            "ietf-te-topology:te-node-id": "198.51.100.88"
          },
          {
            "node-id": "example:198.51.100.99",
            "ietf-te-topology:te-node-id": "198.51.100.99"
          }
        ]
      },
      {
        "network-types": {
          "ietf-te-topology:te-topology": {}
        },
        "network-id": "example:abstract3",
        "ietf-te-topology:te-topology-identifier": {
          "provider-id": 0,
          "client-id": 0,
          "topology-id": "example:abstract3"
        },
```

```
        "node": [
          {
            "node-id": "example:192.0.2.3",
            "ietf-network-topology:termination-point": [
              {
                "tp-id": "example:3-0-1",
                "ietf-te-topology:te-tp-id": "203.0.113.13"
              },
              {
                "tp-id": "example:3-0-3",
                "ietf-te-topology:te-tp-id": "203.0.113.33"
              },
              {
                "tp-id": "example:3-0-6",
                "ietf-te-topology:te-tp-id": "203.0.113.63"
              },
              {
                "tp-id": "example:3-0-7",
                "ietf-te-topology:te-tp-id": "203.0.113.73"
              },
              {
                "tp-id": "example:3-0-8",
                "ietf-te-topology:te-tp-id": "203.0.113.83"
              }
            ],
            "ietf-te-topology:te-node-id": "192.0.2.3",
            "ietf-te-topology:te": {
              "te-node-attributes": {
                "domain-id": 3,
                "is-abstract": [
                  null
                ],
                "connectivity-matrices": {
                  "is-allowed": true,
                  "path-constraints": {
                    "te-bandwidth": {
                      "generic": "0x1p10"
                    }
                  },
                  "connectivity-matrix": [
                    {
                      "id": 30107,
                      "from": {
                        "tp-ref": "example:3-0-1"
                      },
                      "to": {
                        "tp-ref": "example:3-0-7"
                      }
                    },
                    {
                      "id": 30106,
                      "from": {
                        "tp-ref": "example:3-0-1"
                      },
                      "to": {
                        "tp-ref": "example:3-0-6"
                      }
                    },
```

```
                    {
                      "id": 30108,
                      "from": {
                        "tp-ref": "example:3-0-1"
                      },
                      "to": {
                        "tp-ref": "example:3-0-8"
                      }
                    },
                    {
                      "id": 30308,
                      "from": {
                        "tp-ref": "example:3-0-3"
                      },
                      "to": {
                        "tp-ref": "example:3-0-8"
                      }
                    }
                  ]
                }
              }
            }
          }
        ]
      }
    ]
  }
}
```

Acknowledgments

The authors would like to thank Xufeng Liu, Adrian Farrel, Tom Petch, Mohamed Boucadair, Italo Busi, Bo Wu, and Daniel King for their helpful comments and valuable suggestions.

Thanks to:

- Andy Bierman for the YANGDIR review.
- Darren Dukes and Susan Hares for the RTGDIR review.
- Behcet Sarikaya for the GENART review.
- Bo Wu for the OPSDIR review.
- Shivan Sahib for the SECDIR review.
- Deb Cooley, Francesca Palombini, Gunter Van de Velde, and Mahesh Jethanandani for the IESG review.

Contributors' Addresses

Qin Wu
Huawei Technologies
Email: bill.wu@huawei.com

Peter Park
KT
Email: peter.park@kt.com

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

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

Kenichi Ogaki
KDDI
Email: ke-oogaki@kddi.com

Authors' Addresses

Young Lee (EDITOR)
Samsung Electronics
Email: younglee.tx@gmail.com

Dhruv Dhody (EDITOR)
Huawei
India
Email: dhruv.ietf@gmail.com

Daniele Ceccarelli
Cisco
Email: daniele.ietf@gmail.com

Igor Bryskin
Individual
Email: i_bryskin@yahoo.com

Bin Yeong Yoon
ETRI
Email: byyun@etri.re.kr