Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 12, 2019

I. Bryskin
Huawei Technologies
X. Liu
Volta Networks
Y. Lee
J. Guichard
Huawei Technologies
L. Contreras
Telefonica
D. Ceccarelli
Ericsson
J. Tantsura
Apstra Networks

March 11, 2019

# SF Aware TE Topology YANG Model
**draft-ietf-teas-sf-aware-topo-model-03**

## Abstract

This document describes a YANG data model for TE network topologies that are network service and function aware.

## Status of This Memo

## Copyright Notice

Table of Contents

## 1. Introduction

Normally network connectivity services are discussed as a means to inter-connect various abstract or physical network topological elements, such as ports, link termination points and nodes [I-D.ietf-teas-yang-te-topo] [I-D.ietf-teas-yang-te]. However, the connectivity services, strictly speaking, interconnect not the network topology elements per-se, rather, located on/associated with the various network and service functions [RFC7498] [RFC7665]. In many scenarios it is beneficial to decouple the service/network functions from the network topology elements hosting them, describe them in some unambiguous and identifiable way (so that it would be possible, for example, to auto-discover on the network topology service/network functions with identical or similar functionality and characteristics) and engineer the connectivity between the service/ network functions, rather than between their current topological locations.

Today a network offers to its clients far more services than just connectivity across the network. Large variety of physical, logical and/or virtual service functions, network functions and transport functions (collectively named in this document as SFs) could be allocated for and assigned to a client. As described in the appendix of this document, there are some important use cases, in which the network needs to represent to the client SFs at the client's disposal as topological elements in relation to other elements of a topology (i.e. nodes, links, link and tunnel termination points) used by the network to describe itself to the client. Not only would such information allow for the client to auto-discover the network's SFs available for the services provisioned for the client, it would also allow for the client selecting the SFs, duel-optimizing the selection on the SF location on the network and connectivity means (e.g. TE tunnels) to inter-connect the SFs. Consequently thus would give to both the network and the client powerful means for the service function chain (SFC [RFC7498] [RFC7665]) negotiation to achieve most efficient and cost effective (from the network point of view) and most optimal yet satisfying all necessary constraints of SFCs (from the client's point of view).

This document defines a YANG data model that allows service functions to be represented along with TE topology elements.

### 1.1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].

o  Network Function (NF): A functional block within a network infrastructure that has well-defined external interfaces and well-defined functional behaviour [ETSI-NFV-TERM].  Such functions include message router, CDN, session border controller, WAN cceleration, DPI, firewall, NAT, QoE monitor, PE router, BRAS, and radio/fixed access network nodes.

o  Network Service: Composition of Network Functions and defined by its functional and behavioural specification.  The Network Service contributes to the behaviour of the higher layer service, which is characterized by at least performance, dependability, and security specifications.  The end-to-end network service behaviour is the result of the combination of the individual network function behaviours as well as the behaviours of the network infrastructure composition mechanism [ETSI-NFV-TERM].

o  Service Function (SF): A function that is responsible for specific treatment of received packets.  A service function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers).  As a logical component, a service function can be realized as a virtual element or be embedded in a physical network element.  One or more service functions can be embedded in the same network element.  Multiple occurrences of the service function can exist in the same administrative domain.  A non-exhaustive list of service functions includes: firewalls, WAN and application acceleration, Deep Packet Inspection (DPI), server load balancers, NAT44 [RFC3022], NAT64 [RFC6146], HTTP header enrichment functions, and TCP optimizers.  The generic term "L4-L7 services" is often used to describe many service functions [RFC7498].

o  Service Function Chain (SFC): A service function chain defines an ordered or partially ordered set of abstract service functions and ordering constraints that must be applied to packets, frames, and/or flows selected as a result of classification.  An example of an abstract service function is a firewall.  The implied order may not be a linear progression as the architecture allows for SFCs that copy to more than one branch, and also allows for cases where there is flexibility in the order in which service functions need to be applied.  The term "service chain" is often used as shorthand for "service function chain" [RFC7498].

o  Connectivity Service: Any service between layer 0 and layer 3 aiming at delivering traffic among two or more end customer edge nodes connected to provider edge nodes.  Examples include L3VPN, L2VPN etc.

- o Link Termination Point (LTP): 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 [I-D.ietf-teas-yang-te-topo].

- o Tunnel Termination Point (TTP): 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 [I-D.ietf-teas-yang-te-topo].

The following terms are defined in [RFC7950] and are not redefined here:

- o augment

- o data model

- o data node

## 1.2. Tree Diagrams

A simplified graphical representation of the data model is presented in this document, by using the tree format defined in [I-D.ietf-netmod-yang-tree-diagrams].

## 1.3. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

| Prefix | YANG module | Reference |
|---|---|---|
| inet | ietf-inet-types | [RFC6991] |
| nw | ietf-network | [I-D.ietf-i2rs-yang-network-topo] |
| nt | ietf-network-topology | [I-D.ietf-i2rs-yang-network-topo] |
| tet | ietf-te-topology | [I-D.ietf-teas-yang-te-topo] |

Table 1: Prefixes and Corresponding YANG Modules

## 2. Modeling Considerations

The model introduced in this document is an augmentation of the TE Topology model defined in [I-D.ietf-teas-yang-te-topo]. SFs are modeled as child elements of a TE node similarly to how Link Termination Points (LTPs) and Tunnel Termination Points (TTPs) are modeled in the TE Topology model. The SFs are defined as opaque objects identified via topology unique service-function-id's. Each SF has one or more Connection Points (CPs) identified via SF-unique sf-connection-point-id's, over which the SF could be connected to other SFs resided on the same TE node, as well as to other elements of the TE node, in particular, to the node's LTPs and/or TTPs. An interested client may use service-function-id's to look up the SFs in TOSCA or YANG data store(s) defined by [ETSI-NFV-MAN] to retrieve the details of the SFs, for example, to understand the SF's mutual substitutability.

The TE Topology model introduces a concept of Connectivity Matrix (CM), and uses the CM to describe which and at what costs a TE node's LTPs could be inter-connected internally across the TE node. The model defined in this document heavily uses the same concept to describe the SF connectivity via introducing 3 additional CMs:

1. SF2SF CM. This CM describes which pairs of SFs could be locally inter-connected, and, if yes, in which direction, via which CPs and at what costs. In other words, the SF2SF CM describes how SFs residing on the same TE node could be inter-connected into local from the TE node's perspective SFCs;

2. SF2LTP CM. This CM describes how, in which direction and at what costs the TE node's SFs could be connected to the TE node's LTPs and hence to SFs residing on neighboring TE nodes that are connected to LTPs at the remote ends of corresponding TE links;

3. SF2TTP CM. This CM describes how, in which direction and at what costs the TE node's SFs could be connected to the TE node's TTPs and hence to SFs residing on other TE nodes on the topology that could be inter-connected with the TE node in question via TE tunnels terminated by the corresponding TTPs.

In addition to SF2SF CM, the local SF chaining could be described with the help of ETSI models Virtual Links (VLs) [ETSI-NFV-MAN]. This option is especially useful when the costs of the local chaining are negligible as compared to ones of the end-to-end SFCs said local SFCs are part of.

Section 3 and 4 provide the YANG model structure and the YANG module for SF-aware Topology. Section 5 and 6 provide the YANG model

structure and the YANG module for Data Center Compute Node resource abstraction.  This provides an example of SF2LTP CM where DC compute nodes are connected to LTPs at the remote ends of the corresponding TE links.  This use-case is described in Section 10 of Appendix C.

## 3.  Model Structure

```
   module: ietf-te-topology-sf
     augment /nw:networks/nw:network/nw:network-types/tet:te-topology:
       +--rw sf!
     augment /nw:networks/nw:network/nw:node/tet:te
   /tet:te-node-attributes:
       +--rw service-function
          +--rw connectivity-matrices
          |  +--rw connectivity-matrix* [id]
          |     +--rw id                  uint32
          |     +--rw from
          |     |  +--rw service-function-id?      string
          |     |  +--rw sf-connection-point-id?   string
          |     +--rw to
          |     |  +--rw service-function-id?      string
          |     |  +--rw sf-connection-point-id?   string
          |     +--rw enabled?            boolean
          |     +--rw direction?          connectivity-direction
          |     +--rw virtual-link-id?    string
          +--rw link-terminations
             +--rw link-termination* [id]
                +--rw id            uint32
                +--rw from
                |  +--rw tp-ref?   -> ../../../../../../..
   /nt:termination-point/tp-id
                +--rw to
                |  +--rw service-function-id?      string
                |  +--rw sf-connection-point-id?   string
                +--rw enabled?     boolean
                +--rw direction?   connectivity-direction
     augment /nw:networks/nw:network/nw:node/tet:te
   /tet:information-source-entry:
       +--ro service-function
          +--ro connectivity-matrices
          |  +--ro connectivity-matrix* [id]
          |     +--ro id                  uint32
          |     +--ro from
          |     |  +--ro service-function-id?      string
          |     |  +--ro sf-connection-point-id?   string
          |     +--ro to
          |     |  +--ro service-function-id?      string
          |     |  +--ro sf-connection-point-id?   string
```

```
          |     +--ro enabled?           boolean
          |     +--ro direction?         connectivity-direction
          |     +--ro virtual-link-id?   string
          +--ro link-terminations
             +--ro link-termination* [id]
                +--ro id           uint32
                +--ro from
                +--ro to
                |  +--ro service-function-id?      string
                |  +--ro sf-connection-point-id?   string
                +--ro enabled?     boolean
                +--ro direction?   connectivity-direction
     augment /nw:networks/nw:network/nw:node/tet:te
   /tet:tunnel-termination-point:
       +--rw service-function
          +--rw tunnel-terminations
             +--rw tunnel-termination* [id]
                +--rw id                       uint32
                +--rw service-function-id?      string
                +--rw sf-connection-point-id?   string
                +--rw enabled?                  boolean
                +--rw direction?                connectivity-direction
```

## 4.  YANG Modules

```
   <CODE BEGINS> file "ietf-te-topology-sf@2018-02-27.yang"
   module ietf-te-topology-sf {
     yang-version 1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-te-topology-sf";

     prefix "tet-sf";

     import ietf-network {
       prefix "nw";
     }

     import ietf-network-topology {
       prefix "nt";
     }

     import ietf-te-topology {
       prefix "tet";
     }

     organization
       "Traffic Engineering Architecture and Signaling (TEAS)
```

```
        Working Group";

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

        Editors:  Igor Bryskin
                  <mailto:Igor.Bryskin@huawei.com>

                  Xufeng Liu
                  <mailto:Xufeng_Liu@jabil.com>";

     description
       "Network service and function aware aware TE topology model.

        Copyright (c) 2018 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

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

     revision 2018-02-27 {
       description "Initial revision";
       reference "TBD";
     }

     /*
      * Typedefs
      */
     typedef connectivity-direction {
       type enumeration {
         enum "to" {
           description
           "The direction is uni-directional, towards the 'to'
            entity direction.";
         }
         enum "from" {
           description
           "The direction is uni-directional, from the 'to'
            entity direction.";
         }
         enum "bidir" {
           description
           "The direction is bi-directional.";
```

```
         }
       }
       description
         "A type used to indicates whether a connectivity is
          uni-directional, or bi-directional. If the relation is
          uni-directional, the value of this type indicates the
          direction.";
     } // connectivity-direction

     /*
      * Groupings
      */
     grouping service-function-node-augmentation {
       description
         "Augmenting a TE node to be network service and function
          aware.";
       container service-function {
         description
           "Containing attributes related to network services and
            network functions";
         container connectivity-matrices {
           description
             "Connectivity relations between network services/functions
              on a TE node, which can be either abstract or physical.";
           reference
             "ETSI GS NFV-MAN 01: Network Functions Virtualisation
              (NFV); Management and Orchestration.
              RFC7665: Service Function Chaining (SFC) Architecture.";
           list connectivity-matrix {
             key "id";
             description
               "Represents the connectivity relations between network
                services/functions on a TE node.";
             leaf id {
               type uint32;
               description "Identifies the connectivity-matrix entry.";
             }

             container from {
               description
                 "Reference to the source network service or
                  network function.";
               leaf service-function-id {
                 type string;
                 description
                   "Reference to a network service or a network
                    function.";
               }
```

```
              leaf sf-connection-point-id {
                type string;
                description
                  "Reference to a connection point on a network
                   service or a network function.";
              }
            } // from
            container to {
              description
                "Reference to the destination network service or
                 network function.";
              leaf service-function-id {
                type string;
                description
                  "Reference to a network service or a network
                   function.";
                }
              leaf sf-connection-point-id {
                type string;
                description
                  "Reference to a connection point on a network
                   service or a network function.";
              }
            } // to
            leaf enabled {
              type boolean;
              description
                "'true' if this connectivity entry is enabled.";
            }
            leaf direction {
              type connectivity-direction;
              description
                "Indicates whether this connectivity is
                 uni-directional, or bi-directional. If the
                 relation is uni-directional, the value of
                 this leaf indicates the direction.";
            }
            leaf virtual-link-id {
              type string;
              description
                "Reference to a virtual link that models this
                 conectivity relation in the network function
                 model.";
            }
          } // connectivity-matrix
        } // connectivity-matrices

        container link-terminations {
```

```
           description
             "Connectivity relations between network services/functions
              and link termination points on a TE node, which can be
              either abstract or physical.";
           reference
             "ETSI GS NFV-MAN 01: Network Functions Virtualisation
              (NFV); Management and Orchestration.
              RFC7665: Service Function Chaining (SFC) Architecture.";
           list link-termination {
             key "id";
             description
               "Each entry of the list represents the connectivity
                relation between a network service/function and
                a link termination point on a TE node.";
             leaf id {
               type uint32;
               description "Identifies the termination entry.";
             }

             container from {
               description
                 "Reference to the link termination point.";
             } // from
             container to {
               description
                 "Reference to the network service or network
                  function.";
               leaf service-function-id {
                 type string;
                 description
                   "Reference to a network service or a network
                    function.";
                 }
               leaf sf-connection-point-id {
                 type string;
                 description
                   "Reference to a connection point on a network
                    service or a network function.";
               }
             } // to
             leaf enabled {
               type boolean;
               description
                 "'true' if this connectivity entry is enabled.";
             }
             leaf direction {
               type connectivity-direction;
               description
```

```
                 "Indicates whether this connectivity is
                  uni-directional, or bi-directional. If the
                  relation is uni-directional, the value of
                  this leaf indicates the direction.";
             }
           } // link-termination
         }
       }
     } // service-function-node-augmentation

     grouping service-function-ttp-augmentation {
       description
         "Augmenting a tunnel termination point to be network service
          aware.";
       container service-function {
         description
           "Containing attributes related to network services and
            network functions";
         container tunnel-terminations {
           description
             "Connectivity relations between network services/functions
              and tunnel termination points on a TE node, which can be
              either abstract or physical.";
           reference
             "ETSI GS NFV-MAN 01: Network Functions Virtualisation
              (NFV); Management and Orchestration.
              RFC7665: Service Function Chaining (SFC) Architecture.";
           list tunnel-termination {
             key "id";
             description
               "Each entry of the list represents the connectivity
                relation between a network service/function and
                a tunnel termination point on a TE node.";
             leaf id {
               type uint32;
               description "Identifies the termination entry.";
             }

             leaf service-function-id {
               type string;
               description
                 "Reference to a network service or a network
                  function.";
             }
             leaf sf-connection-point-id {
               type string;
               description
                 "Reference to a connection point on a network
```

```
                  service or a network function.";
             }
             leaf enabled {
               type boolean;
               description
                 "'true' if this connectivity entry is enabled.";
             }
             leaf direction {
               type connectivity-direction;
               description
                 "Indicates whether this connectivity is
                  uni-directional, or bi-directional. If the
                  relation is uni-directional, the value of
                  this leaf indicates the direction.";
             }
           } // link-termination
         }
       }
     } // service-function-ttp-augmentation

     grouping sf-topology-type {
       description
         "Identifies the SF aware TE topology type.";
       container sf {
         presence "Indidates that the TE topology is SF aware.";
         description
           "Its presence identifies that the TE topology is SF aware.";
       }
     } // sf-topology-type

     /*
      * Augmentations
      */
     /* Augmentations to network-types/te-topology */
     augment "/nw:networks/nw:network/nw:network-types/"
       + "tet:te-topology" {
       description
         "Defines the SF aware TE topology type.";
       uses sf-topology-type;
     }

     /* Augmentations to te-node-attributes */
     augment "/nw:networks/nw:network/nw:node/tet:te/"
       + "tet:te-node-attributes" {
       description
         "Parameters for SF aware TE topology.";
       uses service-function-node-augmentation;
     }
```

```
     augment "/nw:networks/nw:network/nw:node/tet:te/"
           + "tet:information-source-entry" {
       description
         "Parameters for SF aware TE topology.";
       uses service-function-node-augmentation;
     }

     /* Augmentations to tunnel-termination-point */
     augment "/nw:networks/nw:network/nw:node/tet:te/"
       + "tet:tunnel-termination-point" {
       description
         "Parameters for SF aware TE topology.";
       uses service-function-ttp-augmentation;
     }

     /* Augmentations to connectivity-matrix */
     augment "/nw:networks/nw:network/nw:node/tet:te/"
       + "tet:te-node-attributes/tet-sf:service-function/"
       + "tet-sf:link-terminations/tet-sf:link-termination/"
       + "tet-sf:from" {
       description
         "Add reference to the link termination point.
          This portion cannot be shared with the state module.";
       leaf tp-ref {
         type leafref {
           path "../../../../../../../nt:termination-point/"
             + "nt:tp-id";
         }
         description
           "Reference to the link termination point.";
       }
     }
   }
   <CODE ENDS>
```

## 5.  Model Structure

```
   module: ietf-cso-dc
       +--rw cso
          +--rw dc* [id]
          |  +--rw hypervisor* [id]
          |  |  +--rw ram
          |  |  |  +--rw total?   uint32
          |  |  |  +--rw used?    uint32
          |  |  |  +--rw free?    uint32
          |  |  +--rw disk
```

```
       |  |  |  +--rw total?   uint32
       |  |  |  +--rw used?    uint32
       |  |  |  +--rw free?    uint32
       |  |  +--rw vcpu
       |  |  |  +--rw total?   uint16
       |  |  |  +--rw used?    uint16
       |  |  |  +--rw free?    uint16
       |  |  +--rw instance*   -> /cso/dc/instance/id
       |  |  +--rw id          string
       |  |  +--rw name?       string
       |  +--rw instance* [id]
       |  |  +--rw flavor
       |  |  |  +--rw disk?    uint32
       |  |  |  +--rw ram?     uint32
       |  |  |  +--rw vcpus?   uint16
       |  |  |  +--rw id?      string
       |  |  |  +--rw name?    string
       |  |  +--rw image
       |  |  |  +--rw checksum    string
       |  |  |  +--rw size        uint32
       |  |  |  +--rw format
       |  |  |  |  +--rw container?   enumeration
       |  |  |  |  +--rw disk?        enumeration
       |  |  |  +--rw id?         string
       |  |  |  +--rw name?       string
       |  |  +--rw hypervisor?   -> /cso/dc/hypervisor/id
       |  |  +--rw port*         -> /cso/dc/network/subnetwork/port
/id
       |  |  +--rw project?      string
       |  |  +--rw status?       enumeration
       |  |  +--rw id            string
       |  |  +--rw name?         string
       |  +--rw image* [id]
       |  |  +--rw checksum    string
       |  |  +--rw size        uint32
       |  |  +--rw format
       |  |  |  +--rw container?   enumeration
       |  |  |  +--rw disk?        enumeration
       |  |  +--rw id          string
       |  |  +--rw name?       string
       |  +--rw flavor* [id]
       |  |  +--rw disk?    uint32
       |  |  +--rw ram?     uint32
       |  |  +--rw vcpus?   uint16
       |  |  +--rw id       string
       |  |  +--rw name?    string
       |  +--rw dc-monitoring-param* [name]
       |  |  +--rw name            string
```

```
          |  |  +--rw value-string?   string
          |  +--rw network* [id]
          |  |  +--rw subnetwork* [id]
          |  |  |  +--rw port* [id]
          |  |  |  |  +--rw ip-address?   inet:ip-address
          |  |  |  |  +--rw instance?     -> /cso/dc/instance/id
          |  |  |  |  +--rw project?      string
          |  |  |  |  +--rw status?       enumeration
          |  |  |  |  +--rw id            string
          |  |  |  |  +--rw name?         string
          |  |  |  +--rw project?   string
          |  |  |  +--rw status?    enumeration
          |  |  |  +--rw id         string
          |  |  |  +--rw name?      string
          |  |  +--rw dhcp-agent* [id]
          |  |  |  +--rw enabled?   boolean
          |  |  |  +--rw pools* [ip-address]
          |  |  |  |  +--rw ip-address    inet:ip-address
          |  |  |  +--rw project?   string
          |  |  |  +--rw status?    enumeration
          |  |  |  +--rw id         string
          |  |  |  +--rw name?      string
          |  |  +--rw project?      string
          |  |  +--rw status?       enumeration
          |  |  +--rw id            string
          |  |  +--rw name?         string
          |  |  +--rw cso-ref?      -> /cso/cso-id
          |  +--rw ap*                    -> /actn-vn:actn/ap
   /access-point-list/access-point-id
          |  +--rw cso-ref?               -> /cso/cso-id
          |  +--rw id                     string
          |  +--rw name?                  string
          +--rw cso-id?   string
```

## 6. YANG Modules

```
   <CODE BEGINS> file "ietf-cso-dc@2017-01-16.yang"
   module ietf-cso-dc
   {
     namespace "urn:ietf:params:xml:ns:yang:ietf-cso-dc";
     prefix "dc";

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

```
     import ietf-actn-vn {
       prefix "actn-vn";
     }

     revision 2017-01-16 {
       description
         "Initial revision. This YANG file defines
          the reusable base types for CSO DC description.";
       reference
         "Derived from earlier versions of base YANG files";
     }

     // Abstract models
     grouping resource-element {
       leaf id { type string; }
       leaf name { type string; }
     }

     grouping resource-instance {
       leaf project{ type string; }
       leaf status {
         type enumeration {
           enum active;
           enum inactive;
           enum pending;
         }
       }
       uses resource-element;
     }

     // Compute models
     grouping format {
       leaf container {
         type enumeration {
           enum ami;
           enum ari;
           enum aki;
           enum bare;
           enum ovf;
         }
         default bare;
       }
       leaf disk {
         type enumeration {
           enum ami;
           enum ari;
           enum aki;
           enum vhd;
```

```
            enum vmdk;
            enum raw;
            enum qcow2;
            enum vdi;
            enum iso;
          }
          default qcow2;
        }
      }

      grouping image {
        leaf checksum { type string; mandatory true; }
        leaf size { type uint32; units 'Bytes'; mandatory true; }

        container format {
          uses format;
        }

        uses resource-element;
      }

      grouping flavor {
        leaf disk  { type uint32; units 'GB'; default 0; }
        leaf ram   { type uint32; units 'MB'; default 0; }
        leaf vcpus { type uint16; default 0; }
        uses resource-element;
      }

      grouping ram {
        leaf total { type uint32; units 'MB'; }
        leaf used { type uint32; units 'MB'; }
        leaf free { type uint32; units 'MB'; }
      }

      grouping disk {
        leaf total { type uint32; units 'GB'; }
        leaf used { type uint32; units 'GB'; }
        leaf free { type uint32; units 'GB'; }
      }

      grouping vcpu {
        leaf total { type uint16; }
        leaf used { type uint16; }
        leaf free { type uint16; }
      }

      grouping hypervisor {
```

```
       container ram {
         uses ram;
       }

       container disk {
         uses disk;
       }

       container vcpu {
         uses vcpu;
       }

       leaf-list instance {
         type leafref { path '/cso/dc/instance/id'; } }
       uses resource-element;
     }

     grouping instance {
       container flavor { uses flavor; }
       container image { uses image; }
       leaf hypervisor {
         type leafref { path '/cso/dc/hypervisor/id'; } }
       leaf-list port { type leafref {
           path '/cso/dc/network/subnetwork/port/id'; } }
       uses resource-instance;
     }

     grouping dc-monitoring-param {
       leaf name {
         description "dc-monitoring-param identifier"; type string; }
       leaf value-string {
         description
           "Current value for a string parameter";
         type string;
       }
     }

     grouping dc {

       list hypervisor {
         key id;
         uses hypervisor;
       }

       list instance {
         key id;
         uses instance;
       }
```

```
       list image {
         key id;
         uses image;
       }

       list flavor {
         key id;
         uses flavor;
       }

       list dc-monitoring-param {
           key "name";
           uses dc-monitoring-param;
       }

       list network {
           key id;
           uses network;
       }

       leaf-list ap { type leafref {
           path
             '/actn-vn:actn/actn-vn:ap/actn-vn:access-point-list/'
             + 'actn-vn:access-point-id';
         }
       }
       leaf cso-ref { type leafref { path "/cso/cso-id"; } }
       uses resource-element;
     }



     container cso {
       list dc {
           key id;
           uses dc;
       }

       leaf cso-id { type string; }
     }

     // Network models
     grouping ip-address {
       leaf ip-address { type inet:ip-address; }
     }

     grouping dhcp-agent {
       leaf enabled { type boolean; }
```

```
       list pools {
         key ip-address;
         uses ip-address;
       }
       uses resource-instance;
     }

     grouping network {
       list subnetwork {
         key id;
         uses subnetwork;
       }
       list dhcp-agent {
         key id;
         uses dhcp-agent;
       }
       uses resource-instance;
       leaf cso-ref { type leafref { path "/cso/cso-id"; } }
     }

     grouping subnetwork {
       list port {
         key id;
         uses port;
       }
       uses resource-instance;
     }

     grouping port {
       leaf ip-address { type inet:ip-address; }
       leaf instance { type leafref { path '/cso/dc/instance/id'; } }
       uses resource-instance;
     }



   }
   <CODE ENDS>
```

## 7. IANA Considerations

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

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

   --------------------------------------------------------------------
   URI: urn:ietf:params:xml:ns:yang:ietf-te-topology-sf-state
   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-te-topology-sf
   namespace:    urn:ietf:params:xml:ns:yang:ietf-te-topology-packet
   prefix:       tet-sf
   reference:    RFC XXXX
   --------------------------------------------------------------------

   --------------------------------------------------------------------
   name:         ietf-te-topology-sf-state
   namespace:  urn:ietf:params:xml:ns:yang:ietf-te-topology-packet-state
   prefix:       tet-sf-s
   reference:    RFC XXXX
   --------------------------------------------------------------------
```

## 8.  Security Considerations

The configuration, state, action and notification data defined in this document are designed to be accessed via the NETCONF protocol [RFC6241].  The data-model by itself does not create any security implications.  The security considerations for the NETCONF protocol are applicable.  The NETCONF protocol used for sending the data supports authentication and encryption.

## 9.  References

### 9.1.  Normative References

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

[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>.

[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>.

[ETSI-NFV-MAN]
ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration", ETSI GS NFV-MAN 001 V1.1.1, December 2014.

[I-D.ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo-20 (work in progress), December 2017.

[I-D.ietf-teas-yang-te-topo]
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", draft-ietf-teas-yang-te-topo-18 (work in progress), June 2018.

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-10 (work in progress), January 2018.

[ETSI-NFV-TERM]
ETSI, "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV", ETSI GS NFV 003 V1.2.1, December 2014.

[I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-16 (work in progress), July 2018.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <https://www.rfc-editor.org/info/rfc3022>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[I-D.ietf-sfc-hierarchical]
Dolson, D., Homma, S., Lopez, D., and M. Boucadair, "Hierarchical Service Function Chaining (hSFC)", draft-ietf-sfc-hierarchical-11 (work in progress), June 2018.

[I-D.defoy-netslices-3gpp-network-slicing]
Foy, X. and A. Rahman, "Network Slicing - 3GPP Use Case", draft-defoy-netslices-3gpp-network-slicing-02 (work in progress), October 2017.

[_3GPP.28.801]
3GPP, "Study on management and orchestration of network slicing for next generation network", 3GPP TR 28.801 V2.0.0, September 2017, <http://www.3gpp.org/ftp/Specs/html-info/28801.htm>.

[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>.

## 9.2. Informative References

[RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <https://www.rfc-editor.org/info/rfc7498>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-06 (work in progress), February 2018.

## Appendix A.  Companion YANG Model for Non-NMDA Compliant Implementations

The YANG module ietf-te-topology-sf defined in this document is designed to be used in conjunction with implementations that support the Network Management Datastore Architecture (NMDA) defined in [I-D.ietf-netmod-revised-datastores].  In order to allow implementations to use the model even in cases when NMDA is not supported, the following companion module, ietf-te-topology-sf-state, is defined as state model, which mirrors the module ietf-te-topology-sf defined earlier in this document.  However, all data nodes in the companion module are non-configurable, to represent the applied configuration or the derived operational states.

The companion module, ietf-te-topology-sf-state, is redundant and SHOULD NOT be supported by implementations that support NMDA.

As the structure of the companion module mirrors that of the cooresspinding NMDA model, the YANG tree of the companion module is not depicted separately.

### A.1.  SF Aware TE Topology State Module

```
<CODE BEGINS> file "ietf-te-topology-sf-state@2018-02-27.yang"
module ietf-te-topology-sf-state {
  yang-version 1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-topology-sf-state";

  prefix "tet-sf-s";

  import ietf-te-topology-sf {
    prefix "tet-sf";
  }

  import ietf-network-state {
    prefix "nw-s";
  }

  import ietf-network-topology-state {
    prefix "nt-s";
  }

  import ietf-te-topology-state {
    prefix "tet-s";
  }

  organization
    "Traffic Engineering Architecture and Signaling (TEAS)
```

```
        Working Group";

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

        Editors:  Igor Bryskin
                  <mailto:Igor.Bryskin@huawei.com>

                  Xufeng Liu
                  <mailto:Xufeng_Liu@jabil.com>";

     description
       "Network service and function aware aware TE topology operational
        state model for non-NMDA compliant implementations.

        Copyright (c) 2018 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

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

     revision 2018-02-27 {
       description "Initial revision";
       reference "TBD";
     }

     /*
      * Augmentations
      */
     /* Augmentations to network-types/te-topology */
     augment "/nw-s:networks/nw-s:network/nw-s:network-types/"
       + "tet-s:te-topology" {
       description
         "Defines the SF aware TE topology type.";
       uses tet-sf:sf-topology-type;
     }

     /* Augmentations to connectivity-matrix */
     augment "/nw-s:networks/nw-s:network/nw-s:node/tet-s:te/"
       + "tet-s:te-node-attributes" {
       description
         "Parameters for SF aware TE topology.";
       uses tet-sf:service-function-node-augmentation;
```

```
     }

     augment "/nw-s:networks/nw-s:network/nw-s:node/tet-s:te/"
           + "tet-s:information-source-entry" {
       description
         "Parameters for SF aware TE topology.";
       uses tet-sf:service-function-node-augmentation;
     }

     /* Augmentations to tunnel-termination-point */
     augment "/nw-s:networks/nw-s:network/nw-s:node/tet-s:te/"
       + "tet-s:tunnel-termination-point" {
       description
         "Parameters for SF aware TE topology.";
       uses tet-sf:service-function-ttp-augmentation;
     }

     /* Augmentations to connectivity-matrix */
     augment "/nw-s:networks/nw-s:network/nw-s:node/tet-s:te/"
       + "tet-s:te-node-attributes/tet-sf-s:service-function/"
       + "tet-sf-s:link-terminations/tet-sf-s:link-termination/"
       + "tet-sf-s:from" {
       description
         "Add reference to the link termination point.
          This portion cannot be shared with the state module.";
       leaf tp-ref {
         type leafref {
           path "../../../../../../../nt-s:termination-point/"
             + "nt-s:tp-id";
         }
         description
           "Reference to the link termination point.";
       }
     }
   }
   <CODE ENDS>
```

## Appendix B.  Data Examples

### B.1.  A Topology with Multiple Connected Network Functions

```
                         Node-1
      +----o--o--------------------------o-------+
      |    |  |                          |       |
      |    \__/                          \__     |
      |    *\/ TTP-1   * * * * * * * * * *\/*    |
LTP-4 |* * *         *                TTP-2  *   | LTP-1
      o------------*-----------------------------o
      |          *                             * |
LTP-3 |* * * * *                                *| LTP-2
      o---                                  -----o
      |   \                                /     |
      |    \                              /      |
      |     \ CP01                   CP02/       |
      | +----o--------------------------o------+ |
      | | VL1|                       VL4|      | |
      | |    |CP11                      |CP33  | |
      | |  +-o--+        +----+       +-o--+   | |
      | |  |VNF1|        |VNF2|       |VNF3|   | |
      | |  +-o-o+  VL2   +--o-+  VL2  +-o-o+   | |
      | |CP12| |\----------/ \---------/| |CP32| |
      | |    | |CP13      CP21      CP31| |    | |
      | |    | |      VL2               | |    | |
      | |    | +------------------------+ |    | |
      | |    +----------------------------+    | |
      | |             VL3                      | |
      | |                  Network Service 1   | |
      | +--------------------------------------+ |
      +------------------------------------------+
```

The configuration instance data for Node-1 in the above figure could be as follows:

```
{
  "networks": {
    "network": [
      {
        "network-types": {
          "te-topology": {
            "sf": {}
          }
        },
        "network-id": "network-sf-aware",
        "provider-id": 201,
        "client-id": 300,
        "te-topology-id": "te-topology:network-sf-aware",
        "node": [
          {
            "node-id": "Node-1",
```

```
               "te-node-id": "2.0.1.1",
               "te": {
                 "te-node-attributes": {
                   "domain-id": 1,
                   "is-abstract": [null],
                   "connectivity-matrices": {
                   },
                   "service-function": {
                     "connectivity-matrices": {
                       "connectivity-matrix": [
                         {
                           "id": 10,
                           "from": {
                             "service-function-id": "Network Service 1",
                             "sf-connection-point-id": "CP01"
                           },
                           "to": {
                             "service-function-id": "VNF1",
                             "sf-connection-point-id": "CP11"
                           }
                           "direction": "bidir",
                           "virtual-link-id": "VL1"
                         },
                         {
                           "id": 13,
                           "from": {
                             "service-function-id": "VNF1",
                             "sf-connection-point-id": "CP12"
                           },
                           "to": {
                             "service-function-id": "VNF3",
                             "sf-connection-point-id": "CP32"
                           }
                           "direction": "bidir",
                           "virtual-link-id": "VL3"
                         },
                         {
                           "id": 12,
                           "from": {
                             "service-function-id": "VNF1",
                             "sf-connection-point-id": "CP13"
                           },
                           "to": {
                             "service-function-id": "VNF2",
                             "sf-connection-point-id": "CP21"
                           }
                           "direction": "bidir",
                           "virtual-link-id": "VL2"
```

```
                                 },
                                 {
                                   "id": 23,
                                   "from": {
                                     "service-function-id": "VNF2",
                                     "sf-connection-point-id": "CP21"
                                   },
                                   "to": {
                                     "service-function-id": "VNF3"
                                     "sf-connection-point-id": "CP31"
                                   }
                                   "direction": "bidir",
                                   "virtual-link-id": "VL2"
                                 },
                                 {
                                   "id": 30,
                                   "from": {
                                     "service-function-id": "Network Service 1",
                                     "sf-connection-point-id": "CP02"
                                   },
                                   "to": {
                                     "service-function-id": "VNF3",
                                     "sf-connection-point-id": "CP33"
                                   }
                                   "direction": "bidir",
                                   "virtual-link-id": "VL4"
                                 }
                               ]
                             },
                             "link-terminations": {
                               "link-termination": [
                                 {
                                   "id": 2,
                                   "from": {
                                     "tp-ref": "LTP-2"
                                   },
                                   "to": {
                                     "service-function-id": "Network Service 1",
                                     "sf-connection-point-id": "CP02"
                                   }
                                   "direction": "bidir"
                                 },
                                 {
                                   "id": 3,
                                   "from": {
                                     "tp-ref": "LTP-3"
                                   },
                                   "to": {
```

```
                             "service-function-id": "Network Service 1",
                             "sf-connection-point-id": "CP01"
                           }
                           "direction": "bidir"
                         }
                       ]
                     }
                   }
                 }
                 "tunnel-termination-point": [
                   {
                     "tunnel-tp-id": 10001,
                     "name": "TTP-1",
                     "service-function-terminations": {
                     }
                   },
                   {
                     "tunnel-tp-id": 10002,
                     "name": "TTP-2",
                     "service-function-terminations": {
                     }
                   }
                 ]
               },
               "termination-point": [
                 {
                   "tp-id": "LTP-1",
                   "te-tp-id": 10001
                   "te": {
                     "interface-switching-capability": [
                       {
                         "switching-capability": "switching-l2sc",
                         "encoding": "lsp-encoding-ethernet"
                       }
                     ]
                   }
                 },
                 {
                   "tp-id": "LTP-2",
                   "te-tp-id": 10002
                   "te": {
                     "interface-switching-capability": [
                       {
                         "switching-capability": "switching-l2sc",
                         "encoding": "lsp-encoding-ethernet"
                       }
                     ]
                   }
```

```
                 },
                 {
                   "tp-id": "LTP-3",
                   "te-tp-id": 10003
                   "te": {
                     "interface-switching-capability": [
                       {
                         "switching-capability": "switching-l2sc",
                         "encoding": "lsp-encoding-ethernet"
                       }
                     ]
                   }
                 },
                 {
                   "tp-id": "LTP-4",
                   "te-tp-id": 10004
                   "te": {
                     "interface-switching-capability": [
                       {
                         "switching-capability": "switching-l2sc",
                         "encoding": "lsp-encoding-ethernet"
                       }
                     ]
                   }
                 }
               ]
             }
           ]
         }
       ]
     }
   }
```

## B.2. A Topology with an Encapsulated Network Service

In this example, a network service consists of several inter-connected network functions (NFs), and is represented by this model as an encapsulated opaque object without the details between its internals.

```
                            Node-1
         +----o--o--------------------------o-------+
         |    |  |                          |       |
         |    \__/                          \__     |
         |    *\/ TTP-1   * * * * * * * * * *\/*    |
   LTP-4 |* * *         *                TTP-2  *   | LTP-1
         o------------*-----------------------------o
         |          *                             * |
   LTP-3 |* * * * *                                *| LTP-2
         o---                                  -----o
         |   \                                /     |
         |    \                              /      |
         |     \ CP01                   CP02/       |
         | +----o--------------------------o------+ |
         | |                                      | |
         | |                  Network Service 1   | |
         | +--------------------------------------+ |
         +------------------------------------------+
```

The configuration instance data for Node-1 in the above figure could be as follows:

```
   {
     "networks": {
       "network": [
         {
           "network-types": {
             "te-topology": {
               "sf": {}
             }
           },
           "network-id": "network-sf-aware",
           "provider-id": 201,
           "client-id": 300,
           "te-topology-id": "te-topology:network-sf-aware",
           "node": [
             {
               "node-id": "Node-1",
               "te-node-id": "2.0.1.1",
               "te": {
                 "te-node-attributes": {
                   "domain-id": 1,
                   "is-abstract": [null],
                   "connectivity-matrices": {
                   },
                   "service-function": {
                     "connectivity-matrices": {
                     },
```

```
                    "link-terminations": {
                      "link-termination": [
                        {
                          "id": 2,
                          "from": {
                            "tp-ref": "LTP-2"
                          },
                          "to": {
                            "service-function-id": "Network Service 1",
                            "sf-connection-point-id": "CP02"
                          }
                          "direction": "bidir"
                        },
                        {
                          "id": 3,
                          "from": {
                            "tp-ref": "LTP-3"
                          },
                          "to": {
                            "service-function-id": "Network Service 1",
                            "sf-connection-point-id": "CP01"
                          }
                          "direction": "bidir"
                        }
                      ]
                    }
                  }
                }
                "tunnel-termination-point": [
                  {
                    "tunnel-tp-id": 10001,
                    "name": "TTP-1",
                    "service-function-terminations": {
                    }
                  },
                  {
                    "tunnel-tp-id": 10002,
                    "name": "TTP-2",
                    "service-function-terminations": {
                    }
                  }
                ]
              },
              "termination-point": [
                {
                  "tp-id": "LTP-1",
                  "te-tp-id": 10001
                  "te": {
```

```
                  "interface-switching-capability": [
                    {
                      "switching-capability": "switching-l2sc",
                      "encoding": "lsp-encoding-ethernet"
                    }
                  ]
                }
              },
              {
                "tp-id": "LTP-2",
                "te-tp-id": 10002
                "te": {
                  "interface-switching-capability": [
                    {
                      "switching-capability": "switching-l2sc",
                      "encoding": "lsp-encoding-ethernet"
                    }
                  ]
                }
              },
              {
                "tp-id": "LTP-3",
                "te-tp-id": 10003
                "te": {
                  "interface-switching-capability": [
                    {
                      "switching-capability": "switching-l2sc",
                      "encoding": "lsp-encoding-ethernet"
                    }
                  ]
                }
              },
              {
                "tp-id": "LTP-4",
                "te-tp-id": 10004
                "te": {
                  "interface-switching-capability": [
                    {
                      "switching-capability": "switching-l2sc",
                      "encoding": "lsp-encoding-ethernet"
                    }
                  ]
                }
              }
            ]
          }
        ]
      }
```

```
       ]
     }
   }
```

## Appendix C.  Use Cases for SF Aware Topology Models

### C.1.  **Exporting SF/NF Information to Network Clients and Other Network** SDN Controllers

In the context of Service Function Chain (SFC) orchestration one existing problem is that there is no way to formally describe a Service or Network Function in a standard way (recognizable/ understood by a third party) as a resource of a network topology node.

One implication of this is that there is no way for the orchestrator to give a network client even a ball-park idea as to which network's SFs/NFs are available for the client's use/control and where they are located in the network even in terms of abstract topologies/virtual networks configured and managed specifically for the client. Consequently, the client has no say on how the SFCs provided for the client by the network should be set up and managed (which SFs are to be used and how they should be chained together, optimized, manipulated, protected, etc.).

Likewise, there is no way for the orchestrator to export SF/NF information to other network controllers.  The SFC orchestrator may serve, for example, a higher level controller (such as Network Slicing Orchestrator), with the latter wanting at least some level of control as to which SFs/NFs it wants on its SFCs and how the Service Function Paths (SFPs) are to be routed and provisioned, especially, if it uses services of more than one SFC orchestrator.

The issue of exporting of SF/NF information could be addressed by defining a model, in which formally described/recognizable SF/NF instances are presented as topological elements, for example, hosted by TE, L3 or L2 topology nodes (see Figure 1).  The model could describe whether, how and at what costs the SFs/NFs hosted by a given node could be chained together, how these intra-node SFCs could be connected to the node's Service Function Forwarders (SFFs, entities dealing with SFC NSHs and metadata), and how the SFFs could be connected to the node's Tunnel and Link Termination Points (TTPs and LTPs) to chain the intra-node SFCs across the network topology.

Figure 1: SF/NF aware TE topology

## C.2. Flat End-to-end SFCs Managed on Multi-domain Networks

SFCs may span multiple administrative domains, each of which controlled by a separate SFC controller. The usual solution for such a scenario is the Hierarchical SFCs (H-SFCs), in which the higher level orchestrator controls only SFs located on domain border nodes. Said higher level SFs are chained together into higher level SFCs via lower level (intra-domain) SFCs provisioned and controlled independently by respective domain controllers. The decision as to which higher level SFCs are connected to which lower level SFCs is driven by packet re-classification every time the packet enters a given domain. Said packet re-classification is a very time-consuming operation. Furthermore, the independent nature of higher and lower level SFC control is prone to configuration errors, which may lead to long lasting loops and congestions. It is highly desirable to be able to set up and manage SFCs spanning multiple domains in a flat way as far as the data plane is concerned (i.e. with a single packet classification at the ingress into the multi-domain network but without re-classifications on domain ingress nodes).

One way to achieve this is to have the domain controllers expose SF/ NF- aware topologies, and have the higher level orchestrator operate on the network-wide topology, the product of merging of the topologies catered by the domain controllers. This is similar in spirit to setting up, coordinating and managing the transport connectivity (TE tunnels) on a multi-domain multi-vendor transport network.

## C.3. Managing SFCs with TE Constraints

Some SFCs require per SFC link/element and end-to-end TE constrains (bandwidth, delay/jitter, fate sharing/diversity. etc.). Said constraints could be ensured via carrying SFPs inside overlays that are traffic engineered with the constrains in mind. A good analogy would be orchestrating delay constrained L3 VPNs. One way to support such L3 VPNs is to carry MPLS LSPs interconnecting per-VPN VRFs inside delay constrained TE tunnels interconnecting the PEs hosting the VRFs.

Figure 2: L3 VPN with delay constraints

Planning, computing and provisioning of TE overlays to constrain arbitrary SFCs, especially those that span multiple administrative domains with each domain controlled by a separate controller, is a very difficult challenge. Currently it is addressed by pre-provisioning on the network of multiple TE tunnels with various TE characteristics, and "nailing down" SFs/NFs to "strategic" locations (e.g. nodes terminating many of such tunnels) in a hope that an adequate set of tunnels could be found to carry the SFP of a given TE-constrained SFC. Such an approach is especially awkward in the case when some or all of the SFs/NFs are VNFs (i.e. could be instantiated at multiple network locations).

SF/NF-aware TE topology model in combination with TE tunnel model will allow for the network orchestrator (or a client controller) to compute, set up and manipulate the TE overlays in the form of TE tunnel chains (see Figure 3).

Said chains could be duel-optimized compromising on optimal SF/NF locations with optimal TE tunnels interconnecting them.  The TE tunnel chains (carrying multiple similarly constrained SFPs) could be adequately constrained both at individual TE tunnel level and at the chain end-to-end level.

Figure 3: SFC with TE constraints

## C.4.  SFC Protection and Load Balancing

Currently the combination of TE topology & tunnel models offers to a network controller various capabilities to recover an individual TE tunnel from network failures occurred on one or more network links or transit nodes on the TE paths taken by the TE tunnel's connection(s). However, there is no simple way to recover a TE tunnel from a failure affecting its source or destination node.  SF/NF-aware TE topology model can decouple the association of a given SF/NF with its location on the network topology by presenting multiple, identifiable as mutually substitutable SFs/NFs hosted by different TE topology nodes. So, for example, if it is detected that a given TE tunnel destination node is malfunctioning or has gone out of service, the TE tunnel could be re-routed to terminate on a different node hosting functionally the same SFs/NFs as ones hosted by the failed node (see Figures 6).

This is in line with the ACTN edge migration and function mobility concepts [RFC8453]. It is important to note that the described strategy works much better for the stateless SFs/NFs. This is because getting the alternative stateful SFs/NFs into the same respective states as the current (i.e. active, affected by failure) are is a very difficult challenge.

Figure 4: SFC recovery: SF2 on node NE1 fails

At the SFC level the SF/NF-aware TE topology model can offer SFC dynamic restoration capabilities against failed/malfunctioning SFs/NFs by identifying and provisioning detours to a TE tunnel chain, so that it starts carrying the SFC's SFPs towards healthy SFs/NFs that are functionally the same as the failed ones. Furthermore, multiple parallel TE tunnel chains could be pre-provisioned for the purpose of SFC load balancing and end-to-end protection. In the latter case said parallel TE tunnel chains could be placed to be sufficiently disjoint from each other.

Figure 5: SFC recovery: SFC SF1-SF2-SF6 is recovered after SF2 on node N1 has failed

Figure 6: SFC recovery: SFC SF1-SF2-SF6 is recovered after node N1 has failed

## C.5. Network Clock Synchronization

Many current and future network applications (including 5g and IoT applications) require very accurate time services (PTP level, ns resolution). One way to implement the adequate network clock synchronization for such services is via describing network clocks as NFs on an NF-aware TE topology optimized to have best possible delay variation characteristics. Because such a topology will contain delay/delay variation metrics of topology links and node cross-connects, as well as costs in terms of delay/delay variation of connecting clocks to hosting them node link and tunnel termination points, it will be possible to dynamically select and provision bi-directional time-constrained deterministic paths or trees connecting clocks (e.g. grand master and boundary clocks) for the purpose of exchange of clock synchronization information. Note that network clock aware TE topologies separately provided by domain controllers will enable multi-domain network orchestrator to set up and manipulate the clock synchronization paths/trees spanning multiple network domains.

## C.6. Client - Provider Network Slicing Interface

3GPP defines network slice as "a set of network functions and the resources for these network functions which are arranged and configured, forming a complete logical network to meet certain network characteristics" [I-D.defoy-netslices-3gpp-network-slicing] [ 3GPP.28.801]. Network slice could be also defined as a logical partition of a provider's network that is owned and managed by a tenant. SF/NF-aware TE topology model has a potential to support a very important interface between network slicing clients and providers because, on the one hand, the model can describe holistically and hierarchically the client's requirements and preferences with respect to a network slice functional, topological and traffic engineering aspects, as well as of the degree of resource separation/ sharing between the slices, thus allowing for the client (up to agreed upon extent) to dynamically (re-)configure the slice or (re-)schedule said (re-)configurations in time, while, on the other hand, allowing for the provider to convey to the client the slice's operational state information and telemetry the client has expressed interest in.

## C.7. Dynamic Assignment of Regenerators for L0 Services

On large optical networks, some of provided to their clients L0 services could not be provisioned as single OCh trails, rather, as chains of such trails interconnected via regenerators, such as 3R regenerators. Current practice of the provisioning of such services requires configuration of explicit paths (EROs) describing identity

and location of regenerators to be used.  A solution is highly desirable that could:

- Identify such services based, for example, on optical impairment computations;

- Assign adequate for the services regenerators dynamically out of the regenerators that are grouped together in pools and strategically scattered over the network topology nodes;

- Compute and provision supporting the services chains of optical trails interconnected via so selected regenerators, optimizing the chains to use minimal number of regenerators, their optimal locations, as well as optimality of optical paths interconnecting them;

- Ensure recovery of such chains from any failures that could happen on links, nodes or regenerators along the chain path.

NF-aware TE topology model (in this case L1 NF-aware L0 topology model) is just the model that could provide a network controller (or even a client controller operating on abstract NF-aware topologies provided by the network) to realize described above computations and orchestrate the service provisioning and network failure recovery operations (see Figure 7).

Figure 7: Optical tunnel as TE-constrained SFC of 3R regenerators. Red trail (not regenerated) is not optically reachable, but blue trail (twice regenerated) is

## C.8. Dynamic Assignment of OAM Functions for L1 Services

OAM functionality is normally managed by configuring and manipulating TCM/MEP functions on network ports terminating connections or their segments over which OAM operations, such as performance monitoring, are required to be performed. In some layer networks (e.g. Ethernet) said TCMs/MEPs could be configured on any network ports. In others (e.g. OTN/ODUk) the TCMs/MEPs could be configured on some (but not all network ports) due to the fact that the OAM functionality (i.e. recognizing and processing of OAM messages, supporting OAM protocols and FSMs) requires in these layer networks certain support in the data plane, which is not available on all network nodes. This makes TCMs/MEPs good candidates to be modeled as NFs. This also makes TCM/MEP aware topology model a good basis for placing dynamically an ODUk connection to pass through optimal OAM locations without mandating the client to specify said locations explicitly.

Figure 8: Compute/storage resource aware topology

## C.9. SFC Abstraction and Scaling

SF/NF-aware topology may contain information on native SFs/NFs (i.e. SFs/NFs as known to the provider itself) and/or abstract SFs/NFs (i.e. logical/macro SFs/NFs representing one or more SFCs each made of native and/or lower level abstract SFs/NFs). As in the case of abstracting topology nodes, abstracting SFs/NFs is hierarchical in nature - the higher level of SF/NF-aware topology, the "larger" abstract SFs/NFs are, i.e. the larger data plane SFCs they represent. This allows for managing large scale networks with great number of SFs/NFs (such as Data Center interconnects) in a hierarchical, highly scalable manner resulting in control of very large number of flat in the data plane SFCs that span multiple domains.

## C.10. Dynamic Compute/VM/Storage Resource Assignment

In a distributed data center network, virtual machines for compute resources may need to be dynamically re-allocated due to various reasons such as DCI network failure, compute resource load balancing, etc. In many cases, the DCI connectivity for the source and the destination is not predetermined. There may be a pool of sources and a pool of destination data centers associated with re-allocation of compute/VM/storage resources. There is no good mechanism to date to capture this dynamicity nature of compute/VM/storage resource reallocation. Generic Compute/VM/Storage resources can be described and announced as a SF, where a DC hosting these resources can be modeled as an abstract node. Topology interconnecting these abstract nodes (DCs) in general is of multi-domain nature. Thus, SF-aware topology model can facilitate a joint optimization of TE network resources and Compute/VM/Storage resources and solve Compute/VM/ Storage mobility problem within and between DCs (see Figure 8).

## C.11. Application-aware Resource Operations and Management

Application stratum is the functional grouping which encompasses application resources and the control and management of these resources. These application resources are used along with network services to provide an application service to clients/end-users. Application resources are non-network resources critical to achieving the application service functionality. Examples of application resources include: caches, mirrors, application specific servers, content, large data sets, and computing power. Application service is a networked application offered to a variety of clients (e.g., server backup, VM migration, video cache, virtual network on-demand, 5G network slicing, etc.). The application servers that host these application resources can be modeled as an abstract node. There may be a variety of server types depending on the resources they host. Figure 9 shows one example application aware topology for video cache server distribution.

Figure 9: Application aware topology

## C.12. IANA Considerations

This document has no actions for IANA.

## C.13. Security Considerations

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

## C.14. Acknowledgements

The authors would like to thank Maarten Vissers, Joel Halpern, and Greg Mirsky for their helpful comments and valuable contributions.

Authors' Addresses

Igor Bryskin
Huawei Technologies

EMail: Igor.Bryskin@huawei.com

Xufeng Liu
Volta Networks

EMail: xufeng.liu.ietf@gmail.com

Young Lee
Huawei Technologies

EMail: leeyoung@huawei.com

Jim Guichard
Huawei Technologies

EMail: james.n.guichard@huawei.com

Luis Miguel Contreras Murillo
Telefonica

EMail: luismiguel.contrerasmurillo@telefonica.com

Daniele Ceccarelli
Ericsson

EMail: daniele.ceccarelli@ericsson.com

Jeff Tantsura
Apstra Networks

EMail: jefftant.ietf@gmail.com