CCAMP Working Group
Internet-Draft
Intended status: Standards Track

Xian Zhang
Huawei
R. Jing
China Telecom
W. Jian
China Unicom
Jeong-dong Ryoo
ETRI
Y. Xu
CAICT
Daniel King
Lancaster University

Expires: September 05, 2016 March 07, 2016

YANG Models for the Northbound Interface of a Transport Network Controller: Requirements, Functions, and a List of YANG Models

draft-zhang-ccamp-transport-ctrlnorth-yang-00.txt

Abstract

A transport network is a server-layer network designed to provide connectivity services for a client-layer network to carry the client traffic opaquely across the server-layer network resources. A transport network may be constructed from equipment utilizing any of a number of different transport technologies such as the evolving optical transport infrastructure (Synchronous Optical Networking (SONET) / Synchronous Digital Hierarchy (SDH) and Optical Transport Network (OTN)) or packet transport as epitomized by the MPLS Transport Profile (MPLS-TP).

All transport networks have high benchmarks for reliability and operational simplicity. This suggests a common, technology-independent management/control paradigm that can be extended to represent and configure specific technology attributes.

This document describes the requirements facing transport networks in order to provide open interfaces for resource programmability and control/management automation. A list of existing and additional YANG models is provided to fulfill the functional requirements.

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire on September 05, 2016.

Table of Contents

1. Introduction

A transport network is a server-layer network designed to provide connectivity services, or more advanced services like Virtual Private Networks (VPN) for a client-layer network to carry the client traffic opaquely across the server-layer network resources. It acts as a pipe provider for upper-layer networks, such as IP network and mobile networks.

Transport networks, such as Synchronous Optical Networking (SONET) / Synchronous Digital Hierarchy (SDH), Optical Transport Network (OTN), Wavelength Division Multiplexing (WDM), and flexi-grid networks, are often built using equipment from a single vendor and are managed using private interfaces to dedicated Element Management Systems (EMS) / Network Management Systems (NMS). All transport networks have high benchmarks for reliability and operational simplicity. This suggests a common, technology-independent management/control paradigm that is extended to represent and configure specific technology attributes.

The need of network providers to manage multi-vendor and multi-domain transport networks (where each domain is an island of equipment from a single supplier) has been further stressed by the expansion in network size. At the same time, applications such as data center interconnection require larger and more dynamic connectivity matrices. Therefore, transport networks face new challenges going beyond automatic provisioning of tunnel setup enabled by GMPLS (Generalized Multi-Protocol Label Switching) protocols to achieve automatic service provisioning, as well as

address opportunities enabled by partitioning the network through the process of resource slicing. With lower operational expenditure (OPEX) and capital expenditure (CAPEX) as the usual objectives, open interfaces to transport networks are considered by network providers as a way to meet these requirements. The concept of Software Defined Networking (SDN) leverages these ideas.

The YANG language [RFC6020] is currently the data modeling language of choice within the IETF and has been adopted by a number of industry-wide open management and control initiatives. YANG may be used to model both configuration and operational states; it is vendor-neutral and supports extensible APIs for control and management of elements.

This document analyzes typical scenarios that need transport network control/management openness, and lists functions desired to enable deployment. Moreover, a list of YANG models and their relationships have been identified that can help facilitate the deployment and operation of transport network open interfaces. Note that some of the models discussed meet the requirements described, and are already being developed in the IETF. Thus, this document provides a reference of existing models, and provides information of the missing ones which need further work.

2. Conventions used in this document

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

3. Terminology and Notations

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in the YANG data tree presented later in this draft is defined in [ietf-netmod-rfc6087bis]. They are provided below for reference.

- Brackets "[" and "]" enclose list keys.

- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).

- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

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

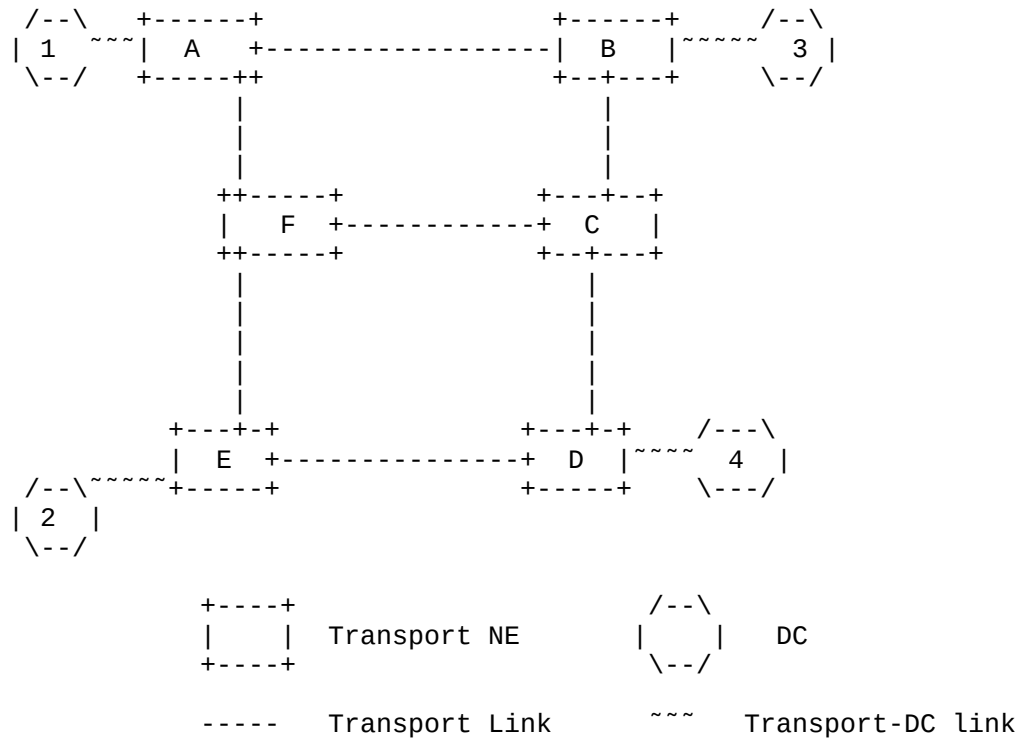
o Ellipsis ("...") stands for contents of subtrees that are not shown.

4. Functional Requirements

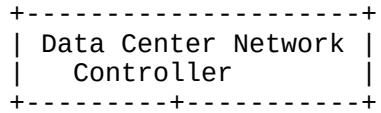
4.1. Scenarios

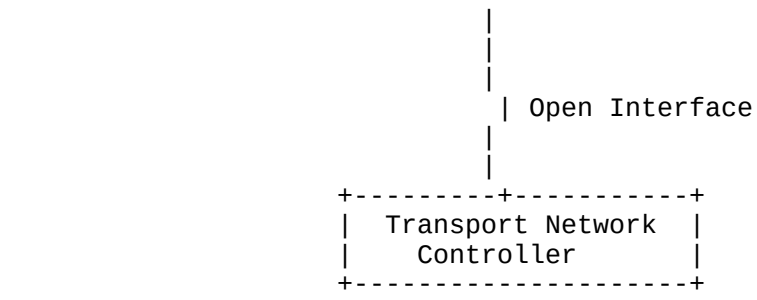
There are several scenarios where an open interface to access server-layer (transport) network resources would be useful. Here two typical scenarios are provided.

The first one is depicted as below (Figure 1):



(a) Data Centers interconnected via a transport network





(b) The controller architecture for data center interconnection

Figure 1: Scenario 1: Data centers interconnected via a transport network and the controller architecture

For the data center operator, assuming the objective is to trigger the transport network to provide connectivity on demand, the following capabilities, at a minimum, would be required on the open itnerface between the two contollers illustrated in Figure 1:

A: The ability to obtain information about a set of access points of the transport network facing the client side, including information such as access point identifiers, capabilities, etc.; for instance, transport-network-side end point identifiers related to the access link between DC1 and Transport NE A.

B: The capability to send a request for a service using the aforementioned access point information, as well as the ability to retrieve a list of service requests and their statuses. In this request, it should at least be possible to include source node, destination node, and requested bandwidth to request the transport network to set up tunnels/paths so as to provide the requested connectivitiy for the service request.

C: Note that in this case acquistion of the topology, be it physical or logical, of the transport network is not a compulsory requirement, but it may indeed be able to give data center providers more control over the transport resource usage. Furtheremore, the client controller can impose a virtual network of its own choice by requesting a slice of network resource with its choice of network parameters (such as network topology type, bandwidth etc.).

The second scenario, more complicated than the first, is depicted as below (Figure 2). In this example, we focus on the management and control via open interfaces for multi-domain networks with homogeneous technologies (such as OTN), but it can be extended

further to multi-domain networks with heterogeneous technologies with higher complexity.

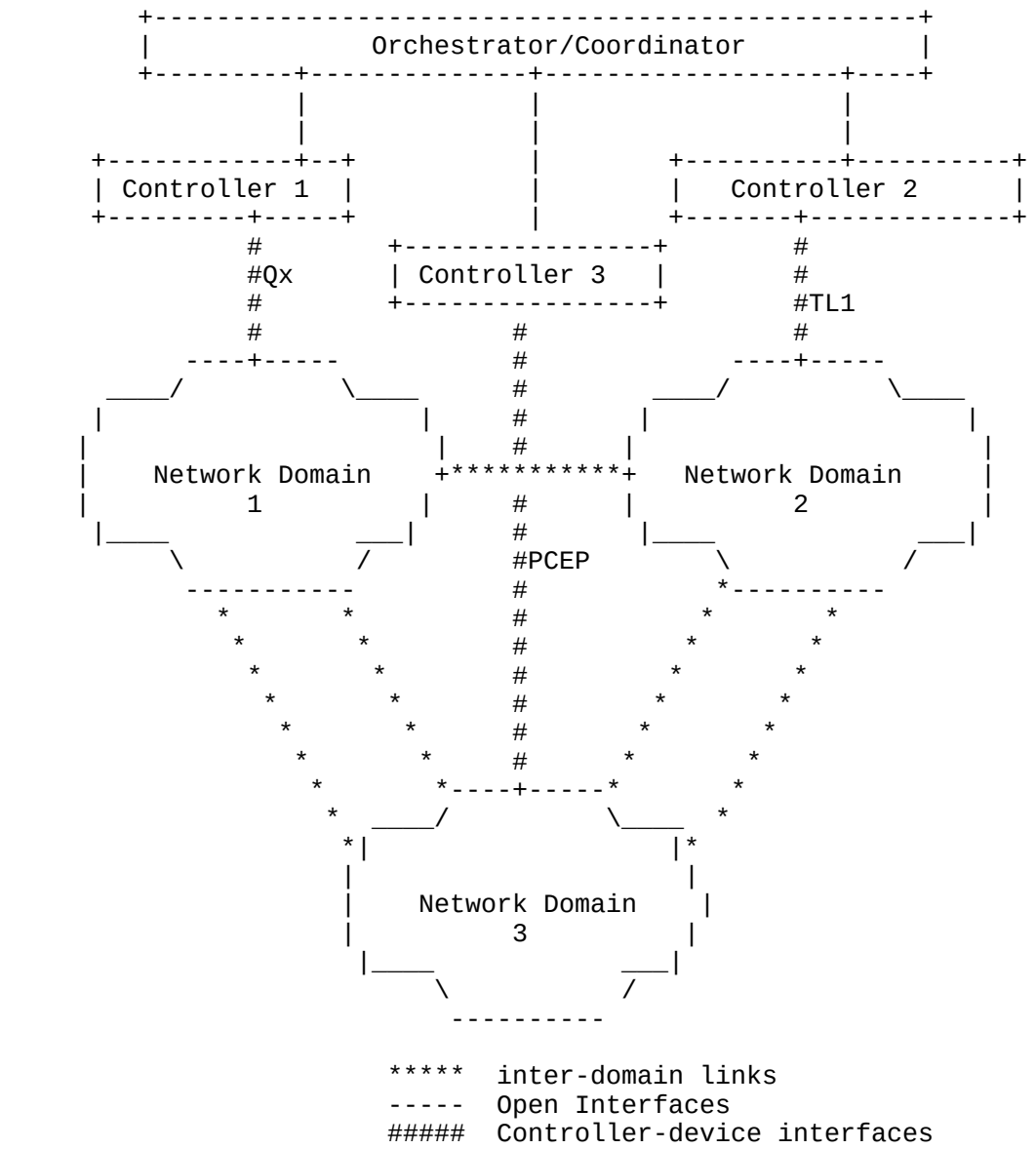


Figure 2: Scenario 2: Multi-domain network control and management

For the second scenario, the orchestrator/coordinator controls and manages three distinct network domains, each controlled/managed by their domain controller. In order to orchestrate across domains/layers, the orchestrator needs its interface between domain controllers to be equipped with the following functions:

A: Access to the topologies reported by each domain controller, including cross-domain links for the purpose of planning and requesting the paths of end-to-end tunnels. Multiple technologies within a domain (i.e., a multi-layer network), this might be reflected in the reported topology. Depending on the abstraction level of the reported topology, the orchestrator has different control granularities.

B: The ability to set up, delete and modify tunnels, be it within one domain or across multiple domains. Furthermore, it should have the abilty to view the tunnels created within each domain as well as thouse that cross domains as reported by each domain controller.

4.2. Function Requirement Summary

For the open interface of a transport controller towards a northbound client, five functions are derived from the scenarios explained in the last section. They are summarized in the table below and we also match these functions with YANG models that are being developed in existing drafts.
Analysis and descriptions of whether and how these functions are supported by the YANG models are provided in more detail in Section 5.

Functions	Description	Related Existing YANG Models
Obtaining Access Point Info	Getting the necessary access points info	[TE-Topo]
Obtaining Topology	Getting the topology info	[TE-Topo], [WDM-Topo] [ODU-Topo]
Tunnel Operations	Tunnel Setup, Deletion Modification and Info Retrieval	[TE-Tunnel]

Service Request	Requesting connectivity service and retrieval the list of service request	[this I.D.]	
Virtual Network Operations	Requesting a virtual network and related control operations, (e.g.,update, deletion)	[TE-Topo], [WDM topo] [ODU-Topo]	

5. Function Analysis

5.1. Toplogy Related Functions

As shown in Section 4, the functions of obtaining access point information, obtaining topology, and imposing virtual network operations can take advantages of the same set of topology YANG models. These functions are briefly explained further in the following sub-sections.

5.1.1 Obtaining Access Point Info

For cases such as scenario 1, a client may have no interest in directly controlling network resources, but might want an automated open control interface for initiating service requests. In this case, a transport controller may provide the access point information. This information can then be used in service request sent over the open interface.

The TE Topology YANG model provided in [TE-topo] can be used to provide a list of links. If the remote node and termination point information is unknown, it is omitted from the reported information. If the client-side node and termination point information is obtained via configuration or a distributed discovery mechanism, then it can also be added into the reported information. Technology-specific details might also be needed to further express the constraints/attributes associated with the access points. Note that all of this information is usually read only.

5.1.2 Obtaining Topology

Refer to [TE-Topo] for explanations and examples on how to obtain the topology. For technology specific topology information, other models such as those provided in [WDM-Topo] and [ODU-Topo] maybe used.

5.1.3 Virtual Network Operations

There are two ways to request the creation of a virtual network. One is to define the topology explicitly using the model provided in the topology YANG drafts listed in Section 5.1.2.. The other way is to provide an estimated traffic information (a traffic matrix) and ask for a network controller of the provider network to provide a virtual network that can fulfill the demand. This second approach does not have a supporting model and need further work.

5.2. Tunnel Operations

Thecurrent [TE-Tunnel] provides a technology agnostic Traffic-Engieering (TE) device tunnel. The model included in that draft is currently being developed to make it generic for both controller and device usage. It is expected that the next version of this draft will provide such a generic TE tunnel model that can cater to the base requirementss for tunnel operations but it may need to be augmented to support controller-specific operations.

Futhermore, technology-specific augmentations of the base generic TE tunnel models are needed. For example, for Optical Channel (OCh) tunnels in WDM networks, information such as the lambda resource usage is needed. Similarly, for ODU tunnels, information such as the usage of tributary slots is needed.

5.3. Service Requests

The service model is an important model that enables automated operations between a client controller and a provider controller. The transport connectivity service model is different from the model of a tunnel since the transport connectivity service model hides technical details from a client.

A transport connectivity service model is provided below:

```
module: ietf-transport-service
   +--rw transport_service
   |  +--rw service* [service-id]
   |     +--rw service-id           uint32
   |     +--rw service-name?        string
   |     +--rw source
   |     |  +--rw node-id?   node-id
   |     |  +--rw tp-id?     tp-id
   |     +--rw destination
   |     |  +--rw node-id?   node-id
```

```
|     |  +--rw tp-id?     tp-id
|     +--rw service-type?        service-types
|     +--rw supporting-tunnel* [name]
|     |  +--rw name    string
|     +--rw bandwidth            decimal64
|     +--rw SLA?                 SLAtypes
|     +--rw intended-policies
|        +--rw schedule
|           +--rw schedules
|              +--rw schedule* [schedule-id]
|                 +--rw schedule-id          uint32
|                 +--rw start?               yang:date-and-time
|                 +--rw schedule-duration?   string
|                 +--rw repeat-interval?     string
|
+--rw service-state
   +--ro service* [service-id]
      +--ro service-id           uint32
      +--ro service-name?        string
      +--ro source
      |  +--ro node-id?   node-id
      |  +--ro tp-id?     tp-id
      +--ro destination
      |  +--ro node-id?   node-id
      |  +--ro tp-id?     tp-id
      +--ro service-type?        service-types
      +--ro supporting-tunnel* [name]
      |  +--ro name    string
      +--ro bandwidth            decimal64
      +--ro SLA?                 SLAtypes
      +--ro applied-policies
      |  +--ro schedule
      |     +--ro schedules
      |        +--ro schedule* [schedule-id]
      |           +--ro schedule-id          uint32
      |           +--ro start?               yang:date-and-time
      |           +--ro schedule-duration?   string
      |           +--ro repeat-interval?     string
      +--ro status?              state-types
```

The corresponding YANG code is provided below:

<CODE BEGINS> file " ietf-transport-service@2016-3-7.yang"

```
module ietf-transport-service {
  yang-version 1;
  namespace "urn:ietf:params:xml:ns:yang:ietf_transport_service";
  prefix tser;
```

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

import ietf-schedule {
  prefix "sch";
}

organization "TBD";
contact
  "WILL-BE-DEFINED-LATER";
description
  "this module describes a service module that is essential
  API for a client to ask for a provider network for a path
  without the need to care about underlying technologies.
  Capability to specify constraints/policies are provided as
  optional features.";

revision 2016-03-07 {
  description
    "Initial revision.";
    reference "to add the draft name";
}


typedef tp-id {  //client termination port
  type union {
    type uint32;
    type inet:ip-address; // IPv4 or IPv6 address
  }
  description
    "the client termination port of a transport device";
}

typedef node-id {  //client termination port
  type union {
    type uint32;
    type inet:ip-address; // IPv4 or IPv6 address
  }
  description
    "the node id of a transport device";
}

typedef service-types {
  type enumeration {
    enum "EPL" {
```

```
          value 0;
          description
          "EPL service";
        }
        enum "EVPL" {
          value 1;
          description
          "EVPL";
        }
        enum "EPLAN" {
          value 2;
          description
          "EPLAN";
        }
        enum "EVPLAN" {
          value 3;
          description
          "EVPLAN";
        }
      }
      description "the type of a service request";
    }

    typedef state-types{
      type enumeration {
        enum "NORMAL" {
          value 0;
          description
          "service is normal/up and running";
        }
        enum "DOWN" {
          value 1;
          description
          "service is down.";
        }
        enum "DEGRADED"{
          value 2;
          description
          "service is in degraded state.";
        }
      }
      description "the state of a service.";
    }

    typedef SLAtypes{
      type enumeration{
        enum "1+1+R"{
```

```
          value 0;
          description
          "A reroute will be provided after both the working and
          protection path fails.";
        }
        enum "1+1"{
          value 1;
          description
          "a protection path is provided.";
        }
        enum "Rerouting"{
          value 2;
          description
          "rerouting after the working path fails";
        }
        enum "unprotected"{
          value 3;
          description
          "no protection provided";
        }
      }
    }

    grouping service-basics {
      //later put all service under so that it can reused
      // in states.
      leaf service-id {
        type uint32;
        description "an unique identificaiton of a service.";
      }

      leaf service-name{
        type string;
        description "name for a service";
      }

      container source{
        leaf node-id {
          type node-id;
          description "node id";
        }
        leaf tp-id {
          type tp-id;
          description "TBD";
        }
        description "Service source information";
      }
```

```
      container destination{
        leaf node-id {
          type node-id;
          description "node id";
        }
        leaf tp-id {
          type tp-id;
          description "TBD";
        }
        description "Service destination information";
      }


      leaf service-type {
        type service-types;
        description "the type of a service request";
      }

      list supporting-tunnel{
        key "name";
        leaf name{
          type string;
          description "the name of a tunnel";
        }

        description "the list of tunnesl to support the list";
      }

      leaf bandwidth {
        type decimal64 {
          fraction-digits 2;
        }
        mandatory true;
        description "the bandwidth requested by a service.";
      }

      leaf SLA{
        type SLAtypes;
        description "the type of protection expected for this
        service";
      }
    }

    container transport_service {
      description
        "serves as a top-level container for a list of services";
```

```
      list service {
        key "service-id";
        description
          "an unique identifier of a service";

        uses service-basics;

        container intended-policies {
          container schedule {
            uses sch:schedules;
            description "to specify bandwidth scheduling
            information of this service.";
          }
          description "specify the policy associated with a
          service";
        }//end of policy
      }//end of service list
    }//service top container

    container service-state
    {
      list service {
        config false;
        key "service-id";
        description "operational state of a service";

        uses service-basics;

        container applied-policies{
          container schedule {
            uses sch:schedules;
            description "to specify bandwidth scheduling
            information of this service.";
          }
        }

        leaf status {
          type state-types;
          description "TBD";
        }
      }//end of a service state
    }//end of state
  }

  <CODE ENDS>
```

6. Security Considerations

Clearly modifying server-layer resources will have a significant impact on network infrastructure. More specifically they will provide the services and applications running across client-layers, which the server-layer is supporting. Therefore, security must be an important consideration when implementing the architecture, models and protocol mechanisms discussed in this document.

Communicating service and network information (including access point identifiers, capabilities, topologies, etc.) across external interfaces represents a security risk. Thus, mechanisms to encrypt or preserve the domain topology confidentiality should be used.

A key consideration are the external protocols (those shown as entering or leaving the orchestrator and controllers shown in Figure 2 (Scenario 2: Multi-domain network control and management)) which must be appropriately secured. This security should include authentication and authorization to control access to different functions that the orchestrator may perform to modify or create state in the server-layer, and the establishment and management of the orchestrator to controller relationship.

The orchestrator will contain significant data about the network domains, the services carried by each domain, and customer type information. Therefore, access to information held in the orchestrator must be secured. Since such access will be largely through external mechanisms, it may be pertinent to apply policy-based controls to restrict access and functions.

7. Manageability Considerations

The core objectives of this document are to assist in the deployment and operation of transport services across server-layer network infrastructure. The model-driven management/control principles, which are vendor-neutral and supported by extensible APIs, should be utilized.

The open models described in this document are based on YANG [RFC6020] and the RESTCONF [RESTCONF] messaging protocol, a REST-

like protocol running over HTTP for accessing data defined in YANG, may also be used.

8. IANA Considerations

TBD.

9. Acknowledgements

Thank Igor Bryskin for useful discussions on relevant YANG models.

10. References

10.1. Normative References

[RFC2119] S. Bradner, "Key words for use in RFCs to indicate requirements levels", RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-01, work in progress, October 2014.

10.2. Informative References

[TE-Topo] Liu X., Bryskin I., et al, "YANG Data Model for TE Topologies", draft-ietf-teas-yang-te-topo-02, October 2015.

[WDM-Topo] Lee Y., et al, "A Yang Data Model for WSON Optical Networks", draft-lee-ccamp-wson-yang-02, work in progress, July 2015.

[ODU-Topo] Zhang X., Rao B., Liu X., "A YANG Data Model for Layer 1 Network Topology", draft-zhang-ccamp-l1-topo-yang-02, December 2015.

[TE-Tunnel] Saad T., Gandhi R., Liu X., et al, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-02, October, 2015.

[RESTCONF] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", Work in Progress, draft-ietf-netconf-restconf-09, December 2015.

11. Contributors' Address

Sergio Belotti
Nokia
Sergio.belotti@nokia.com

Young Lee
Futurewei Technologies
leeyoung@huawei.com

Aihua Guo
Huawei Technologies Canada
aihuaguo@huawei.com

Authors' Addresses

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

Ruiquan Jing
China Telecom
jingrq@ctbri.com.cn

Wei Jian
China Unicom
jianwei@chinaunicom.cn

Jeong-dong Ryoo
ETRI
ryoo@etri.re.kr

Yunbin Xu
China Academy of Information and Communication Technology (CAICT)
xuyunbin@ritt.cn

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