Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2014

A. Clemm
Cisco
H. Ananthakrishnan
Juniper Networks
J. Medved
T. Tkacik
Cisco
R. Varga
Pantheon Technologies SRO
N. Bahadur
Juniper Networks

October 21, 2013

A YANG Data Model for Network Topologies
draft-clemm-netmod-yang-network-topo-01.txt

Abstract

This document defines a YANG data model for network topologies.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 24, 2014.

Copyright Notice

Table of Contents

1. Introduction

This document introduces a YANG [RFC6020] [RFC6021] data model for network topologies. The model allows an application to have a

holistic view of an entire network, all contained in a single conceptual YANG datastore.

In order to capture information that is specific to of a particular type of network topology, the basic model can be augmented and adapted. As a result, the data model is generic in nature and can be applied to many network topologies. For this reason, it is suitable for use as a general YANG data model framework to capture network topologies also beyond the types that are introduced here. Specific topology types that are covered in this document include Layer 3 Unicast IGP, IS-IS [RFC1195], and OSPF [RFC2178]. Adaptations and extensions to other types of topologies are possible, using similar model patterns to the ones that are illustrated.

There are multiple applications for such a data model. For example, a network controller can use the data model to represent the controller's view of a topology it controls and expose it to northbound applications via Netconf [RFC6241] or via a ReST Interface [I-D.bierman-netconf-restconf] [I-D.lhotka-netmod-yang-json]. Alternatively, nodes within the network can use the data model to capture their understanding of the overall network topology that they are contained in, as well as propagate this understanding and compare it with that of other nodes. The data model is generic in nature and can be applied to any type of network topology.

The data model is defined in several YANG modules:

- Module "network-topology" contains a generic network topology model. It defines a network topology at its most general level of abstraction. It models aspects such as the nodes and edges that a topology graph is composed of, as well as termination points contained in the nodes that actually terminate the edges of the graph. A network can contain multiple topologies, for example topologies at different layers and overlay topologies. The model therefore allows also to capture the relationship between topologies, as well as the dependencies between nodes and termination points across topologies.

- Module "l3-unicast-igp-topology" applies the general network topology model to Layer 3 Unicast IGP topologies. It augments the general topology with information specific to Layer 3 Unicast IGP. In doing so, it also illustrates the extension patterns associated with extending respectively augmenting the general topology model to meet the needs of a specific topology.

- Module "ospf-topology" defines a topology model for OSPF, building on and extending the Layer 3 Unicast IGP topology model. It serves as an example of how the general topology model can be refined across multiple levels.

- Module "isis-topology" defines a topology model for IS-IS, again building on and extending the Layer 3 Unicast IGP topology model.

- Module "ted", finally, is a helper module, defining information kept in the Traffic Engineering Database (TED) that is leveraged by IS-IS and OSPF topologies.

2. Definitions and Acronyms

Datastore: A conceptual store of instantiated management information, with individual data items represented by data nodes which are arranged in hierarchical manner.

Data subtree: An instantiated data node and the data nodes that are hierarchically contained within it.

HTTP: Hyper-Text Transfer Protocol

IGP: Interior Gateway Protocol

IS-IS: Intermediate System to Intermediate System protocol

LSP: Label Switched Path

NETCONF: Network Configuration Protocol

OSPF: Open Shortest Path First, a link state routing protocol

URI: Uniform Resource Identifier

ReST: Representational State Transfer, a style of stateless interface and protocol that is generally carried over HTTP

SRLG: Shared Risk Link Group

TED: Traffic Engineering Database

YANG: A data definition language for NETCONF

3. Network topology model overview

This section provides an overview of the network topology model. We start with the structure of the foundational model that represents a

generic topology. Subsequently, an overview of the specific topologies is given - Layer 3 Unicast IGP, OSPF, and IS-IS, respectively. During the course of the discussion, sected design choices are explained and the pattern that should be applied to extend the model to new types of topologies is presented.

3.1. Model structure

The network topology model is defined by the following YANG modules, whose relationship is roughly depicted in the figure below.

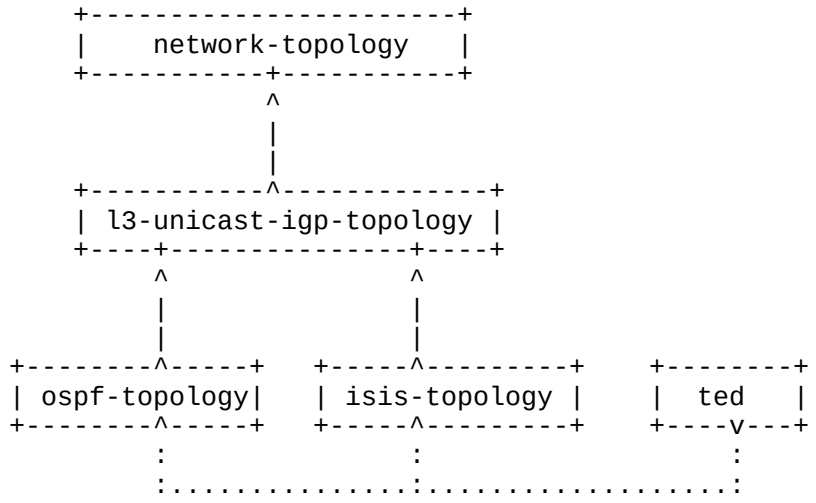

Figure 1: Overall model structure

YANG module network-topology defines the basic network topology model. YANG module l3-unicast-igp-topology builds on top of this model, augmenting network-topology with additional definitions needed to represent Layer 3 Unicast IGP topologies. This module in turn is augmented by YANG modules with additional definitions for OSPF and for IS-IS topologies, ospf-topology and isis-topology, respectively. Finally, YANG module "ted" contains a set of auxiliary definitions used by both ospf-topology and isis-topology, capturing data related to traffic engineering.

3.2. Base model: Network Topology

The structure of the network topology data model, as later defined in the YANG module "network-topology", is depicted in the following diagram. Brackets enclose list keys, "rw" means configuration data, "ro" means operational state data, and "?" designates optional nodes. The figure does not depict all definitions; it is intended to illustrate the overall structure.

```
module: network-topology
   +--rw network-topology
      +--rw topology [topology-id]
         +--rw topology-id          topology-id
         +--ro server-provided?     boolean
         +--rw topology-types
         +--rw underlay-topology [topology-ref]
         |  +--rw topology-ref    topology-ref
         +--rw node [node-id]
         |  +--rw node-id              node-id
         |  +--rw supporting-node [node-ref]
         |  |  +--rw node-ref    node-ref
         |  +--rw termination-point [tp-id]
         |     +--rw tp-id     tp-id
         |     +--ro tp-ref*   tp-ref
         +--rw link [link-id]
            +--rw link-id            link-id
            +--rw source
            |  +--rw source-node    node-ref
            |  +--rw source-tp?     tp-ref
            +--rw destination
            |  +--rw dest-node    node-ref
            |  +--rw dest-tp?     tp-ref
            +--rw supporting-link [link-ref]
               +--rw link-ref    link-ref
```

3.2.1. Main building blocks

A network can contain multiple topologies. Each topology is captured in its own list element, distinguished via a topology-id. This is captured by list "topology", contained underneath the root container for this module, "network-topology".

A topology has a certain type, such as OSPF or IS-IS. A topology can even have multiple types simultaneously. The type, or types, are captured underneath container "topology-types". This serves as container for data nodes that represent specific topology types. In this module, it serves merely as an augmentation target; topology-specific modules will later introduce new data nodes to represent new

topology types below this target, i.e. insert them below "topology-types" by ways of augmentation.

Topology types SHOULD always be represented using containers, not leafs of empty type. This allows to represent hierarchies of topology subtypes within the instance information. For example, an instance of an OSPF topology (which, at the same time, is a layer 3 unicast IGP topology) would contain underneath "topology-types" another container "l3-unicast-igp-topology", which in turn would contain a container "ospf-topology".

A topology can in turn be part of a hierarchy of topologies, building on top of other topologies. Any such topologies are captured in list "underlay-topology".

Furthermore, a topology contains nodes and links, each captured in their own list.

A node has a node-id. This distinguishes the node from other nodes in the list. In addition, a node has a list of termination points, used to terminate links. An examples of a termination point might be a physical or logical port or, more generally, an interface. Also, a node can in turn map onto other nodes in an underlay topology. This is captured in list "supporting-node".

A link is identified by a link-id, uniquely identifying the link within the topology. Links are point-to-point and unidirectional. Accordingly, a link contains a source and a destination. Both source and destination reference a corresponding node, as well as a termination point on that node. Analogous to a node, a link can in turn map onto other links an underlay topology. This is captured in list "supporting-link".

3.2.2. Discussion and selected design decisions

Rather than maintaining lists in separate containers, the model is kept relatively flat in terms of its containment structure. This way, path specifiers used to refer to specific nodes, be it in management operations or in specifications of constraints, can remain relatively compact. Of course, this means there is no separate structure in instance information that separates elements of different lists from one another. Such structure is semantically not required, although it might enhance human readability in some cases.

In an effort to minimize assumptions of what a topology might actually represent, mappings between topologies, nodes, links, and termination points are kept strictly generic. For example, no assumptions are made whether a termination point actually refers to

an interface, or whether a node refers to a specific "system" or device; the model at this generic level makes no provisions for that. Any greater specifics about mappings between upper and lower layers can be captured in augmenting modules. For example, if a termination point maps to an interface, an augmenting module can augment the termination point with a leaf that references the corresponding interface [I-D.ietf-netmod-interfaces-cfg]. If a node maps to a particular device or network element, an augmenting module can augment node with a leaf that references the network element.

The model makes extensive use of groupings, instead of simply defining data nodes "in-line". This allows to more easily include the corresponding data nodes in notifications, which then do not need to respecify each data node that is to be included. The tradeoff for this is that it makes the specification of constraints more complex, because constraints involving data nodes outside the grouping need to be specified in conjunction with a "uses" statement where the grouping is applied. This also means that constraints and XPath-statmeents need to specified in such a way that the navigate "down" first and select entire sets of nodes, as opposed to being able to simply specify them against individual data nodes.

The topology model includes links that are point-to-point and unidirectional. It does not directly support multipoint and bidirectional links. While this may appear as a limitation, it does keep the model simple, generic, and allows it to very easily be subjected applications that make use of graph algorithms. Birectional conections can be represented through pairs of unidirectional links. By introducing hierarchies of nodes, with nodes at one level mapping onto a set of other nodes at another level, and the introducing new links for nodes at that level, topologies with connections representing non-point-to-point communication patterns can be represented.

Links are terminated by a single termination point, not sets of termination points. Connections involving multihoming or link aggregation schemes need to be represented using multiple point-to-point links, then defining a link at a higher layer that is supported by those individual links.

In a hierarchy of topologies, there are nodes mapping to nodes, links mapping to links, and termination points mapping to termination points. Some of this information is redundant. Specifically, with the link-to-links mapping known, and the termination points of each link known, maintaining separate termination point mapping information is not needed but can be derived via transitive closure. The model does provide for the option to include this information explicitly, but does not allow for it to be configured to avoid the

potential to introduce (and having to validate) corresponding integrity issues.

A topology's topology types are represented using a container which contains a data node for each of its topology types. A topology can encompass several types of topology simultaneously, hence a container is used instead of a case construct, with each topology type in turn represented by a dedicated presence container itself. The reason for not simply using an empty leaf, or even simpler, do away even with the topology container and just use a leaf-list of topology-type instead, is to be able to represent "class hierarchies" of topology types, with one topology type refining the other. Topology-type specific containers are to be defined in the topology-specific modules, augmenting the topology-types container.

3.2.3. Open issues and items for further discussion

YANG requires data needs to be designated as either configuration or operational data, but not both, yet it is important to have all topology information, including vertical cross-topology dependencies, captured in one coherent model. In most cases topology information is discovered about a network; the topology is considered a property of the network that is reflected in the model. That said, it is conceivable that certain types of topology need to also be configurable by an application.

There are several alternatives in which this can be addressed. The alternative chosen in this draft does not restrict topology information as read-only, but includes a flag that indicates for each topology whether it should be considered as read-only or configurable by applications.

An alternative would be to designate topology list elements as read only. The read-only topology list includes each topology; it is the complete reference. In parallel a second topology list is introduced. This list serves the purpose of being able to configure topologies which are then mirrored in the read-only list. The configurable topology list adheres to the same structure and uses the same groupings as its read-only counterpart. As most data is defined in those groupings, the amount of additional definitions required will be limited. A configurable topology will thus be represented twice: once in the read-only list of all topologies, a second time in a configuration sandbox.

3.3. Extension of the model with specific topologies

3.3.1. Layer 3 Unicast - IGP

In order to represent a general Layer 3 Unicast IGP topology, the basic network topology model needs to be extended. The corresponding extensions are introduced in a separate YANG module "l3-unicast-igp-topology". The structure of those extensions is depicted in the following diagram. Brackets enclose list keys, "rw" means configuration, "ro" operational state data, "?" designates optional nodes, "*" designates nodes that can have multiple instances. Parantheses enclose choice and case nodes. Data nodes from the network-topology module are omitted (indicated by "....."), as long as not required to indicate containment structure. Notifications are not depicted.

```
 module: network-topology
    +--rw network-topology
       +--rw topology [topology-id]
          +.....
          +--rw topology-types
          |  +--rw l3t:l3-unicast-igp-topology?
          .....
          +--rw node [node-id]
          |  .....
          |  +--rw termination-point [tp-id]
          |  |  .....
          |  |  +--rw l3t:igp-termination-point-attributes
          |  |     +--rw (termination-point-type)?
          |  |        +--:(ip)
          |  |        |  +--rw l3t:ip-address*      inet:ip-address
          |  |        +--:(unnumbered)
          |  |           +--rw l3t:unnumbered-id?   uint32
          |  +--rw l3t:igp-node-attributes
          |     +--rw l3t:name?        inet:domain-name
          |     +--rw l3t:flag*        flag-type
          |     +--rw l3t:router-id*   inet:ip-address
          |     +--rw l3t:prefix [prefix]
          |        +--rw l3t:prefix    inet:ip-prefix
          |        +--rw l3t:metric?   uint32
          |        +--rw l3t:flag*     flag-type
          +--rw link [link-id]
          |  .....
          |  +--rw l3t:igp-link-attributes
          |     +--rw l3t:name?     string
          |     +--rw l3t:flag*     flag-type
          |     +--rw l3t:metric?   uint32
          +--rw l3t:igp-topology-attributes
             +--rw l3t:name?   string
             +--rw l3t:flag*   flag-type
```

The module augments the original network-topology module as follows:

- o A new topology type is introduced, l3-unicast-igp-topology-type. This is represented by a container object, which is inserted under the "topology-types" container of the network topology module.

- o Additional topology attributes are introduced, defined in a grouping, which augments the "topology" list of the network topology module. The attributes include an IGP name, as well as a set of flags (represented through a leaf-list). Each type of flag is represented by a separate identity. This allows to introduce additional flags in augmenting modules that are associated with specific IGP topologies, without needing to revise this module.

- o Additional data objects for nodes are introduced by augmenting the "node" list of the network topology module. New objects include again a set of flags, as well as a list of prefixes. Each prefix in turn includes an ip prefix, a metric, and a prefix-specific set of flags.

- o Links are augmented as well with a set of parameters, allowing to associate a link with an IGP name, another set of flags, and a link metric.

In addition, the module defines a set of notifications to alert clients of any events concerning links, nodes, prefixes, and termination points. Each notification includes an indication of the type of event, the topology from which it originated, and the affected node, or link, or prefix, or termination point. In addition, as a convenience to applications, additional data of the affected node, or link, or termination point (respectively) is included. While this makes notifications larger in volume than they would need to be, it avoids the need for subsequent retrieval of context information, which also might have changed in the meantime.

3.3.2. OSPF Topology

OSPF is the next type of topology represented in the model. OSPF represents a particular type of Layer 3 Unicast IGP. Accordingly, this time the Layer 3 Unicast IGP topology model needs to be extended. The corresponding extensions are introduced in a separate YANG module "ospf-topology", whose structure is depicted in the following diagram. For the most part, this module augments "l3 -unicast-igp-topology". Like before, brackets enclose list keys, "rw" means configuration, "ro" operational state data, "?" designates optional nodes, "*" designates nodes that can have multiple instances. Parantheses enclose choice and case nodes. Data nodes from the network-topology module are omitted (indicated by "....."),

as long as not required to indicate containment structure.
Notifications are not depicted.

```
module: network-topology
   +--rw network-topology
      +--rw topology [topology-id]
         .....
         +--rw topology-types
         |  +--rw l3t:l3-unicast-igp-topology?
         |     +--rw ospf:ospf?
         .....
         +--rw node [node-id]
         |  .....
         |  +--rw l3t:igp-node-attributes
         |     .....
         |     +--rw l3t:prefix [prefix]
         |     |  +.....
         |     |  +--rw ospf:ospf-prefix-attributes
         |     |     +--rw ospf:forwarding-address?   inet:ipv4-address
         |     +--rw ospf:ospf-node-attributes
         |     |  +--rw (router-type)?
         |     |  |  +--:(abr)
         |     |  |  |  +--rw ospf:abr?                empty
         |     |  |  +--:(asbr)
         |     |  |  |  +--rw ospf:asbr?               empty
         |     |  |  +--:(internal)
         |     |  |  |  +--rw ospf:internal?           empty
         |     |  |  +--:(pseudonode)
         |     |  |     +--rw ospf:pseudonode?         empty
         |     |  +--rw ospf:dr-interface-id?     uint32
         |     |  +--rw ospf:multi-topology-id*   uint8
         |     |  +--rw ospf:capabilities?        bits
         |     |  +--rw ospf:ted
         |     |     +--rw ospf:te-router-id-ipv4?    inet:ipv4-address
         |     |     +--rw ospf:te-router-id-ipv6?    inet:ipv6-address
         |     |     +--rw ospf:ipv4-local-address [ipv4-prefix]
         |     |     |  +--rw ospf:ipv4-prefix    inet:ipv4-prefix
         |     |     +--rw ospf:ipv6-local-address [ipv6-prefix]
         |     |     |  +--rw ospf:ipv6-prefix      inet:ipv6-prefix
         |     |     |  +--rw ospf:prefix-option?   uint8
         |     |     +--rw ospf:pcc-capabilities?     pcc-capabilities
         |     .....
         +--rw link [link-id]
         |  .....
         |  +--rw l3t:igp-link-attributes
         |     .....
         |     +--rw ospf:ospf-link-attributes
         |     |  +--rw ospf:multi-topology-id?   uint8
```

```
            |     |  +--rw ospf:ted
            |     |     +--rw ospf:color?                      uint32
            |     |     +--rw ospf:max-link-bandwidth?         decimal64
            |     |     +--rw ospf:max-resv-link-bandwidth?    decimal64
            |     |     +--rw ospf:unreserved-bandwidth [priority]
            |     |     |  +--rw ospf:priority     uint8
            |     |     |  +--rw ospf:bandwidth?   decimal64
            |     |     +--rw ospf:te-default-metric?          uint32
            |     |     +--rw ospf:srlg
            |     |        +--rw ospf:interface-switching-capabilities [switching
-capability]
            |     |        |  +--rw ospf:switching-capability               ted:s
witching-capabilities
            |     |        |  +--rw ospf:encoding?                          uint8
            |     |        |  +--rw ospf:max-lsp-bandwidth [priority]
            |     |        |  |  +--rw ospf:priority     uint8
            |     |        |  |  +--rw ospf:bandwidth?   decimal64
            |     |        |  +--rw ospf:packet-switch-capable
            |     |        |  |  +--rw ospf:minimum-lsp-bandwidth?   decimal64
            |     |        |  |  +--rw ospf:interface-mtu?           uint16
            |     |        |  +--rw ospf:time-division-multiplex-capable
            |     |        |     +--rw ospf:minimum-lsp-bandwidth?   decimal64
            |     |        |     +--rw ospf:indication?              uint16
            |     |        +--rw ospf:srlg-values [srlg-value]
            |     |        |  +--rw ospf:srlg-value    uint32
            |     |        +--rw ospf:link-protection-type?              uint16
            |     .....
            +--rw l3t:igp-topology-attributes
               .....
               +--rw ospf:ospf-topology-attributes
               |  +--rw ospf:area-id?   area-id
               .....
```

The module augments "l3-unicast-igp-topology" as follows:

- A new topology type for an OSPF topology is introduced. This is represented by a container object, which is inserted under the "l3-unicast-igp-topology" container of the l3-unicast-igp-topology module. This way, an ospf topology represents both a l3-unicast-igp topology and an ospf topology.

- Additional topology attributes are defined in a new grouping which augments igp-topology-attributes of the l3-unicast-igp-topology module. The attributes include an OSPF area-id identifying the OSPF area.

- Additional data objects for nodes are introduced by augmenting the igp-node-attributes of the l3-unicast-igp-topology module. New objects include router-type, de-interface-id for pseudonodes, list

of multi-topology-ids, ospf node capabilities and traffic engineering attributes.

- Links are augmented with a multi-topology-id and traffic engineering link attributes.

- Prefixes are augmented with OSPF specific forwarding address.

In addition, the module extends IGP node, link and prefix notifications with OSPF attributes.

3.3.3. IS-IS Topology

IS-IS is another type of Layer 3 Unicast IGP. Like OSPF topology, IS-IS topology is defined in a separate module, "isis-topology", which augments "l3-unicast-igp-topology". The structure is depicted in the following diagram. Like before, brackets enclose list keys, "rw" means configuration, "ro" operational state data, "?" designates optional nodes, "*" designates nodes that can have multiple instances. Parantheses enclose choice and case nodes. Data nodes from the network-topology module are omitted (indicated by "....."), as long as not required to indicate containment structure. Notifications are not depicted.

```
module: network-topology
   +--rw network-topology
      +--rw topology [topology-id]
         .....
         |  +--rw l3t:l3-unicast-igp-topology?
         |     .....
         |     +--rw isis:isis?
         .....
         +--rw node [node-id]
         .....
         |  +--rw l3t:igp-node-attributes
         |     .....
         |     +--rw isis:isis-node-attributes
         |        +--rw isis:iso
         |        |  +--rw isis:iso-system-id?       iso-system-id
         |        |  +--rw isis:iso-pseudonode-id?   iso-pseudonode-id
         |        +--rw isis:net*                    iso-net-id
         |        +--rw isis:multi-topology-id*      uint8
         |        +--rw (router-type)?
         |        |  +--:(level-2)
         |        |  |  +--rw isis:level-2?              empty
         |        |  +--:(level-1)
         |        |  |  +--rw isis:level-1?              empty
         |        |  +--:(level-1-2)
```

```
            |        |     +--rw isis:level-1-2?          empty
            |        +--rw isis:ted
            |           +--rw isis:te-router-id-ipv4?    inet:ipv4-address
            |           +--rw isis:te-router-id-ipv6?    inet:ipv6-address
            |           +--rw isis:ipv4-local-address [ipv4-prefix]
            |           |  +--rw isis:ipv4-prefix    inet:ipv4-prefix
            |           +--rw isis:ipv6-local-address [ipv6-prefix]
            |           |  +--rw isis:ipv6-prefix      inet:ipv6-prefix
            |           |  +--rw isis:prefix-option?   uint8
            |           +--rw isis:pcc-capabilities?     pcc-capabilities
            +--rw link [link-id]
            |  .....
            |  +--rw l3t:igp-link-attributes
            |     .....
            |     +--rw isis:isis-link-attributes
            |        +--rw isis:multi-topology-id?   uint8
            |        +--rw isis:ted
            |           +--rw isis:color?                      uint32
            |           +--rw isis:max-link-bandwidth?         decimal64
            |           +--rw isis:max-resv-link-bandwidth?    decimal64
            |           +--rw isis:unreserved-bandwidth [priority]
            |           |  +--rw isis:priority     uint8
            |           |  +--rw isis:bandwidth?   decimal64
            |           +--rw isis:te-default-metric?          uint32
            |           +--rw isis:srlg
            |              +--rw isis:interface-switching-capabilities [switching
-capability]
            |              |  +--rw isis:switching-capability               ted:s
witching-capabilities
            |              |  +--rw isis:encoding?                          uint8
            |              |  +--rw isis:max-lsp-bandwidth [priority]
            |              |  |  +--rw isis:priority     uint8
            |              |  |  +--rw isis:bandwidth?   decimal64
            |              |  +--rw isis:packet-switch-capable
            |              |  |  +--rw isis:minimum-lsp-bandwidth?   decimal64
            |              |  |  +--rw isis:interface-mtu?           uint16
            |              |  +--rw isis:time-division-multiplex-capable
            |              |     +--rw isis:minimum-lsp-bandwidth?   decimal64
            |              |     +--rw isis:indication?              uint16
            |              +--rw isis:srlg-values [srlg-value]
            |              |  +--rw isis:srlg-value    uint32
            |              +--rw isis:link-protection-type?               uint16
            +--rw l3t:igp-topology-attributes
               ......
               +--rw isis:isis-topogloy-attributes
                  +--rw isis:net?   iso-net-id
```

The module augments the l3-unicast-igp-topology as follows:

- o A new topology type is introduced, "isis-topology-type". This is represented by a container object, which is inserted under the "l3-unicast-igp-topology" container of the l3-unicast-igp-topology module. This way, an isis topology represents both a l3-unicast-igp-topology and an isis topology.

- o Additional topology attributes are introduced in a new grouping which augments "igp-topology-attributes" of the l3-unicast-igp-topology module. The attributes include an ISIS NET-id identifying the area.

- o Additional data objects for nodes are introduced by augmenting "igp-node-attributes" of the l3-unicast-igp-topology module. New objects include router-type, iso-system-id to identify the router, a list of multi-topology-id, a list of NET ids, and traffic engineering attributes.

- o Links are augmented with multi-topology-id and traffic engineering link attributes.

In addition, the module augments IGP nodes and links with ISIS attributes.

3.3.4. TED - Traffic Engineering Data

Traffic Engineering Data is required both by OSPF and IS-IS, which are defined in separate modules. Information shared by both is defined in another module, "ted". This module defines a set of groupings with auxiliary information required and shared by those other modules. This module details traffic-engineering node and link attributes:

- o TED node attributes include te-router-id for IPv4 and IPv6, local IPv4 and IPv6 addresses and path computation client capabilities. The path computation client capabilities in turn include a bit vector for various path computation capabilities.

- o TED link attributes comprise link color, max-link-bandwidth, max-resv-link-bandwidth, unreserved bandwidth and re-metric. They also include SRLG attributes which contains interface switching capabilities, a list of SRLG values, and a link protection type. The interface switching capabilities in turn contain a list element for each switching capability, defining encoding, max-lsp-bandwidth, and interface switching specific attributes.

4. Network Topology YANG module

<CODE BEGINS>

```
   file "network-topology@2013-10-21.yang"
   module network-topology  {
       yang-version 1;
       namespace "urn:TBD:params:xml:ns:yang:network-topology";
       // replace with IANA namespace when assigned
       prefix "nt";

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

       organization "TBD";

       contact "WILL-BE-DEFINED-LATER";

       description
           "This module defines a model for the topology of a network.
           Key design decisions are as follows:
           A topology consists of a set of nodes and links.
           Links are point-to-point and unidirectional.
           Bidirectional connections need to be represented through
           two separate links.
           Multipoint connections, broadcast domains etc can be represented
           through a hierarchy of nodes, then connecting nodes at
           upper layers of the hierarchy.";

       revision 2013-10-21 {
           description
               "Initial revision.";
       }

       typedef topology-id {
           type inet:uri;
           description
               "An identifier for a topology.";
       }

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

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

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

       typedef tp-ref {
           type leafref {
               path "/network-topology/topology/node/termination-point/tp-id";
           }
           description
               "A type for an absolute reference to a termination point.
               (This type should not be used for relative references.
               In such a case, a relative path should be used instead.)";
       }
       typedef topology-ref {
           type leafref {
               path "/network-topology/topology/topology-id";
           }
           description
               "A type for an absolute reference a topology instance.";
       }

       typedef node-ref {
           type leafref {
               path "/network-topology/topology/node/node-id";
           }
           description
```

```
            "A type for an absolute reference to a node instance.
            (This type should not be used for relative references.
            In such a case, a relative path should be used instead.)";
        }

        typedef link-ref {
            type leafref {
                path "/network-topology/topology/link/link-id";
            }
            description
                "A type for an absolute reference a link instance.
                (This type should not be used for relative references.
                In such a case, a relative path should be used instead.)";
        }

        grouping tp-attributes {
            description
                "The data objects needed to define a termination point.
                (This only includes a single leaf at this point, used
                to identify the termination point.)
                Provided in a grouping so that in addition to the datastore,
                the data can also be included in notifications.";
            leaf tp-id {
                type tp-id;
            }
            leaf-list tp-ref {
                type tp-ref;
                config false;
                description
                    "The leaf list identifies any termination points that the
                    termination point is dependent on, or maps onto.
                    Those termination points will themselves be contained
                    in a supporting node.
                    This dependency information can be inferred from
                    the dependencies between links.  For this reason,
                    this item is not separately configurable.  Hence no
                    corresponding constraint needs to be articulated.
                    The corresponding information is simply provided by the
                    implementing system.";
            }
        }

        grouping node-attributes {
            description
                "The data objects needed to define a node.
                The objects are provided in a grouping so that in addition to
                the datastore, the data can also be included in notifications
                as needed.";
```

```
            leaf node-id {
                type node-id;
                description
                    "The identifier of a node in the topology.
                    A node is specific to a topology to which it belongs.";
            }
            list supporting-node {
                description
                    "This list defines vertical layering information for nodes.
                    It allows to capture for any given node, which node (or nodes)
                    in the corresponding underlay topology it maps onto.
                    A node can map to zero, one, or more nodes below it;
                    accordingly there can be zero, one, or more elements in the li
st.
                    If there are specific layering requirements, for example
                    specific to a particular type of topology that only allows
                    for certain layering relationships, the choice
                    below can be augmented with additional cases.
                    A list has been chosen rather than a leaf-list in order
                    to provide room for augmentations, e.g. for
                    statistics or priorization information associated with
                    supporting nodes.";
                key "node-ref";
                leaf node-ref {
                    type node-ref;
                }
            }
        }

        grouping link-attributes {
            // This is a grouping, not defined inline with the link definition its
elf,
            // so it can be included in a notification, if needed
            leaf link-id {
                type link-id;
                description
                    "The identifier of a link in the topology.
                    A link is specific to a topology to which it belongs.";
            }
            container source {
                leaf source-node {
                    mandatory true;
                    type node-ref;
                    description
                        "Source node identifier, must be in same topology.";
                }
                leaf source-tp {
                    type tp-ref;
                    description
                        "Termination point within source node that terminates the
link.";
```

```
            }
        }
        container destination {
            leaf dest-node {
                mandatory true;
                type node-ref;
                description
                    "Destination node identifier, must be in same topology.";
            }
            leaf dest-tp {
                type tp-ref;
                description
                    "Termination point within destination node that terminates
 the link.";
            }
        }
        list supporting-link {
            key "link-ref";
            leaf link-ref {
                type link-ref;
            }
        }
    }


    container network-topology {
        list topology {
            description "
                This is the model of an abstract topology.
                A topology contains nodes and links.
                Each topology MUST be identified by
                unique topology-id for reason that a network could contain man
y
                topologies.
            ";
            key "topology-id";
            leaf topology-id {
                type topology-id;
                description "
                    It is presumed that a datastore will contain many topologi
es. To
                    distinguish between topologies it is vital to have UNIQUE
                    topology identifiers.
                ";
            }
            leaf server-provided {
                type boolean;
                config false;
                description "
                    Indicates whether the topology is configurable by clients,
                    or whether it is provided by the server.  This leaf is
```

```
                        populated by the server implementing the model.
                        It is set to false for topologies that are created by a cl
ient;
                        it is set to true otherwise.  If it is set to true, any
                        attempt to edit the topology MUST be rejected.
                    ";
                }
                container topology-types {
                    description
                        "This container is used to identify the type, or types
                        (as a topology can support several types simultaneously),
                        of the topology.
                        Topology types are the subject of several integrity constr
aints
                        that an implementing server can validate in order to
                        maintain integrity of the datastore.
                        Topology types are indicated through separate data nodes;
                        the set of topology types is expected to increase over tim
e.
                        To add support for a new topology, an augmenting module
                        needs to augment this container with a new empty optional
                        container to indicate the new topology type.
                        The use of a container allows to indicate a subcategorizat
ion
                        of topology types.
                        The container SHALL NOT be augmented with any data nodes
                        that serve a purpose other than identifying a particular
                        topology type.
                    ";
                }
                list underlay-topology {
                    key "topology-ref";
                    leaf topology-ref {
                        type topology-ref;
                    }
                    // a list, not a leaf-list, to allow for potential augmentatio
n
                    // with properties specific to the underlay topology,
                    // such as statistics, preferences, or cost.
                    description
                        "Identifies the topology, or topologies, that this topolog
y
                        is dependent on.";
                }

                list node {
                    description "The list of network nodes defined for the topolog
y.";
                    key "node-id";
                    uses node-attributes;
                    must "boolean(../underlay-topology[*]/node[./supporting-nodes/
node-ref])";
                        // This constraint is meant to ensure that a referenced no
de is in fact
                        // a node in an underlay topology.
                    list termination-point {
                        description
```

```
                            "A termination point can terminate a link.
                            Depending on the type of topology, a termination point
 could,
                            for example, refer to a port or an interface.";
                        key "tp-id";
                        uses tp-attributes;
                    }
                }

                list link {
                    description "
                        A Network Link connects a by Local (Source) node and
                        a Remote (Destination) Network Nodes via a set of the
                        nodes' termination points.
                        As it is possible to have several links between the same
                        source and destination nodes, and as a link could potentia
lly
                        be re-homed between termination points, to ensure that we
                        would always know to distinguish between links, every link
                        is identified by a dedicated link identifier.
                        Note that a link models a point-to-point link, not a multi
point
                        link.
                        Layering dependencies on links in underlay topologies are
                        not represented as the layering information of nodes and o
f
                        termination points is sufficient.
                    ";
                    key "link-id";
                    uses link-attributes;
                    must "boolean(../underlay-topology/link[./supporting-link]";
                        // Constraint: any supporting link must be part of an unde
rlay topology
                    must "boolean(../node[./source/source-node])";
                        // Constraint: A link must have as source a node of the sa
me topology
                    must "boolean(../node[./destination/dest-node])";
                        // Constraint: A link must have as source a destination of
 the same topology
                    must "boolean(../node/termination-point[./source/source-tp])";
                        // Constraint: The source termination point must be contai
ned in the source node
                    must "boolean(../node/termination-point[./destination/dest-tp]
)";
                        // Constraint: The destination termination point must be c
ontained
                        // in the destination node
                }
            }
        }
    }
```

<CODE ENDS>

5. Layer 3 Unicast IGP Topology YANG Module

<CODE BEGINS>

```
   file "l3-unicast-igp-topology@2013-10-21.yang"
   module l3-unicast-igp-topology {
       yang-version 1;
       namespace "urn:TBD:params:xml:ns:yang:nt:l3-unicast-igp-topology";
       // replace with IANA namespace when assigned
       prefix "l3t";
       import network-topology {
           prefix "nt";
       }

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

       organization "TBD";
       contact "TBD";

       revision "2013-10-21" {
           description "Initial revision";
           reference "TBD";
       }

       typedef igp-event-type {
           description "IGP Event type for notifications";
               type enumeration {
                   enum "add" {
                       value 0;
                           description "An IGP node or link or prefix or terminat
ion-point has been added";
                   }
                   enum "remove" {
                       value 1;
                       description "An IGP node or link or prefix or termination-
point has been removed";
                   }
                   enum "update" {
                       value 2;
                       description "An IGP node or link or prefix or termination-
point has been updated";
                   }
               }
           } // igp-event-type

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

       typedef flag-type {
```

```
            type identityref {
                base "flag-identity";
            }
        }

        grouping igp-prefix-attributes {
            leaf prefix {
                type inet:ip-prefix;
            }
            leaf metric {
                type uint32;
            }
            leaf-list flag {
                type flag-type;
            }
        }

        grouping l3-unicast-igp-topology-type {
            container l3-unicast-igp-topology {
                presence "indicates L3 Unicast IGP Topology";
            }
        }

        grouping igp-topology-attributes {
            container igp-topology-attributes {
                leaf name {
                    description "Name of the topology";
                        type string;
                }
                leaf-list flag {
                    description "Topology flags";
                    type flag-type;
                }
            }
        }

        grouping igp-node-attributes {
            container igp-node-attributes {
                leaf name {
                    description "Node name";
                    type inet:domain-name;
                }
                leaf-list flag {
                    description "Node operational flags";
                    type flag-type;
                }
                leaf-list router-id {
                    description "Router-id for the node";
```

```
                type inet:ip-address;
            }
            list prefix {
                key "prefix";
                uses igp-prefix-attributes;
            }
        }
    }

    grouping igp-link-attributes {
        container igp-link-attributes {
            leaf name {
                description "Link Name";
                type string;
            }
            leaf-list flag {
                description "Link flags";
                type flag-type;
            }
            leaf metric {
                description "Link Metric";
                type uint32 {
                    range "0..16777215"  {
                        description "
                        ";
                        // OSPF/ISIS supports max 3 byte metric.
                        // Ideally we would like this restriction to be
                        // defined in the derived models, however,
                        // we are not allowed to augment a "must" statement.
                    }
                }
            }
        }
    } // grouping igp-link-attributes

    grouping igp-termination-point-attributes {
        container igp-termination-point-attributes {
            choice termination-point-type {
                case ip {
                   leaf-list ip-address {
                    description "IPv4 or IPv6 address";
                    type inet:ip-address;
                    }
                }
                case unnumbered {
                    leaf unnumbered-id {
                    description "Unnumbered interface identifier";
                    type uint32;
```

```
                    }
                }
            }
        }
    } // grouping igp-termination-point-attributes


    augment "/nt:network-topology/nt:topology/nt:topology-types" {
        uses l3-unicast-igp-topology-type;
    }

    augment "/nt:network-topology/nt:topology" {
        when "nt:topology-types/l3-unicast-igp-topology";
        uses igp-topology-attributes;
    }

    augment "/nt:network-topology/nt:topology/nt:node" {
        when "../nt:topology-types/l3-unicast-igp-topology";
        uses igp-node-attributes;
    }
    augment "/nt:network-topology/nt:topology/nt:link" {
        when "../nt:topology-types/l3-unicast-igp-topology";
        uses igp-link-attributes;
    }
    augment "/nt:network-topology/nt:topology/nt:node/nt:termination-point" {
        when "../../nt:topology-types/l3-unicast-igp-topology";
        uses igp-termination-point-attributes;
    }

    notification igp-node-event {
        leaf igp-event-type {
            type igp-event-type;
        }
        leaf topology-ref {
            type nt:topology-ref;
        }
        uses l3-unicast-igp-topology-type;
        uses nt:node-attributes;
        uses igp-node-attributes;
    }

    notification igp-link-event {
        leaf igp-event-type {
            type igp-event-type;
        }
        leaf topology-ref {
            type nt:topology-ref;
        }
```

```
            uses l3-unicast-igp-topology-type;
            uses nt:link-attributes;
            uses igp-link-attributes;
        }

        notification igp-prefix-event {
            leaf igp-event-type {
                type igp-event-type;
            }
            leaf topology-ref {
                type nt:topology-ref;
            }
            leaf node-ref {
                type nt:node-ref;
            }
            uses l3-unicast-igp-topology-type;
            container prefix {
                uses igp-prefix-attributes;
            }
        }

        notification termination-point-event {
            leaf igp-event-type {
                type igp-event-type;
            }
            leaf topology-ref {
                type nt:topology-ref;
            }
            leaf node-ref {
                type nt:node-ref;
            }
            uses l3-unicast-igp-topology-type;
            uses nt:tp-attributes;
            uses igp-termination-point-attributes;
        }
    }

    <CODE ENDS>
```

6. OSPF Topology YANG Module

```
    <CODE BEGINS>
    file "ospf-topology@2013-10-21.yang"
    module ospf-topology {
        yang-version 1;
        namespace "urn:TBD:params:xml:ns:yang:ospf-topology";
        // replace with IANA namespace when assigned
```

```
       prefix "ospf";

       import network-topology {
           prefix "nt";
       }

       import l3-unicast-igp-topology {
           prefix "l3t";
       }
       import ietf-inet-types {
           prefix "inet";
       }
       import ted {
           prefix "ted";
       }

       organization "TBD";
       contact "TBD";
       description "OSPF Topology model";

       revision "2013-10-21" {
           description "Initial revision";
           reference "TBD";
       }

       typedef area-id {
           description "OSPF Area ID";
           type uint32;
       }

       grouping ospf-topology-type {
           container ospf {
               presence "indiates OSPF Topology";
           }
       }

       augment "/nt:network-topology/nt:topology/nt:topology-types/l3t:l3-unicast
-igp-topology" {
           uses ospf-topology-type;
       }

       augment "/nt:network-topology/nt:topology/l3t:igp-topology-attributes" {
           when "../nt:topology-types/l3t:l3-unicast-igp-topology/ospf";
           container ospf-topology-attributes {
               leaf area-id {
                   type area-id;
               }
           }
       }
```

```
       augment "/nt:network-topology/nt:topology/nt:node/l3t:igp-node-attributes"
 {
           when "../../nt:topology-types/l3t:l3-unicast-igp-topology/ospf";
           uses ospf-node-attributes;
       }

       augment "/nt:network-topology/nt:topology/nt:link/l3t:igp-link-attributes"
 {
           when "../../nt:topology-types/l3t:l3-unicast-igp-topology/ospf";
           uses ospf-link-attributes;
       }

       augment "/nt:network-topology/nt:topology/nt:node/l3t:igp-node-attributes/
l3t:prefix" {
           when "../../../nt:topology-types/l3t:l3-unicast-igp-topology/ospf";
           uses ospf-prefix-attributes;
       }

       grouping ospf-node-attributes {
           container ospf-node-attributes {
               choice router-type {
                   case abr {
                       leaf abr {
                           type empty;
                       }
                   }
                   case asbr {
                       leaf asbr {
                           type empty;
                       }
                   }
                   case internal {
                       leaf internal {
                           type empty;
                       }
                   }
                   case pseudonode {
                       leaf pseudonode {
                           type empty;
                       }
                   }
               }
               leaf dr-interface-id {
                   when "../router-type/pseudonode";
                   description "For pseudonodes, DR interface-id";
                   default "0";
                   type uint32;
               }
               leaf-list multi-topology-id {
                   description "List of Multi-Topology Identifier up-to 128 (0-12
7). RFC 4915";
                   max-elements "128";
```

```
                    type uint8 {
                        range "0..127";
                    }
                }
                leaf capabilities {
                    description "OSPF capabilities as bit vector. RFC 4970";
                    type bits {
                        bit graceful-restart-capable {
                            position 0;
                        }
                        bit graceful-restart-helper {
                            position 1;
                        }
                        bit stub-router-support {
                            position 2;
                        }
                        bit traffic-engineering-support {
                            position 3;
                        }
                        bit point-to-point-over-lan {
                            position 4;
                        }
                        bit experimental-te {
                            position 5;
                        }
                    }
                }
                container ted {
                    uses ted:ted-node-attributes;
                }
            } // ospf
        } // ospf-node-attributes

        grouping ospf-link-attributes {
            container ospf-link-attributes {
                leaf multi-topology-id {
                    type uint8 {
                        range "0..127";
                    }
                }
                container ted {
                    uses ted:ted-link-attributes;
                }
            }
        } // ospf-link-attributes

        grouping ospf-prefix-attributes {
            container ospf-prefix-attributes {
```

```
                leaf forwarding-address {
                    when "../../l3t:l3-unicast-igp-topology/l3t:ospf/l3t:router-ty
pe/l3t:asbr";
                    type inet:ipv4-address;
                }
            }
        }

        augment "/l3t:igp-node-event" {
            uses ospf-topology-type;
            uses ospf:ospf-node-attributes;
        }

        augment "/l3t:igp-link-event" {
            uses ospf-topology-type;
            uses ospf:ospf-link-attributes;
        }

        augment "/l3t:igp-prefix-event" {
            uses ospf-topology-type;
            uses ospf:ospf-prefix-attributes;
        }
    }

    <CODE ENDS>
```

7. ISIS Topology YANG Module

```
    <CODE BEGINS>
    file "isis-topology@2013-10-21.yang"
    module isis-topology {
        yang-version 1;
        namespace "urn:TBD:params:xml:ns:yang:network:isis-topology";
        // replace with IANA namespace when assigned
        prefix "isis";
        import network-topology {
            prefix nt;
        }
        import l3-unicast-igp-topology {
            prefix igp;
        }
        import ted {
            prefix ted;
        }

        organization "TBD";
        contact "TBD";
        description "ISIS Topology model";
```

```
       revision "2013-10-21" {
           description "Initial version";
       }
       typedef iso-system-id {
           description "ISO System ID. RFC 1237";
           type string {
               pattern '[0-9a-fA-F]{4}(\.[0-9a-fA-F]{4}){2}';
           }
       }

       typedef iso-pseudonode-id {
           description "ISO pseudonode id for broadcast network";
           type string {
               pattern '[0-9a-fA-F]{2}';
           }
       }
       typedef iso-net-id {
           description "ISO NET ID. RFC 1237";
           type string {
               pattern '[0-9a-fA-F]{2}((\.[0-9a-fA-F]{4}){6})';
           }
       }

       grouping isis-topology-type {
           container isis {
               presence "Indicates ISIS Topology";
           }
       }

       augment "/nt:network-topology/nt:topology/nt:topology-types/igp:l3-unicast
-igp-topology" {
           uses isis-topology-type;
       }

       augment "/nt:network-topology/nt:topology/igp:igp-topology-attributes" {
           when "../nt:topology-types/l3t:l3-unicast-igp-topology/isis";
           container isis-topology-attributes {
               leaf net {
                   type iso-net-id;
               }
           }
       }

       augment "/nt:network-topology/nt:topology/nt:node/igp:igp-node-attributes"
 {
           when "../../nt:topology-types/l3t:l3-unicast-igp-topology/isis";
           uses isis-node-attributes;
       }

       augment "/nt:network-topology/nt:topology/nt:link/igp:igp-link-attributes"
 {
```

```
            when "../../nt:topology-types/l3t:l3-unicast-igp-topology/isis";
            uses isis-link-attributes;
        }

        grouping isis-node-attributes {
            container isis-node-attributes {
                container iso {
                    leaf iso-system-id {
                        type iso-system-id;
                    }
                    leaf iso-pseudonode-id {
                        default "0";
                        type iso-pseudonode-id;
                    }
                }
                leaf-list net {
                    max-elements 3;
                    type iso-net-id;
                }
                leaf-list multi-topology-id {
                    description "List of Multi Topology Identifier upto 128 (0-127
). RFC 4915";
                    max-elements "128";
                    type uint8 {
                        range "0..127";
                    }
                }
                choice router-type {
                    case level-2 {
                        leaf level-2 {
                            type empty;
                        }
                    }
                    case level-1 {
                        leaf level-1 {
                            type empty;
                        }
                    }
                    case level-1-2 {
                        leaf level-1-2 {
                            type empty;
                        }
                    }
                }
                container ted {
                    uses ted:ted-node-attributes;
                }
            }
        }
```

```
       grouping isis-link-attributes {
           container isis-link-attributes {
               leaf multi-topology-id {
                   type uint8 {
                       range "0..127";
                   }
               }
               container ted {
                   uses ted:ted-link-attributes;
               }
           }
       }

       augment "/igp:igp-node-event" {
           uses isis-topology-type;
           uses isis-node-attributes;
       }

       augment "/igp:igp-link-event" {
           uses isis-topology-type;
           uses isis-link-attributes;
       }
   } // Module isis-topology

   <CODE ENDS>
```

8. TED YANG Module

```
   <CODE BEGINS>
   file "ted@2013-10-21.yang"
   module ted {
     yang-version 1;
     namespace "urn:TBD:params:xml:ns:yang:network:ted";
     // replace with IANA namespace when assigned
     prefix ted;

     import ietf-inet-types {
       prefix inet;
     }

     organization "TBD";
     contact
       "TBD";
     description
       "Helper module to hold TED attributes for OSPF/ISIS";

     revision 2013-10-21 {
```

```
       description
         "Initial revision";
     }

     typedef switching-capabilities {
       description
         "Switching Capabilities of an interface.";
       reference
         "RFC 5307: IS-IS Extensions in Support of Generalized
          Multi-Protocol Label Switching (GMPLS)";
       type enumeration {
         enum "PSC-1" {
           description
             "Packet-Switch Capable-1 (PSC-1)";
           value 1;
         }
         enum "PSC-2" {
           description
             "Packet-Switch Capable-2 (PSC-2)";
           value 2;
         }
         enum "PSC-3" {
           description
             "Packet-Switch Capable-3 (PSC-3)";
           value 3;
         }
         enum "PSC-4" {
           description
             "Packet-Switch Capable-4 (PSC-4)";
           value 4;
         }
         enum "L2SC" {
           description
             "Layer-2 Switch Capable (L2SC)";
           value 51;
         }
         enum "TDM" {
           description
             "Time-Division-Multiplex Capable (TDM)";
           value 100;
         }
         enum "LSC" {
           description
             "Lambda-Switch Capable (LSC)";
           value 150;
         }
         enum "FSC" {
           description
```

```
              "Fiber-Switch Capable (FSC)";
            value 200;
          }
        }
      }

      typedef pcc-capabilities {
        description
          "Path Computation Capabilities.";
        reference
          "RFC 5088, draft-ietf-pce-disco-protoc-isis-07.txt
           OSPF/ISIS Protocol Extensions for Path Computation Element (PCE) Discov
ery.";
        type bits {
          bit path-computation-with-gmpls-link-constraints {
            position 0;
          }
          bit bidirectional-path-computation {
            position 1;
          }
          bit diverse-path-computation {
            position 2;
          }
          bit load-balanced-path-computation {
            position 3;
          }
          bit synchronized-path-computation {
            position 4;
          }
          bit support-for-multiple-objective-functions {
            position 5;
          }
          bit support-for-additive-path-constraints {
            position 6;
          }
          bit support-for-request-prioritization {
            position 7;
          }
          bit support-for-multiple-requests-per-message {
            position 8;
          }
        }
      }

      grouping ted-node-attributes {
        description
          "Identifier to uniquely identify a node in TED";
        reference "RFC 5305, RFC 6119: IPv6 Traffic Engineering in IS-IS/OSPF";
        leaf te-router-id-ipv4 {
```

```
        description
          "Globally unique IPv4 Traffic Engineering Router ID.";
        type inet:ipv4-address;
      }
      leaf te-router-id-ipv6 {
        description
          "Globally unique IPv6 Traffic Engineering Router ID";
        type inet:ipv6-address;
      }
      list ipv4-local-address {
        description
          "List of IPv4 Local Address(OSPF). RFC 5786";
        key "ipv4-prefix";
        leaf ipv4-prefix {
          description
            "Local IPv4 address for the node";
          type inet:ipv4-prefix;
        }
      }
      list ipv6-local-address {
        description
          "List of IPv6 Local Address.";
        reference
          "RFC 5786: Advertising a Router's Local Addresses
           in OSPF Traffic Engineering (TE) Extensions";
        key "ipv6-prefix";
        leaf ipv6-prefix {
          description
            "Local IPv6 address for the node";
          type inet:ipv6-prefix;
        }
        leaf prefix-option {
          description
            "IPv6 prefix option.";
          type uint8;
        }
      }
      leaf pcc-capabilities {
        description
          "OSPF/ISIS PCC capabilities";
        type pcc-capabilities;
      }
    }

    grouping ted-link-attributes {
      description
        "TED Attributes associated with the link.";
      reference "RFC 3630, RFC 3784: IS-IS / OSPF Traffic Engineering (TE)";
```

```
       leaf color {
         description
           "Administrative group or color of the link";
         type uint32;
       }
       leaf max-link-bandwidth {
         description
           "Maximum bandwidth that can be see on this link in this direction. Uni
ts in bytes per second";
         type decimal64 {
           fraction-digits 2;
         }
       }
       leaf max-resv-link-bandwidth {
         description
           "Maximum amount of bandwidth that can be reserved in this direction in
 this link. Units in bytes per second";
         type decimal64 {
           fraction-digits 2;
         }
       }
       list unreserved-bandwidth {
         description
           "Unreserved bandwidth for 0-7 priority levels. Units in bytes per seco
nd";
         max-elements "8";
         key "priority";
         leaf priority {
           type uint8 {
             range "0..7";
           }
         }
         leaf bandwidth {
           description
             "Unreserved bandwidth for this level";
           type decimal64 {
             fraction-digits 2;
           }
         }
       }
       leaf te-default-metric {
         description
           "Traffic Engineering Metric";
         type uint32;
       }
       container srlg {
         description
           "Shared Risk Link Group Attributes";
         uses srlg-attributes;
       }
     }
```

```
     grouping srlg-attributes {
       description
         "Shared Risk Link Group Attributes";
       reference
         "RFC 5307, RFC 4203: ISIS / OSPF Extensions in Support of
          Generalized Multi-Protocol Label Switching (GMPLS)";
       list interface-switching-capabilities {
         description
           "List of interface capabilities for this interface";
         key "switching-capability";
         leaf switching-capability {
           description
             "Switching Capability for this interface";
           type ted:switching-capabilities;
         }
         leaf encoding {
           description
             "Encoding supported by this interface";
           type uint8;
         }
         list max-lsp-bandwidth {
           description
             "Maximum LSP Bandwidth at priorities 0-7";
           max-elements "8";
           key "priority";
           leaf priority {
             type uint8 {
               range "0..7";
             }
           }
           leaf bandwidth {
             description
               "Max LSP Bandwidth for this level";
             type decimal64 {
               fraction-digits 2;
             }
           }
         }
         container packet-switch-capable {
           when "../switching-capability = PSC-1 or ../switching-capability = PSC
-2 or ../switching-capability = PSC-3 or ../switching-capability = PSC-4";
           description
             "Interface has packet-switching capabilities";
           leaf minimum-lsp-bandwidth {
             description
               "Minimum LSP Bandwidth. Units in bytes per second";
             type decimal64 {
               fraction-digits 2;
             }
```

```
            }
            leaf interface-mtu {
              description
                "Interface MTU";
              type uint16;
            }
          }
          container time-division-multiplex-capable {
            when "../switching-capability = TDM";
            description
              "Interface has time-division multiplex capabilities";
            leaf minimum-lsp-bandwidth {
              description
                "Minimum LSP Bandwidth. Units in bytes per second";
              type decimal64 {
                fraction-digits 2;
              }
            }
            leaf indication {
              description
                "Indication whether the interface supports Standard or Arbitrary S
ONET/SDH";
              type uint16;
            }
          }
        }
        list srlg-values {
          description
            "List of Shared Risk Link Group this interface belongs to.";
          key "srlg-value";
          leaf srlg-value {
            description
              "Shared Risk Link Group value";
            type uint32;
          }
        }
        leaf link-protection-type {
          description
            "Link Protection Type desired for this link";
          type uint16;
        }
      }
    }

    <CODE ENDS>
```

9. Security Considerations

The transport protocol used for sending the topology data MUST support authentication and SHOULD support encryption. The data-model by itself does not create any security implications.

10. Contributors

The model presented in this paper was contributed to by more people than can be listed on the author list. Additional contributors include:

- Ken Gray, Juniper Networks
- Tom Nadeau, Juniper Networks
- Aleksandr Zhdankin, Cisco

11. Acknowledgements

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from Ladislav Lhotka, Andy Bierman, Carlos Pignataro, and Juergen Schoenwaelder.

12. References

12.1. Normative References

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, December 1990.

[RFC2178] Moy, J., "OSPF Version 2", RFC 2178, July 1997.

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

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

12.2. Informative References

[I-D.bierman-netconf-restconf] Bierman, A., Bjorklund, M., Watsen, K., and R. Fernando, "RESTCONF Protocol", draft-bierman-netconf-restconf-02 (work in progress), October 2013.

[I-D.ietf-netmod-interfaces-cfg]
Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-interfaces-cfg-12 (work in progress), July 2013.

[I-D.lhotka-netmod-yang-json]
Lhotka, L., "Modeling JSON Text with YANG", draft-lhotka-netmod-yang-json-02 (work in progress), September 2013.

Authors' Addresses

Alexander Clemm
Cisco

EMail: alex@cisco.com

Hariharan Ananthakrishnan
Juniper Networks

EMail: hanantha@juniper.net

Jan Medved
Cisco

EMail: jmedved@cisco.com

Tony Tkacik
Cisco

EMail: ttkacik@cisco.com

Robert Varga
Pantheon Technologies SRO

EMail: robert.varga@pantheon.sk

Nitin Bahadur
Juniper Networks

EMail: nitinb@juniper.net