Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 15, 2020

I. Bryskin
Individual
V. Beeram
T. Saad
Juniper Networks
X. Liu
Volta Networks
Y. Lee
Sung Kyun Kwan University
A. Guo
Futurewei

January 12, 2020

Basic YANG Model for Steering Client Services To Server Tunnels
draft-bryskin-teas-service-tunnel-steering-model-04

Abstract

This document describes a YANG data model for managing pools of transport tunnels and steering client services on them.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Client layer services/signals are normally mapped onto carrying them across the network transport tunnels via client/server layer adaptation relationships. Such relationships are usually modeled as multi-layer topologies, whereas tunnels set up in underlay (server) topologies support links in respective overlay (client) topologies. In this respect having a link in a client topology means that the client layer traffic could be forwarded between link termination points (LTPs) terminating the link on opposite sides by the supporting tunnel(s) provisioned in the server layer topology.

This said there are numerous use cases in which describing the client service to server tunnel bindings via the topology formalism is impractical. Below are some examples of such use cases:

o Mapping client services onto tunnels within the same network layer, for example, mapping L3 VPNs or MPLS-SR services onto IP MPLS tunnels;

o Mapping client services onto tunnels provisioned in the highest layer topology supported by the network. For example, mapping

L2VPNs or E(V)PL services onto IP MPLS tunnels provisioned in an IP network;

o Mapping client services to tunnels provisioned in separate network layers at the network's access points. Consider, for example, an OTN/ODUk network that is used to carry client signals of, say, 20 different types (e.g. Ethernet, SDH, FKON, etc.) entering and exiting the network over client facing interfaces. Although it is possible to describe such a network as a 21-layer TE topology with the OTN/ODUk topology serving each of the 20 client layer topologies [I-D.ietf-teas-yang-te-topo], such a description would be verbose, cumbersome, difficult to expand to accommodate additional client signals and unnecessary, because the client layer topologies would have zero switching flexibility inside the network (i.e. contain only unrelated links connecting access points across respective layer networks), and all what is required to know from the point of view of a management application is what ODUk tunnels are established or required, which client signals the tunnels could carry and at which network border nodes and how the client signals could be bound (i.e. adopted) to the tunnels.

It is worth noting that such non-topological client-service-to-server-tunnel mapping almost always happens on network border nodes. However, there are also important use cases where such a mapping is required in the middle of the network. One such use case is controlling on IP/MPLS FRR PLRs which LSPs are mapped onto which backup tunnels.

It is important to bear in mind that service2tunnel mappings could be very complex: large number of instances of services of the same or different types (possibly governed by different models) could be mapped on the same set of tunnels, with the latter being set in different network layers and of either TE or non-TE nature, P2P or P2MP or MP2MP type. Furthermore, the mappings could be hierarchical: tunnels carrying services could be clients of other tunnels.

Despite of the differences of transport tunnels and of services they carry the srvice2tunnel mappings could be modeled in a simple uniform way. Access to a data store of such mappings could be beneficial to network management applications. It would be possible, for example, to discover which services depend on which tunnels, which services will be affected if a given tunnel goes out of service, how many more services could be placed onto a given TE tunnel without the latter violating its TE commitments (such as bandwidth and delay). It would be also possible to demand in a single request moving numerous (ranges of) service instances from one set of tunnels to another.

This document defines a YANG data model for facilitating said xervice2tunnel mappings.

The YANG model in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342].

1.1. Terminology

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

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

- o augment
- o data model
- o data node

1.2. Tree Diagrams

A simplified graphical representation of the data model is presented in this document, by using the tree format defined in [RFC8340].

1.3. Prefixes in Data Node Names

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

Prefix	YANG module	Reference
inet	ietf-inet-types	[RFC6991]
te-types	ietf-te-types	[I-D.ietf-teas-yang-te]

Table 1: Prefixes and Corresponding YANG Modules

2. **Explicit vs. Implicit Service2tunnel Mapping.** Steering Services to Transport Tunnel Pools

There are use cases in which client services require hard separation of the transport carrying them from the transport carrying other services. However, environment in which the services may share the same transport tunnels is far more common. For this reason the model defined in this document suggests replacing (or at least augmenting) the explicit service2tunnel mapping configuration (in which the tunnels are referred to by their IDs/names) with an implicit mapping. Specifically, the model introduces the notion of tunnel pool. A tunnel pool could be referred to by its network unique color and requires a service2tunnel mapping configuration to specify the tunnel pool color(s) instead of tunnel IDs/names. The model governs tunnel pool data store independently from the services steered on the tunnels. It is assumed (although not required) that the tunnels - constituents/components of a tunnel pool - are of the same type, provisioned using a common template. Importantly they could be dynamically added to/removed from the pool without necessitating service2tunnel mapping re-configuration. Such a service to tunnel pool steering approach has the following advantages:

o Scalability and efficiency: pool component bandwidth utilization could be monitored, tunnels could be added to/removed from the pool if/when detected that current component bandwidth utilization has crossed certain thresholds. This allows for a very efficient network resource utilization and obviates the network management application from a very difficult task of service to tunnel mapping planning;

o Automation and elasticity: pool component attributes could be modified - bandwidth auto-adjusted, protection added, delay constrained, etc.. The tunnels could be completely or partially replaced with tunnels of different types (e.g. TE vs. non-TE, P2P vs. P2MP, etc.) or even provisioned in different network layers (OTN/ODUk tunnels replacing IP TE tunnels). Importantly, all such modifications do not require service2tunell mapping re-configurations as long as the modified or new tunnels remain within the same tunnel pool(s);

o Transparency: new service sites supported by additional PEs could be added without service2tunnel mapping re-configuration.

3. **The purpose of the model**

The model is targeted to facilitate for network management applications, such as service orchestrators, the control of pools of transport tunnels and steering onto them client services

independently of network technology/layer specifics of both the services and the tunnels. The model could be applied to/implemented on physical devices, such as IP routers, as well as on abstract topology nodes. Furthermore, the model could be supported by a network (domain) controller, such as ACTN PNC, to act as a proxy server on behalf of any network element/node (physical or abstract) under its control.

4. Model Design

The data store described/governed by the model is comprised of a single top level list - TunnelPools. A TunnelPool, list element, is a container describing a set of transport tunnels (presumably with similar characteristics) identified by a network unique ID (color). A given TunnelPool could be generic to the entire network or specific to a particular netwrok slice or network abstract topology. Furthermore, a TunnelPool may have no tunnels (i.e. may have empty Tunnels list). Service steered onto such a TunnelPool will be carried by best effort forwarding technique and flexibility available in the slice/topology the TunnelPool is assigned to or generally in the network

The TunnelPool container has the following fields:

o Color [uint32 list key];

o Slice/Abstract topology ID (if zero, the TunnelPool is generic to the network).

o Tunnels list;

o Services list.

The Tunnels list describes the pool constituents - active transport tunnels. The list members - Tunnel containers - include the following information:

o tunnel type [e.g. P2P-TE, P2MP-TE, SR-TE, SR P2P, LDP P2P, LDP MP2MP, GRE, PBB, etc]

o tunnel type specific tunnel ID [provided that a data store of the tunnel type, e.g. TE tunnels, is supported, the tunnelID allows for the management application to look up the tunnel in question to obtain detailed information about the tunnel];

o tunnel encapsulation [e.g. MPLS label stack, Ethernet STAGs, GRE header, PBB header, etc].

The Services list describes services currently steered on the tunnel pool. The list members - Service containers - have the following attributes:

- service type [e.g. fixed/transparent, L3VPN, L2VPN, EVPN, ELINE, EPL, EVPL, L1VPN, ACTN VN, etc.];

- service type specific service ID [provided that a data store of the service type, e.g. L2VPN, is supported, the service ID allows for the management application to look up the service in question to obtain detailed information about the service];

- client ports (source/destination node LTPs over which the service enters/exits the node/network, relevant only for fixed/transparent services);

- service encapsulation [e.g. MPLS label stack, Ethernet CTAGs, etc.] - for service multiplexing/de-multiplexing on/from a statistically shared tunnel].

5. Tree Structure

```
   module: ietf-tunnel-steering
     +--rw tunnel-pools
        +--rw tunnel-pool* [color]
           +--rw color                     uint32
           +--rw description?              string
           +--rw te-topology-identifier
           |  +--rw provider-id?   te-types:te-global-id
           |  +--rw client-id?     te-types:te-global-id
           |  +--rw topology-id?   te-types:te-topology-id
           +--rw service* [service-type id]
           |  +--rw service-type     identityref
           |  +--rw id               string
           |  +--rw encapsulation
           |  |  +--rw type?    identityref
           |  |  +--rw value?   binary
           |  +--rw access-point* [node-address link-termination-point]
           |     +--rw node-address              inet:ip-address
           |     +--rw link-termination-point    string
           |     +--rw direction?                enumeration
           +--rw tunnel* [tunnel-type source destination tunnel-id]
              +--rw tunnel-type      identityref
              +--rw source           inet:ip-address
              +--rw destination      inet:ip-address
              +--rw tunnel-id        binary
              +--rw encapsulation
                 +--rw type?    identityref
                 +--rw value?   binary
```

6. YANG Modules

```
   <CODE BEGINS> file "ietf-tunnel-steering@2020-01-05.yang"
   module ietf-tunnel-steering {
     yang-version 1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tunnel-steering";

     prefix "tnl-steer";

     import ietf-inet-types {
       prefix inet;
     }

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

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

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

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

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

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

        Editor:   Xufeng Liu
                  <mailto:xufeng.liu.ietf@gmail.com>

        Editor:   Young Lee
                  <mailto:leeyoung@huawei.com> ";

     description
       "This data model is for steering client service to server
        tunnels.

        Copyright (c) 2020 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

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

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

     revision 2020-01-05 {
       description "Initial revision";
       reference "TBD";
     }

     /*
      * Typedefs
      */
```

```
     /*
      * Identities
      */
     identity service-type {
       description "Base identity for client service type.";
     }
     identity service-type-l3vpn {
       base service-type;
       description
         "L3VPN service.";
     }
     identity service-type-l2vpn {
       base service-type;
       description
         "L2VPN service.";
     }
     identity service-type-evpn {
       base service-type;
       description
         "EVPN service.";
     }
     identity service-type-eline {
       base service-type;
       description
         "ELINE service.";
     }
     identity service-type-epl {
       base service-type;
       description
         "EPL service.";
     }
     identity service-type-evpl {
       base service-type;
       description
         "EVPL service.";
     }
     identity service-type-l1vpn {
       base service-type;
       description
         "L1VPN service.";
     }
     identity service-type-actn-vn {
       base service-type;
       description
         "ACTN VN service.";
     }
     identity service-type-transparent {
       base service-type;
```

```
         description
           "Transparent LAN service.";
       }

       identity tunnel-type {
         description "Base identity for tunnel type.";
       }
       identity tunnel-type-te-p2p {
         base tunnel-type;
         description
           "TE point-to-point tunnel type.";
       }
       identity tunnel-type-te-p2mp {
         base tunnel-type;
         description
           "TE point-to-multipoint tunnel type.";
         reference "RFC4875";
       }
       identity tunnel-type-te-sr {
         base tunnel-type;
         description
           "Segment Rouging TE tunnel type.";
       }
       identity tunnel-type-sr {
         base tunnel-type;
         description
           "Segment Rouging tunnel type.";
       }
       identity tunnel-type-ldp-p2p {
         base tunnel-type;
         description
           "LDP point-to-point tunnel type.";
       }
       identity tunnel-type-ldp-mp2mp {
         base tunnel-type;
         description
           "Multicast LDP multipoint-to-multipoint tunnel type.";
       }
       identity tunnel-type-gre {
         base tunnel-type;
         description
           "GRE tunnel type.";
       }
       identity tunnel-type-pbb {
         base tunnel-type;
         description
           "PBB tunnel type.";
       }
```

```
     identity service-encapsulation-type {
       description "Base identity for tunnel encapsulation.";
     }
     identity service-encapsulation-type-mpls-label {
       base service-encapsulation-type;
       description
         "Encapsulated by MPLS label stack, as an inner lable to
          identify the customer service.";
     }
     identity service-encapsulation-type-ethernet-c-tag {
       base service-encapsulation-type;
       description
         "Encapsulated by Ethernet C-TAG, to identify the customer
          service.";
     }

     identity tunnel-encapsulation-type {
       description "Base identity for tunnel encapsulation.";
     }
     identity tunnel-encapsulation-type-mpls-label {
       base tunnel-encapsulation-type;
       description
         "Encapsulated by MPLS label stack, as an outer label to
          be pushed into the tunnel.";
     }
     identity tunnel-encapsulation-type-ethernet-s-tag {
       base tunnel-encapsulation-type;
       description
         "Encapsulated by Ethernet S-TAG.";
     }
     identity tunnel-encapsulation-type-pbb {
       base tunnel-encapsulation-type;
       description
         "Encapsulated by PBB header.";
     }
     identity tunnel-encapsulation-type-gre {
       base tunnel-encapsulation-type;
       description
         "Encapsulated by GRE header.";
     }

     /*
      * Groupings
      */

     /*
      * Configuration data and operational state data nodes
      */
```

```
     container tunnel-pools {
       description
         "A list of mappings that steer client services to transport
          tunnel pools. The tunnel pools are managed independently from
          the services steered on them.";

       list tunnel-pool {
         key "color";
         description
           "A set of transport tunnels (presumably with similar
            characteristics) identified by a network unique ID, named
            'color'.";
         leaf color {
           type uint32;
           description
             "Unique ID of a tunnel pool.";
         }
         leaf description {
           type string;
           description
             "Client provided description of the tunnel pool.";
         }
         uses te-types:te-topology-identifier;

         list service {
           key "service-type id";
           description
             "A list of client services that are steered on this tunnel
              pool.";
           leaf service-type {
             type identityref {
               base service-type;
             }
             description
               "Service type required by the client.";
           }
           leaf id {
             type string;
             description
               "Unique ID of a client service for the specified
                service type.";
           }
           container encapsulation {
             description
               "The encapsulation information used to identify the
                customer service for multiplexing over shared tunnels.";
             leaf type {
               type identityref {
```

```
               base service-encapsulation-type;
             }
             description
               "The encapsulation type used to identify the customer
                service for multiplexing over shared tunnels.";
           }
           leaf value {
             type binary;
             description
               "The encapsulation value pushed to the tunnel to
                identify this service.
                If not specified, the system decides what
                value to be used for multiplexing.";
           }
         }
         list access-point {
           key "node-address link-termination-point";
           description
             "A list of client ports (Link Termination Points) for the
              service to enter or exist.";
           leaf node-address {
             type inet:ip-address;
             description
               "Node over which the service enters or exists.";
           }
           leaf link-termination-point {
             type string;
             description
               "Client port (Link Termination Point) over which the
                service enters or exits.";
           }
           leaf direction {
             type enumeration {
               enum "in" {
                 description "The service enters to the network.";
               }
               enum "out" {
                 description "The service exists from the network.";
               }
               enum "in-out" {
                 description
                   "The service enters to and exists from the
                    network.";
               }
             }
             description
               "Whether the service enters to or exits from the
                network.";
```

```
             }
           }
         }
         list tunnel {
           key "tunnel-type source destination tunnel-id";
           description
             "A list of tunnels in the tunnel pool.";

           leaf tunnel-type {
             type identityref {
               base tunnel-type;
             }
             description
               "Tunnel type based on constructing technologies and
                multipoint types, including P2P-TE, P2MP-TE, SR-TE,
                SR P2P, LDP P2P, LDP MP2MP, GRE, PBB, etc";
           }
           leaf source {
             type inet:ip-address;
             description
               "For a p2p or p2mp tunnel, this is the source address;
                for a mp2mp tunnel, this is the root address.";
             reference "RFC3209, RFC4875, RFC6388, RFC7582.";
           }
           leaf destination {
             type inet:ip-address;
             description
               "For a p2p tunnel, this is the tunnel endpoint address
                extracted from SESSION object;
                for a p2mp tunnel, this identifies the destination
                group, or p2mp-id;
                for a mp2mp tunnel identified by root and opaque-value,
                this value is set to '0.0.0.0'.";
             reference "RFC3209, RFC4875, RFC6388, RFC7582.";
           }
           leaf tunnel-id {
             type binary;
             description
               "For a p2p or p2mp tunnel, this is the tunnel identifier
                used in the SESSION that remains constant over the life
                of the tunnel;
                for a mp2mp tunnel, this is the opaque-value in the
                FEC element.";
             reference "RFC3209, RFC4875, RFC6388, RFC7582.";
           }
           container encapsulation {
             description
               "The encapsulation information used by the tunnel.";
```

```
             leaf type {
               type identityref {
                 base service-encapsulation-type;
               }
               description
                 "The encapsulation type used by the tunnel.";
             }
             leaf value {
               type binary;
               description
                 "The encapsulation value pushed to the tunnel data to
                  identify the traffic in this tunnel.
                  If not specified, the system decides what
                  value to be used.";
             }
           }
         }
       }
     }
   }
   <CODE ENDS>
```

7. IANA Considerations

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

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

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

This document registers the following YANG modules in the YANG Module Names registry [RFC7950]:

```
   --------------------------------------------------------------------
   name:         ietf-tunnel-steering
   namespace:    urn:ietf:params:xml:ns:yang:ietf-tunnel-steering
   prefix:       tnl-steer
   reference:    RFC XXXX
   --------------------------------------------------------------------
```

8. Security Considerations

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

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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

/tunnel-pools/tunnel-pool
   This subtree specifies a list of tunnel pools. Modifying the configurations cause interruption to related services and tunnels.

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

/tunnel-pools/tunnel-pool
   Unauthorized access to this subtree can disclose the information of related services and tunnels.

9. References

9.1. Normative References

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

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

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

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

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

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

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

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

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

[I-D.ietf-teas-yang-te-topo]
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", draft-ietf-teas-yang-te-topo-22 (work in progress), June 2019.

[I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-22 (work in progress), November 2019.

9.2. Informative References

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

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

Authors' Addresses

Igor Bryskin
Individual

EMail: i_bryskin@yahoo.com

Vishnu Pavan Beeram
Juniper Networks

EMail: vbeeram@juniper.net

Tarek Saad
Juniper Networks

EMail: tsaad@juniper.net

Xufeng Liu
Volta Networks

EMail: xufeng.liu.ietf@gmail.com

Young Lee
Sung Kyun Kwan University

EMail: younglee.tx@gmail.com

Aihua Guo
Futurewei

EMail: aihuaguo@futurewei.com