Teas Working Group
Internet Draft

Intended status: Informational

Expires: December 21, 2018

Young Lee
Huawei

Sergio Belotti
Nokia

Dhruv Dhody
Huawei

Daniele Ceccarelli
Ericsson

Bin Yeong Yoon
ETRI

June 21, 2018

Information Model for Abstraction and Control of TE Networks (ACTN)

draft-ietf-teas-actn-info-model-10.txt

Abstract

This draft provides an information model for Abstraction and Control of Traffic Engineered Networks (ACTN).

Status of this Memo

Table of Contents

1. Introduction

This draft provides an information model for Abstraction and Control of Traffic Engineered Networks (ACTN). The information model described in this document covers the interface requirements identified in the ACTN architecture and framework document [ACTN-Frame].

The ACTN reference architecture [ACTN-Frame] identifies a three-tier control hierarchy comprising the following as depicted in Figure 1:

- Customer Network Controllers (CNCs)
- Multi-Domain Service Coordinator (MDSC)
- Provisioning Network Controllers (PNCs).

```
   +-------+              +-------+                 +-------+
   | CNC-A |              | CNC-B |                 | CNC-C |
   +-------+              +-------+                 +-------+
        \                     |                         /
         ------------         | CMI         -------------
                     \        |            /
                      +-----------------------+
                      |         MDSC          |
                      +-----------------------+
                     /        |            \
         ------------         | MPI         -------------
        /                     |                         \
   +-------+              +-------+                 +-------+
   |  PNC  |              |  PNC  |                 |  PNC  |
   +-------+              +-------+                 +-------+
```

Figure 1: A Three-tier ACTN control hierarchy

The two interfaces with respect to the MDSC, one north of the MDSC and the other south of the MDSC are referred to as CMI (CNC-MDSC Interface) and MPI (MDSC-PNC Interface), respectively. This document models these two interfaces and derivative interfaces thereof (e.g., MDSC to MDSC in a hierarchy of MDSCs) as a single common interface.

1.1. Terminology

The terms "Virtual Network (VN)" and "Virtual Network Service (VNS)" are defined in [ACTN-Frame] and the other key terms such as "abstraction", "abstract topology", "Path", "VN node", and "VN link" are defined in [RFC7926].

2. ACTN Common Interfaces Information Model

This section provides an ACTN common interface information model to describe primitives, objects, their properties (represented as attributes), their relationships, and the resources for the service applications needed in the ACTN context.

The standard interface is described between a client controller and a server controller. A client-server relationship is recursive between a CNC and an MDSC and between an MDSC and a PNC. In the CMI,

the client is a CNC while the server is an MDSC. In the MPI, the client is an MDSC and the server is a PNC. There may also be MDSC-MDSC interface(s) that need to be supported. This may arise in a hierarchy of MDSCs in which workloads may need to be partitioned to multiple MDSCs.

Basic primitives (messages) are required between the CNC-MDSC and MDSC-PNC controllers. These primitives can then be used to support different ACTN network control functions like network topology request/query, VN service request, path computation and connection control, VN service policy negotiation, enforcement, routing options, etc.

There are two different types of primitives depending on the type of interface:

- Virtual Network primitives at CMI
- Traffic Engineering primitives at MPI

As well described in [ACTN-Frame], at the CMI level, there is no need for detailed TE information since the basic functionality is to translate customer service information into virtual network service operation.

At the MPI level, MDSC has the main scope for multi-domain coordination and creation of a single e2e abstracted network view which is strictly related to TE information.

As for topology, this document employs two types of topology:

- The first type is referred to as virtual network topology which is associated with a VN. Virtual network topology is a customized topology for view and control by the customer. See Section 3.1 for details.

- The second type is referred to as TE topology which is associated with provider network operation on which we can apply policy to obtain the required level of abstraction to represent the underlying physical network topology.

3. Virtual Network primitives

This section provides a list of main VN primitives related to virtual network which are necessary to satisfy ACTN requirements specified in [ACTN-REQ]

The following VN Action primitives are supported:

- VN Instantiate

- VN Modify

- VN Delete

- VN Update

- VN Path Compute

- VN Query

VN Action is an object describing the main VN primitives.

VN Action can assume one of the mentioned above primitives values.

```
<VN Action> ::= <VN Instantiate> |

                <VN Modify> |

                <VN Delete> |

                <VN Update> |

                <VN Path Compute> |

                <VN Query>
```

All these actions will solely happen at CMI level between Customer Network Controller (CNC) and Multi Domain Service Coordinator (MDSC).

3.1. VN Instantiate

VN Instantiate refers to an action from customers/applications to request the creation of VNs. VN Instantiate is for CNC-to-MDSC communication. Depending on the agreement between client and provider, VN instantiate can imply different VN operations. There are two types of VN instantiation:

VN type 1: VN is viewed as a set of edge-to-edge links (VN members).

VN type 2: VN is viewed as a VN-topology comprising virtual nodes and virtual links.

Please see [ACTN-Frame] for full details regarding the types of VN.

3.2. VN Modify

VN Modify refers to an action issued from customers/applications to modify an existing VN (i.e., an instantiated VN). VN Modify is for CNC-to-MDSC communication.

VN Modify, depending of the type of VN instantiated, can be a modification of the characteristics of VN members (edge-to-edge links) in case of VN type 1, or a modification of an existing virtual topology (e.g., adding/deleting virtual nodes/links) in case of VN type 2.

3.3. VN Delete

VN Delete refers to an action issued from customers/applications to delete an existing VN. VN Delete is for CNC-to-MDSC communication.

3.4. VN Update

VN Update refers to any update to the VN that needs to be updated to the customers. VN Update is MDSC-to-CNC communication. VN Update fulfills a push model at CMI level, to make customers aware of any specific changes in the topology details related to the instantiated VN.

VN Update, depending of the type of VN instantiated, can be an update of VN members (edge-to-edge links) in case of VN type 1, or an update of virtual topology in case of VN type 2.

The connection-related information (e.g., LSPs) update association with VNs will be part of the "translation" function that happens in MDSC to map/translate VN request into TE semantics. This information will be provided in case customer optionally wants to have more detailed TE information associated with the instantiated VN.

3.5. VN Compute

VN Compute consists of Request and Reply. Request refers to an action from customers/applications to request a VN computation.

VN Compute Reply refers to the reply in response to VN Compute Request.

VN Compute Request/Reply is to be differentiated from a VN Instantiate. The purpose of VN Compute is a priori exploration to compute network resources availability and getting a possible VN view in which path details can be specified matching customer/applications constraints. This a priori exploration may not guarantee the availability of the computed network resources at the time of instantiation.

3.6. VN Query

VN Query refers to an inquiry pertaining to a VN that has already been instantiated. VN Query fulfills a pull model that permits getting a topology view.

VN Query Reply refers to the reply in response to VN Query. The topology view returned by VN Query Reply would be consistent with the topology type instantiated for any specific VN.

4. Traffic Engineering (TE) primitives

This section provides a list of the main TE primitives necessary to satisfy ACTN requirements specified in [ACTN-REQ] related to typical TE operations supported at the MPI level.

The TE action primitives defined in this section should be supported at the MPI consistently with the type of topology defined at the CMI.

The following TE action primitives are supported:

- TE Instantiate/Modify/Delete

- TE Topology Update (See Section 4.4. for the description)

- Path Compute

TE Action is an object describing the main TE primitives.

TE Action can assume one of the mentioned above primitives values.

```
<TE Action> ::= <TE Instantiate> |

                <TE Modify> |

                <TE Delete> |

                <TE Topology Update> |
```

<Path Compute> |

All these actions will solely happen at MPI level between Multi Domain Service Coordinator (MDSC) and Provisioning Network Controller (PNC).

4.1. TE Instantiate

TE Instantiate refers to an action issued from MDSC to PNC to instantiate new TE tunnels.

4.2. TE Modify

TE Modify refers to an action issued from MDSC to PNC to modify existing TE tunnels.

4.3. TE Delete

TE Delete refers to an action issued from MDSC to PNC to delete existing TE tunnels.

4.4. TE Topology Update (for TE resources)

TE Topology Update is a primitive specifically related to MPI to provide TE resource update between any domain controller towards MDSC regarding the entire content of any "domain controller" actual TE topology or an abstracted filtered view of TE topology depending on negotiated policy.

See [TE-TOPO] for detailed YANG implementation of TE topology update.

<TE Topology Update> ::= <TE-topology-list>

<TE-topology-list> ::= <TE-topology> [<TE-topology-list>]

<TE-topology> ::= [<Abstraction>] <TE-Topology-identifier> <Node-list> <Link-list>

<Node-list> ::= <Node>[<Node-list>]

<Node> ::= <Node> <TE Termination Point-list>

<TE Termination Point-list> ::= <TE Termination Point> [<TE-Termination Point-list>]

<Link-list> ::= <Link>[<Link-list>]

Where

Abstraction provides information on level of abstraction (as determined a priori).

TE-topology-identifier is an identifier that identifies a specific te-topology, e.g., te-types:te-topology-id [TE-TOPO].

Node-list is detailed information related to a specific node belonging to a te-topology, e.g., te-node-attributes [TE-TOPO].

Link-list is information related to the specific link related belonging to a te-topology, e.g., te-link-attributes [TE-TOPO].

TE Termination Point-list is detailed information associated with the termination points of te-link related to a specific node, e.g., interface-switching-capability [TE-TOPO].

4.5. Path Compute

Path Compute consists of Request and Reply. Request refers to an action from MDSC to PNC to request a path computation.

Path Compute Reply refers to the reply in response to Path Compute Request.

The context of Path Compute is described in [Path-Compute].

5. VN Objects

This section provides a list of objects associated to VN action primitives.

5.1. VN Identifier

VN Identifier is a unique identifier of the VN.

5.2. VN Service Characteristics

VN Service Characteristics describes the customer/application requirements against the VNs to be instantiated.

```
<VN Service Characteristics> ::= <VN Connectivity Type>

                                 <VN Directionality>

                                 (<VN Traffic Matrix>...)

                                 <VN Survivability>
```

Where

```
<VN Connectivity Type> ::= <P2P>|<P2MP>|<MP2MP>|<MP2P>|<Multi-
destination>
```

The Connectivity Type identifies the type of required VN Service. In addition to the classical type of services (e.g. P2P/P2MP etc.), ACTN defines the "multi-destination" service that is a new P2P service where the end points are not fixed. They can be chosen among a list of pre-configured end points or dynamically provided by the CNC.

VN Directionality indicates if a VN is unidirectional or bidirectional. This implies that each VN member that belongs to the VN has the same directionality as the VN.

```
<VN Traffic Matrix> ::= <Bandwidth>

                        [<VN Constraints>]
```

The VN Traffic Matrix represents the traffic matrix parameters for the required service connectivity. Bandwidth is a mandatory parameter and a number of optional constraints can be specified in the VN Constraints (e.g. diversity, cost). They can include objective functions and TE metrics bounds as specified in [RFC5541].

Further details on the VN constraints are specified below:

```
      <VN Constraints> ::= [<Layer Protocol>]
```

[<Diversity>]

(<Metric> | <VN Objective Function>)

Where:

Layer Protocol identifies the layer topology at which the VN service is requested. It could be for example MPLS, ODU, and OCh.

Diversity allows asking for diversity constraints for a VN Instantiate/Modify or a VN Path Compute. For example, a new VN or a path is requested in total diversity from an existing one (e.g. diversity exclusion).

<Diversity> ::= (<VN-exclusion> (<VN-id>...)) |

(<VN-Member-exclusion> (<VN-Member-id>...))

Metric can include all the Metrics (cost, delay, delay variation, latency), bandwidth utilization parameters defined and referenced by [RFC3630] and [RFC7471].

As for VN Objective Function See Section 5.4.

VN Survivability describes all attributes related to the VN recovery level and its survivability policy enforced by the customers/applications.

<VN Survivability> ::= <VN Recovery Level>

[<VN Tunnel Recovery Level>]

[<VN Survivability Policy>]

Where:

VN Recovery Level is a value representing the requested level of resiliency required against the VN. The following values are defined:

. Unprotected VN
. VN with per tunnel recovery: The recovery level is defined against the tunnels composing the VN and it is specified in the VN Tunnel Recovery Level.

<VN Tunnel Recovery Level> ::= <0:1>|<1+1>|<1:1>|<1:N>|<M:N>|

<On the fly restoration>

The VN Tunnel Recovery Level indicates the type of protection or restoration mechanism applied to the VN. It augments the recovery types defined in [RFC4427].

```
<VN Survivability Policy> ::= [<Local Reroute Allowed>]

                              [<Domain Preference>]

                              [<Push Allowed>]

                              [<Incremental Update>]
```

Where:

Local Reroute Allowed is a delegation policy to the Server to allow or not a local reroute fix upon a failure of the primary LSP.

Domain Preference is only applied on the MPI where the MDSC (client) provides a domain preference to each PNC (server), e.g., when an inter-domain link fails, then PNC can choose the alternative peering with this info.

Push Allowed is a policy that allows a server to trigger an updated VN topology upon failure without an explicit request from the client. Push action can be set as default unless otherwise specified.

Incremental Update is another policy that triggers an incremental update from the server since the last period of update. Incremental update can be set as default unless otherwise specified.

5.3. VN End-Point

VN End-Point Object describes the VN's customer end-point characteristics.

```
<VN End-Point> ::= (<Access Point Identifier>

                   [<Access Link Capability>]
```

[<Source Indicator>])...

Where:

Access Point Identifier represents a unique identifier of the client end-point. They are used by the customer to ask for the setup of a virtual network instantiation. A VN End-Point is defined against each AP in the network and is shared between customer and provider. Both the customer and the provider will map it against their own physical resources.

Access Link Capability identifies the capabilities of the access link related to the given access point. (e.g., max-bandwidth, bandwidth availability, etc.)

Source Indicator indicates if an end-point is source or not.

5.4. VN Objective Function

The VN Objective Function applies to each VN member (i.e., each E2E tunnel) of a VN.

The VN Objective Function can reuse objective functions defined in [RFC5541] section 4.

For a single path computation, the following objective functions are defined:

- o MCP is the Minimum Cost Path with respect to a specific metric (e.g. shortest path).
- o MLP is the Minimum Load Path, that means find a path composted by te-link least loaded.
- o MBP is the Maximum residual Bandwidth Path.

For a concurrent path computation, the following objective functions are defined:

- o MBC is to Minimize aggregate Bandwidth Consumption.
- o MLL is to Minimize the Load of the most loaded Link.
- o MCC is to Minimize the Cumulative Cost of a set of paths.

5.5. VN Action Status

VN Action Status is the status indicator whether the VN has been successfully instantiated, modified, or deleted in the server network or not in response to a particular VN action.

Note that this action status object can be implicitly indicated and thus not included in any of the VN primitives discussed in Section 3.

5.6. VN Topology

When a VN is seen by the customer as a topology, it is referred to as VN topology. This is associated with VN Type 2, which is composed of virtual nodes and virtual links.

<VN Topology> ::= <VN node list> <VN link list>

<VN node list> ::= <VN node> [<VN node list>]

<VN link list> :: = <VN link> [<VN link list>]

5.7. VN Member

VN Member describes details of a VN Member which is a list of a set of VN Members represented as VN_Member_List.

<VN_Member_List> ::= <VN Member> [<VN_Member_List>]

Where <VN Member> ::= <Ingress VN End-Point>

[<VN Associated LSP>]

<Egress VN End-Point>

Ingress VN End-Point is the VN End-Point information for the ingress portion of the AP. See Section 5.3 for VN End-Point details.

Egress VN End-Point is the VN End-Point information for the egress portion of the AP. See Section 5.3 for VN End-Point details.

VN Associated LSP describes the instantiated LSPs in the Provider's network for the VN Type 1. It describes the instantiated LSPs over the VN topology for VN Type 2.

5.7.1. VN Computed Path

The VN Computed Path is the list of paths obtained after the VN path computation request from a higher controller. Note that the computed path is to be distinguished from the LSP. When the computed path is signaled in the network (and thus the resource is reserved for that path), it becomes an LSP.

```
<VN Computed Path> ::= (<Path>...)
```

5.7.2. VN Service Preference

This section provides VN Service preference. VN Service is defined in Section 2.

```
<VN Service Preference> ::= [<Location Service Preference >]

                            [<Client-specific Preference >]

                            [<End-Point Dynamic Selection Preference >]
```

Where

Location Service Preference describes the End-Point Location's (e.g. Data Centers) support for certain Virtual Network Functions (VNFs) (e.g., security function, firewall capability, etc.) and is used to find the path that satisfies the VNF constraint.

Client-specific Preference describes any preference related to Virtual Network Service (VNS) that application/client can enforce via CNC towards lower level controllers. For example, CNC can enforce client-specific preferences, e.g., selection of a destination data center from the set of candidate data centers based on some criteria in the context of VM migration. MSDC/PNC should then provide the data center interconnection that supports the client-specific preference.

End-Point Dynamic Selection Preference describes if the End-Point (e.g. Data Center) can support load balancing, disaster recovery or VM migration and so can be part of the selection by MDSC following service Preference enforcement by CNC.

6. TE Objects

6.1. TE Tunnel Characteristics

Tunnel Characteristics describes the parameters needed to configure TE tunnel.

```
<TE Tunnel Characteristics> ::= [<Tunnel Type>]

                                <Tunnel Id>

                                [<Tunnel Layer>]

                                [<Tunnel end-point>]

                                [<Tunnel protection-restoration>]

                                <Tunnel Constraints>

                               [<Tunnel Optimization>]
```

Where

<Tunnel Type> ::= <P2P>|<P2MP>|<MP2MP>|<MP2P>

The Tunnel Type identifies the type of required tunnel. In this draft, only P2P model is provided.

Tunnel Id is the TE tunnel identifier

Tunnel Layer represents the layer technology of the LSPs supporting the tunnel

<Tunnel End Points> ::= <Source> <Destination>

<Tunnel protection-restoration> ::= <prot 0:1>|<prot 1+1>|<prot 1:1>|<prot 1:N>|prot <M:N>|<restoration>

Tunnel Constraints are the base tunnel configuration constraints parameters.

```
Where <Tunnel Constraints> ::= [<Topology Id>]

                               [<Bandwidth>]

                               [<Disjointness>]

                               [<SRLG>]

                               [<Priority>]

                               [<Affinities>]

                               [<Tunnel Optimization>]

                               [<Objective Function>]
```

Topology Id references the topology used to compute the tunnel path.

Bandwidth is the bandwidth used as parameter in path computation

<Disjointness> ::= <node> | <link> | <srlg>

Disjointness provides the type of resources from which the tunnel has to be disjointed

SRLG is a group of physical resources impacted by the same risk from which an E2E tunnel is required to be disjointed.

<Priority> ::= <Holding Priority> <Setup Priority>

where

Setup Priority indicates the level of priority for taking resources from another tunnel [RFC3209]

Holding Priority indicates the level of priority to hold resources avoiding preemption from another tunnel [RFC3209]

Affinities it represent structure to validate link belonging to path of the tunnel [RFC3209]

<Tunnel Optimization> ::= <Metric> | <Objective Function>

Metric can include all the Metrics (cost, delay, delay variation, latency), bandwidth utilization parameters defined and referenced by [RFC3630] and [RFC7471].

```
<Objective Function> ::= <objective function type>

<objective function type> ::= <MCP> | <MLP> | <MBP> | <MBC> | <MLL>
| <MCC>
```

See chapter 5.4 for objective function type description.

7. Mapping of VN primitives with VN Objects

This section describes the mapping of VN primitives with VN Objects based on Section 5.

```
<VN Instantiate> ::= <VN Service Characteristics>

                     <VN Member-List>

                     [<VN Service Preference>]

                     [<VN Topology>]



<VN Modify> ::= <VN identifier>

                <VN Service Characteristics>

                <VN Member-List>

                [<VN Service Preference>]

                [<VN Topology>]



<VN Delete> ::= <VN Identifier>



<VN Update> :: = <VN Identifier>
```

```
                    [<VN Member-List>]

                    [<VN Topology>]


   <VN Path Compute Request> ::= <VN Service Characteristics>

                                 <VN Member-List>

                                 [<VN Service Preference>]


   <VN Path Compute Reply> ::= <VN Computed Path>


   <VN Query> ::= <VN Identifier>


   <VN Query Reply> ::= <VN Identifier>

                        <VN Associated LSP>

                        [<TE Topology Reference>]
```

8. Mapping of TE primitives with TE Objects

This section describes the mapping of TE primitives with TE Objects based on Section 6.

```
   <TE Instantiate> ::= <TE Tunnel Characteristics>

   <TE Modify> ::=  <TE Tunnel Characteristics>


   <TE Delete> ::= <Tunnel Id>
```

<TE Topology Update> ::= <TE-topology-list>

<Path Compute Request> ::= <TE Tunnel Characteristics>

<Path Compute Reply> ::= <TE Computed Path>

<TE Tunnel Characteristics>

9. Security Considerations

The ACTN information model is not directly relevant when considering potential security issues. Rather, it defines a set of interfaces for traffic engineered networks. The underlying protocols, procedures, and implementations used to exchange the information model described in this draft will need to secure the request and control of resources with proper authentication and authorization mechanisms. In addition, the data exchanged over the ACTN interfaces discussed in this document requires verification of data integrity. Backup or redundancies should also be available to restore the affected data to its correct state.

Implementations of the ACTN framework will have distributed functional components that will exchange a concrete instantiation that adheres to this information model. Implementations should encrypt data that flows between them, especially when they are implemented at remote nodes and irrespective of whether these data flows are on external or internal network interfaces. The information model may contain customer, application and network data that for business or privacy reasons may be considered sensitive. It should be stored only in an encrypted data store.

The ACTN security discussion is further split into two specific interfaces:

- Interface between the Customer Network Controller and Multi Domain Service Coordinator (MDSC), CNC-MDSC Interface (CMI)

- Interface between the Multi Domain Service Coordinator and Provisioning Network Controller (PNC), MDSC-PNC Interface (MPI)

See the detailed discussion of the CMI and MPI in Sections 9.1 and 9.2, respectively in [ACTN-Frame].

The conclusion is that all data models and protocols used to realize the ACTN info model should have rich security features as discussed in this section. Additional security risks may still exist. Therefore, discussion and applicability of specific security functions and protocols will be better described in documents that are use case and environment specific.

10. IANA Considerations

This document has no actions for IANA.

11. References

11.1. Normative References

[ACTN-Frame] D. Ceccarelli, et al., "Framework for Abstraction and Control of Transport Networks", draft-ietf-teas-actn-framework, work in progress.

11.2. Informative References

[TE-TOPO] Liu, X. et al., "YANG Data Model for TE Topologies", draft-ietf-teas-yang-te-topo, work in progress.

[RFC3209] D. Awduche, et al, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC3630] D. Katz, K. Kompella, D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, September 2003.

[RFC4427] E. Mannie, D. Papadimitriou (Editors), "Recovery (Protection and Restoration) Terminology for Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4427, March 2006.

[RFC5541] JL. Le Roux, JP. Vasseur and Y. Lee, "Encoding of Objective Functions in the Path Computation Element Communication Protocol (PCEP)", RFC 5541, June 2009.

[RFC7471] S. Giacalone, et al, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, March 2015.

[RFC7926] A. Farrel, et al., "Problem Statement and Architecture for Information Exchange between Interconnected Traffic-Engineered Networks", RFC 7926, July 2016.

[Path-Compute] I. Busi, S. Belotti, et al., "Yang model for requesting Path Computation", draft-ietf-teas-yang-path-computation", work in progress.

13. Contributors

Contributors' Addresses

Haomian Zheng
Huawei Technologies
Email: zhenghaomian@huawei.com

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

Authors' Addresses

Young Lee (Editor)
Huawei Technologies
5340 Legacy Drive
Plano, TX 75023, USA
Phone: (469)277-5838
Email: leeyoung@huawei.com

Sergio Belotti (Editor)
Alcatel Lucent
Via Trento, 30
Vimercate, Italy
Email: sergio.belotti@alcatel-lucent.com

Dhruv Dhody
Huawei Technologies,
Divyashree Technopark, Whitefield
Bangalore, India
Email: dhruv.ietf@gmail.com

Daniele Ceccarelli
Ericsson
Torshamnsgatan,48
Stockholm, Sweden
Email: daniele.ceccarelli@ericsson.com

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