OPEN NETWORKING FOUNDATION

Functional Requirements for Transport API

Version Number - 0.11
Sept 30, 2015

ONF Document Type: Technical Recommendation
ONF Document Name: Functional Requirements for Transport API

Disclaimer

Open Networking Foundation
2275 E. Bayshore Road, Suite 103, Palo Alto, CA 94303
www.opennetworking.org

List of Figures

List of Tables

No table of figures entries found.

1 Introduction

Software defined networking (SDN) paradigm revolves around separation of forwarding/data plane and control plane, logically centralized control and application-focused programmable interfaces. In transport networks where logically-centralized control/management and control-data separation are not new concepts and the network-control function and behavior are well-understood and established, standardizing application programmer's interfaces (APIs) to the network control functions become important.

1.1 Purpose

The purpose of this document is to specify the information that is relevant to an application programmer's interface (API) to transport network-control functions and serve as a "Functional Requirements Specification" (FRS) document for the transport API work in ONF.

Since the APIs are defined at interface boundaries and are intended to mask the need to understand the internal architectures on either side of the interface boundary, the focus has been to define purpose-specific use case scenarios from an application point of view treating the API provider as a "black box". These application use cases have been used to drive the API requirements as well as to harmonize, generalize and normalize the API specifications. To facilitate the understanding of these requirements, some of the use cases are described in the appendices.

Although these requirements are based off few use cases, a key purpose is to ensure that the APIs do not limit valid application use cases not considered here.

1.2 Scope

Based on the use case contributions and early discussions in the ONF/OTWG "Transport API" project discussion group, this work is initially scoped to specifying APIs for the following transport network controller services:

- Topology Service
- Connectivity Service
- Path Computation Service
- Virtual Network Service
- Notification Service

For the purposes of this document, it is assumed that access control and policy details are conveyed via a sideways/orthogonal interface. It is understood that all API requests would be subject to filtering and scoping based on the privileges assigned to the calling entity and these would be based on business contracts as well as security and organizational roles. Application of such policy constraints and filtering to the API requests and responses is out of scope for this document. In other words, the API considerations in this document are from the perspective of the most privileged super-user.

1.3 References

ONF TR-512	Core Information Model
ONF TR-502	SDN Architecture
ONF TR-516	Framework for SDN: Scope and Requirements
ONF2015.276.xx	SDN Notifications Framework (draft)
ONF2015.320.xx	Transport API IM Concepts
ONF2015.381.xx	Transport API Examples
ONF2015.338.xx	State Information Model
ONF2015.323.xx	LTP/End-Point Directionality

1.4 Abbreviations

API	Application Programmer's Interface
IM	Information Model
OTWG	Optical Transport Working Group
OTN	Optical Transport Network
OAM	Operations, Administration and Maintenance
MPLS-TP	MPLS-Transport Profile
EMS/NMS	Element/Network Management System
ASON/GMPLS	Automatic Switched Optical Network/Generalized MultiProtocol Label Switching
SLA	Service Level Agreement
NE	Network Element
FD	Forwarding Domain
NCD	Network Control Domain
EP	End Point
LTP	Logical Termination Point
T-API/TAPI	Transport API
TED	Traffic Engineering Database

1.5 Terms and Definitions

This section defines some key terms that aid in understating the requirements. More information is provided in the appendices and it is recommended that the reader familiarize themselves with the basic concepts, constructs and use cases described in those sections.

In general, the T-API uses terminology that is familiar to the transport network management industry, but maps to constructs defined in the ONF Core Information Model in form of purpose-specific realizations. So it must be noted that these definitions are neither authoritative nor exhaustive, and the reader should refer to the realized/mapped entities defined in ONF Core Information Model document.

Also it should be noted that API IM relates to information exchanged over an interface and the entity definitions are intended to provide a logical structure for encapsulating information that is exchanged, and not intended to specify the information model patterns for implementations on either side of the interface.

API Context (Context)

The T-API *Context* is a realization of the *Network-Control-Domain* entity defined in the ONF Core Information Model and defines the scope of control and naming that a particular SDN

controller or application module has with respect to a specific instance of network abstraction and provides a context for naming of entities within the scope. This *Context* is shared between the API provider and its client and determines the makeup of the network resource database over which the API operates.

Topology (Topology)

The T-API *Topology* is a realization of the transparent topological-aspects of the *Forwarding-Domain (FD)* entity defined in the ONF Core Information Model and exposes the underlying lower-level topological network of *Nodes* and *Links* that enable the forwarding function provided by the *FD*.

Node

The T-API *Node* is a realization of the opaque forwarding-aspects of the *Forwarding-Domain (FD)* entity defined in the ONF Core Information Model and exposes the edge ports of the *FD* (*Node-Edge-Point*) and the forwarding capabilities between those edge ports.

Link

The T-API *Link* is a realization of the *Link* entity defined in the ONF Core Information Model and represents effective adjacency between two or more associated *Nodes* in a *Topoplogy*. The T-API *Link* is terminated by *Node-Edge-Points* of the associated *Nodes*. The *Node-Edge-Points* have a specific *end-point-role* with respect to the *Link*

Node-Edge-Point

The T-API *Node-Edge-Point* is a realization of the *Logical-Termination-Point* (*LTP*) entity defined in the ONF Core Information Model that represents the node-facing aspect of the edge-port functions to access the forwarding capabilities provided by the *Node*. Hence it is an encapsulation of addressing, mapping, termination, adaptation and OAM functions of one or more transport layers (including circuit and packet forms) performed at the entry and exit points of the *Node*.

Service-End-Point

The T-API *Service-End-Point* is a realization of the *Logical-Termination-Point* (*LTP*) entity defined in the ONF Core Information Model that represents the outward-facing aspect of the edge-port functions to access the forwarding capabilities provided by the *Node*. Hence it provides a limited, simplified view of interest to external clients (e.g. client addressing, capacity, resource availability, etc), that enable the clients to make a connectivity service request without the need to understand the provider network internals.

Connection-End-Point

The T-API *Connection-End-Point* is a realization of the *Logical-Termination-Point* (*LTP*) entity defined in the ONF Core Information Model and represents access to the forwarding function provided by the *Connection*. Hence it represents ingress/egress ports that terminate a *Connection*. The *Connection-End-Point* are related to the *Node-Edge-Points* as client *LTPs* (either direct containment or through intermediate *LTPs*)

Connectivity-Service (Service)

The T-API *Service* is a realization of the *Forwarding-Construct* (*FC*) entity defined in the ONF Core Information Model that represents an “intent-like” *request* for connectivity between two or more *Service-End-Points* of a *Node*. The *Service-End-Points* have a specific *end-point-role* with respect to the *Service* request. As such, *Service* is a container for connectivity request details and is distinct from the *Connection* that realizes the request. But it must be noted these entities are defined for API purposes to refer to different aspects of information and internally could map to same instance if there is one-to-one mapping.

Connection

The T-API *Connection* is a realization of the *Forwarding-Construct* (*FC*) entity defined in the ONF Core Information Model that represents an *enabled* (provisioned) potential for forwarding (including all circuit and packet forms) between two or more *Node-Edge-Points* of a *Node*. The T-API *Connection* is terminated by *Connection-End-Points* which are clients of the associated *Node-Edge-Points*. The *Connection-End-Points* have a specific *end-point-role* with respect to the *Connection.* As such, the Connection is a container for provisioned connectivity that tracks the state of the allocated resources and is distinct from the connectivity *Service* request. But it must be noted these entities are defined for API purposes to refer to different aspects of information and internally could map to same instance if there is one-to-one mapping.

Connection Route/Path (Path)

The T-API *Path* is a realization of the *FC-Route* entity defined in the ONF Core Information Model that represents the route of a *Connection* through the lower-level *Nodes* in the underlying *Topology*, described as a list of references to the lower-level *Connections*.

1.6 Conventions

This document uses the keywords "may" and "must" to qualify optional and mandatory requirements.

2 Functional Architecture

The of Transport APIs are defined in the background context of network programmability and applies SDN principles to enable cost reduction, innovation and reduced time to market of new services by exposing standard SDN Controller control plane functions API. The Transport API abstracts a common set of control plane functions, such as Network Topology, Connectivity Requests, Path Computation and Network Virtualization to a set of Service interfaces.

These APIs are defined to be applicable on the interface between a Transport SDN controller "Black Box" and its client application. The actors involved in the information exchange over this interface include transport network provider domain controllers in the role of producers and the transport network application systems in the role of the consumers. The transport network application systems could be either a business client system (which itself may include some control functions) or the network operator's upper level control, orchestration and/or operations systems. This includes privileged application systems that would expect access to internal views of the network model and states using these same set of APIs - for example, usage of topology APIs to access abstract/virtual network topologies provided to business clients as well the underlying actual network topology and entities to which the abstract /virtual entities are mapped. The T-APIs are also intended to be equally applicable between the controllers within a transport controller recursive hierarchy.

It is understood that the APIs are executed within a shared *Context* between the API provider and its client application. The negotiation and setup of the shared *Context* is outside the T-API scope, but is expected to minimally involve agreement on *Service End Points*, its naming (TRI) and its capabilities. This *Context* determines the makeup of the network resource database over which the API operates. For example, the API client could

- Request retrieval of the Service End Points in the shared context
- Request creation of a Connectivity Service between the Service End Points
- Request creation/modification of Virtual Network topology within the shared context
- Request retrieval of the (Virtual) Network Topology (either provider-assigned or client-created-VN) within the shared context

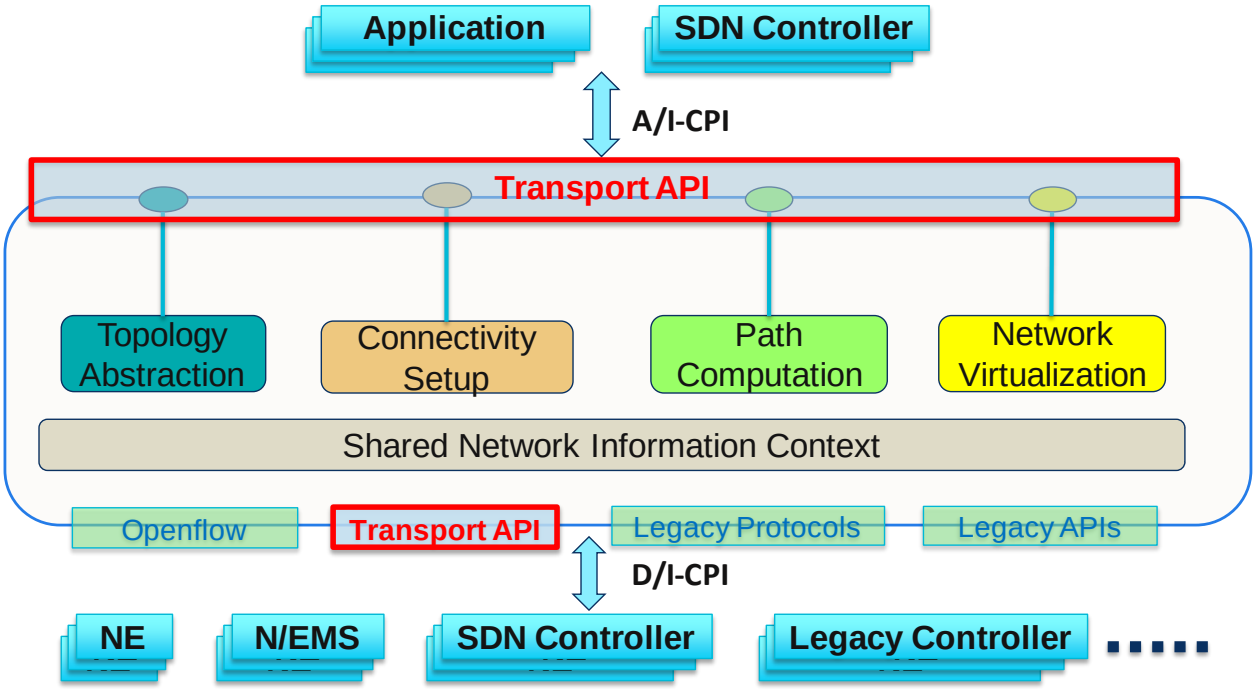

Figure 1: Transport API Functional Architecture

3 Functional Requirements

3.1 Topology Service

3.1.1 Topology Retrieval APIs

TAPI_FR_000x	**Get Topology Details**
Description	• Returns attributes of the *Topology* identified by the provided input Id. • This includes references to lower-level *Nodes* and *Links* encompassed by the *Topology*
Pre-conditions	
Inputs	• *Topology* ID or Name : String - A NULL input value is expected to return the top-most *Topology* that corresponds to the scope of the entire *Context* including any *Off-Network-Links*. • Retrieve Scope Filter: *Layer-Protocol* List : Enumeration value - If set/non-empty, the API call will return references to only those encompassed *Nodes* and *Links* that support at least one of the specified layer protocols
Outputs	• List of IDs, Names, Labels and Extensions (if any) • List of references to encompassed *Nodes* indexed by Layer • List of references to encompassed *Links* indexed by Layer
Notifications	
Error-conditions	
Post-conditions	
Sources	TMF MTNM/MTOSI 3.5 OIF-ONF Demo Topology API Spec oif-p105.11.xx (Get Topology) OIF Topology API Draft Spec oif-2014.91.xx (List Vertices)

TAPI_FR_000x	**Get Node Details**
Description	• Returns attributes of the *Node* identified by the provided input Id. • This includes references to *NodeEdgePoints* aggregated by the *Node* • This also includes attributes representing the identification/naming, states and capabilities of the *Node*.
Pre-conditions	
Inputs	• *Node* ID or Name : String - When NULL is provided, this API call should return an error condition • Retrieve Scope Filter: *Layer-Protocol* List : Enumeration value - If set/non-empty, the API call will return references to only those aggregated *NodeEdgePoints* that support at least one of the specified layer protocols
Outputs	• List of IDs, Names, Labels and Extensions (if any) • List of supported *Layer-Protocol* Names • Administrative, and Operational, • Transfer characteristics such as Cost, Timing, Integrity and Capacity • List of references to aggregated *NodeEdgePoints* indexed by Layer
Notifications	

Error-conditions	
Post-conditions	
Sources	TMF MTNM/MTOSI 3.5 OIF-ONF Demo Topology API Spec oif-p105.11.xx (Get Vertex) OIF Topology API Draft Spec oif-2014.91.xx (Get Vertex)

TAPI_FR_000x	**Get Link Details**
Description	• Returns attributes of the *Link* identified by the provided input Id. • This includes references to *NodeEdgePoints* terminating the *Link*. • This includes references to the *Nodes* associated by the *Link*. • This refers to an abstract/logical entity and could represent virtual links and/or compound links (internally aggregate lower-level serial/parallel links)
Pre-conditions	
Inputs	• *Link* ID or Name : String - When NULL is provided, this API call should return an error • Retrieve Scope Filter: *Layer-Protocol* List : Enumeration value - If set/non-empty, the API call will return references to only those terminating *NodeEdgePoints* that support at least one of the specified layer protocols
Outputs	• List of IDs, Names, Labels and Extensions (if any) • Administrative, Operational, Control and Usage States • List of supported *Layer-Protocol* Names • Transfer characteristics such as Cost, Timing, Integrity and Capacity • Risk characteristics including shared-risk • Validation characteristics • List of *Link-Ends* indexed by Layer and details of each including – Role of the terminating *NodeEdgePoint* in the context of the Link – Reference to terminating *NodeEdgePoint* – Reference to associated *Node*
Notifications	
Error-conditions	
Post-conditions	
Sources	TMF MTNM/MTOSI 3.5 OIF-ONF Demo Topology API Spec oif-p105.11.xx (Get Edge) OIF Topology API Draft Spec oif-2014.91.xx (Get Edge)

TAPI_FR_000x	**Get Node Edge Point Details**
Description	• Returns attributes of the *NodeEdgePoint* identified by the provided input Id. • This includes references to *ServiceEndPoints* mapped to this *NodeEdgePoint*.
Pre-conditions	
Inputs	• *NodeEdgePoint* ID or Name : String - When NULL is provided, this API call should return an error • Retrieve Scope Filter: *Layer-Protocol* List : Enumeration value - If set/non-empty, the API call will return only the specified *Layer-Protocol* attribute-details indexed by Layer

Outputs	• List of IDs, Names, Labels and Extensions (if any) • Physical Port Reference/Name • Administrative, Operational, Control and Usage States • List of supported *Layer-Protocols* including attribute-details indexed by Layer • List of references to any mapped *ServiceEndPoints*
Notifications	
Error-conditions	
Post-conditions	
Sources	TMF MTNM/MTOSI 3.5 OIF-ONF Demo Topology API Spec oif-p105.11.xx (Get End-point, Edge-End, Edge-End-Resource) OIF Topology API Draft Spec oif-2014.91.xx (Get End-point, Edge-End, Edge-End-Resource)

3.2 Connectivity Service

3.2.1 Connectivity Retrieval APIs

TAPI_FR_000x	**Get Connectivity Service End Point List**
Description	• Returns list of *ServiceEndPoints* that can be used or are being used in a connectivity *Service* request • This also includes the attribute details for each *ServiceEndPoint*
Pre-conditions	
Inputs	• Retrieve Scope Filter: *Layer-Protocol* List : Enumeration value - If set/non-empty, the API call will return references to only those encompassed *ServiceEndPointss* that support at least one of the specified layer protocols
Outputs	List of *ServiceEndPoints* indexed by *Layer* and details for each including: • List of IDs, Names, Labels and Extensions (if any) • Physical Port Reference/Name • Administrative, Operational, and Lifecycle States • List of supported *Layer-Protocols* including attribute-details indexed by Layer
Notifications	
Error-conditions	
Post-conditions	
Sources	TMF MTNM/MTOSI 3.5 OIF-ONF Demo Topology API Spec oif-p105.11.xx (Get End-point, Edge-End, Edge-End-Resource) OIF Topology API Draft Spec oif-2014.91.xx (Get End-point, Edge-End, Edge-End-Resource)

TAPI_FR_000x	**Get Connectivity Service End Point Details**
Description	• Returns attributes of the *ServiceEndPoint* identified by the provided input Id.
Pre-conditions	
Inputs	• *ServiceEndPoint* ID or Name : String - When NULL is provided, this API call should return an error condition

Outputs	• List of IDs, Names, Labels and Extensions (if any) • Administrative, Operational, Control and Usage States • List of supported *Layer-Protocols* including attribute-details indexed by Layer
Notifications	
Error-conditions	
Post-conditions	
Sources	TMF MTNM/MTOSI 3.5 OIF-ONF Demo Topology API Spec oif-p105.11.xx (Get End-point, Edge-End, Edge-End-Resource) OIF Topology API Draft Spec oif-2014.91.xx (Get End-point, Edge-End, Edge-End-Resource)

TAPI_FR_000x	**Get Connectivity Service List**
Description	• Returns list of *Service* entities that represent the connectivity requests that were received • This also includes attribute details for each *Service* including – References to *ServiceEndPoints* terminating the *Service* – Optionally References to any *Connections* realizing the *Service*
Pre-conditions	
Inputs	• Retrieve Scope Filter: *Layer-Protocol* List : Enumeration value - If set/non-empty, the API call will return references to only those encompassed *Services* that support at least one of the specified layer protocols • Include *Connections* : true or false
Outputs	List of *Services* indexed by *Layer* and details for each including: • List of IDs, Names, Labels and Extensions (if any) • Administrative, Operational, and Lifecycle States • Connectivity Requirements such as Layer and Capacity • Connectivity Constraints such as Latency, Cost, etc • Validation characteristics • List of *Service-Ends* and details of each including – Role of the terminating *ServiceEndPoint* in the context of the *Service* – Directionality of the terminating *ServiceEndPoint* in the context of the *Service* – Reference to terminating *ServiceEndPoint* • Optionally List of references to the *Connections* realizing the *Service*
Notifications	
Error-conditions	
Post-conditions	
Sources	TMF MTNM/MTOSI 3.5 OIF-ONF Demo Topology API Spec oif-p105.11.xx (Get Edge) OIF Topology API Draft Spec oif-2014.91.xx (Get Edge)

TAPI_FR_000x	**Get Connectivity Service Details**

Description	• Returns attributes of the *Service* entity identified by the provided input Id, that represents a received connectivity request. • This includes references to *ServiceEndPoints* terminating the *Service*. • This optionally includes references to any *Connections* realizing the *Service*.
Pre-conditions	
Inputs	• *Service* ID or Name : String - When NULL is provided, this API call should return an error condition • Include *Connections* : true or false
Outputs	• List of IDs, Names, Labels and Extensions (if any) • Administrative, Operational, Control and Usage States • Connectivity Requirements such as Layer and Capacity • Connectivity Constraints such as Latency, Cost, etc • Validation characteristics • List of *Service-Ends* and details of each including – Role of the terminating *ServiceEndPoint* in the context of the *Service* – Directionality of the terminating *ServiceEndPoint* in the context of the *Service* – Reference to terminating *ServiceEndPoint* • Optionally List of references to the *Connections* realizing the *Service*
Notifications	
Error-conditions	
Post-conditions	
Sources	TMF MTNM/MTOSI 3.5 OIF-ONF Demo Topology API Spec oif-p105.11.xx (Get Edge) OIF Topology API Draft Spec oif-2014.91.xx (Get Edge)

TAPI_FR_000x	**Get Connection Details**
Description	• Returns attributes of the *Connection* entity identified by the provided input Id. • This includes references to *ConnectionEndPoints* terminating the *Connection*. • This includes references to *Paths* in the underlying *Topology*. • This includes reference to the *Node* containing this *Connection*.
Pre-conditions	
Inputs	• *Connection* ID or Name : String - When NULL is provided, this API call should return an error condition

Outputs	• List of IDs, Names, Labels and Extensions (if any) • Administrative, Operational, Control and Usage States • Connectivity requirements such as Layer and Capacity • Connectivity Constraints such as Latency, Cost, etc • Validation characteristics • Reference to the parent (containing) *Node* • List of *Connection-Ends* and details of each including – Role of the terminating *ConnectionEndPoint* in the context of the *Connection* – Directionality of the terminating *ConnectionEndPoint* in the context of the *Connection* – Reference to terminating *ConnectionEndPoint* • List of *Paths* of the specified *Connection* and details of each including – List of references to lower-level *Connections* that describe the *Path* of the specified *Connection* through the *Nodes* in the underlying *Topology*
Notifications	
Error-conditions	
Post-conditions	
Sources	TMF MTNM/MTOSI 3.5 OIF-ONF Demo Topology API Spec oif-p105.11.xx (Get Edge) OIF Topology API Draft Spec oif-2014.91.xx (Get Edge)

TAPI_FR_000x	**Get Connection End Point Details**
Description	• Returns attributes of *ConnectionEndPoint* identified by the provided input Id. • This includes reference to the *NodeEdgePoint* containing this *ConnectionEndPoint*.
Pre-conditions	
Inputs	• *ConnectionEndPoint* ID or Name : String - When NULL is provided, this API call should return an error condition
Outputs	• List of IDs, Names, Labels and Extensions (if any) • Administrative, Operational, Control and Usage States • List of supported *Layer-Protocols* including attribute-details indexed by Layer • Reference to the parent (containing) *NodeEdgePoint*
Notifications	
Error-conditions	
Post-conditions	
Sources	TMF MTNM/MTOSI 3.5 OIF-ONF Demo Topology API Spec oif-p105.11.xx (Get End-point, Edge-End, Edge-End-Resource) OIF Topology API Draft Spec oif-2014.91.xx (Get End-point, Edge-End, Edge-End-Resource)

3.2.2 Connectivity Request APIs

TAPI_FR_0001	**Create Connectivity Service**

Description	• Causes creation of a *Forwarding-Construct* representing the *Service* request to connect the *Service-End-Points* within the shared *Context* between API Client and Provider • Returns Service ID to be used as reference for future actions • Initial definition will be for a basic point-to-point bidirectional service
Pre-conditions	• Requestor/Client has visibility of the set of *Service-End-Points* between which connectivity is desired within the *Context* • Requestor/Client has information about the types of connectivity available and constraints it can specify such as Service Level • Requestor/Client may be aware of other existing Connectivity *Services* and their IDs
Inputs	• List of *ServiceEnds* and details of each including – Role of the terminating *ServiceEndPoint* in the context of the *Service* – Directionality of the terminating *ServiceEndPoint* in the context of the *Service* – Reference (Name/ID) to terminating *ServiceEndPoint* • Connectivity Requirements such as Layer and Capacity • Connectivity Constraints such as Latency, Cost, etc • Start Time & End Time
Outputs	• Service ID • Operational State • Lifecycle State • Confirmation of Service Characteristics : See above inputs
Notifications	Success/Failure Change of Operational State
Error-conditions	Service not supported Service input not supported Endpoint not recognized
Post-conditions	
Sources	Oif – cite specific documents Onf IETF

TAPI_FR_0002	**Update Connectivity Service**
Description	• Causes modification of an existing *Forwarding-Construct* representing the *Service* request identified by the input *Service* ID • Returns confirmation or rejection of modification
Pre-conditions	• Requestor/Client already knows the existing *Service* ID • Requestor/Client has information about the types of Service Characteristics that can be modified
Inputs	• Service ID or Name • Connectivity Requirements to be modified • Connectivity Constraints to be modified • Start Time & End Time
Outputs	• Success/Failure • Operational State • Lifecycle State • Confirmation of Service Characteristics : See inputs above

Notifications	Success/Failure
Error-conditions	Modification could not be supported Modification parameter not understood Modification not allowed
Post-conditions	Service modified
Sources	

TAPI_FR_0003	**Delete Connectivity Service**
Description	• Causes deletion of an existing *Service* • Returns confirmation of deletion
Pre-conditions	• Requestor/Client already knows the existing *Service* ID
Inputs	• *Service* ID or Name: String
Outputs	• Success/Failure
Notifications	Change of operational mode
Error-conditions	
Post-conditions	Service deleted
Sources	

3.2.3 Connectivity Inputs & Outputs

This section identifies values and formats commonly associated with the parameters of the service request API.

TAPI_FR_000x	**Connectivity Requirements**
Values	The following are the required Connectivity parameters • Service Type: The type of connectivity requested • Service Layer: Represents the layer of transported service • Capacity: Requested bandwidth (fixed or range)

TAPI_FR_000x	**Connectivity Constraints**
Values	The following are the optional Connectivity constraint parameters • Service Level – a abstract value the meaning of which is mutually agreed – typically represents metrics such as - Class of service, priority, resiliency, availability • Latency – integer value and unit - upper bound • Cost – Vector of one or more metrics that would enable the provider to make a decision when implementing the Service • Diversity – an exclude Service ID – indicates that the requested service should be diverse (not share resources) from specified service • Include *Path* – indicates partial set of nodes and links to be used • Exclude *Path* - indicates partial set of nodes and links to be avoided

TAPI_FR_000x	**End-Point TRI format**

Values	The End-Point Name shall minimally support following formats • TRI • URI • Domain-specific String The following are the suggested formats: • ETH - <TBD> • MPLS-TP – <TBD> • ODU – <TBD> • OTS – <TBD>

TAPI_FR_000x	**End Point Role**
Values	Denotes the role of the *End-Point* with respect to the *Forwarding-Construct* • Symmetric • Root • Leaf

TAPI_FR_000x	**End Point Directionality**
Values	Denotes the directionality of the signal-flow in the *End-Point* with respect to the *Forwarding-Construct* • Input • Output • Bidirectional

TAPI_FR_000x	**Layer-Protocol**
Values	The Layer-Protocol attribute shall minimally support following values • OCH • ODU • ETH • MPLS-TP

TAPI_FR_000x	**Capacity (Fixed Bandwidth)**
Values	The Capacity (Bandwidth) attribute is applicable for digital layers and shall minimally support following values in Mbps • 10 • 100 • 1000 • 2400 • 10000 • 40000 • 100000 • 400000

TAPI_FR_000x	**Capacity (non-Fixed Bandwidth)**
Values	• Capacity for ODUFlex or other flexible bandwidth services

TAPI_FR_000x	Administration State
Values	• LOCKED • UNLOCKED

TAPI_FR_000x	Operational State
Values	• ENABLED • DISABLED

TAPI_FR_000x	Lifecycle State
Values	• PLANNED • POTENTIAL • INSTALLED • IN_CONFLICT • PENDING_REMOVAL

3.3 Virtual Network Service

3.3.1 Virtual Network Request APIs

TAPI_FR_00xx	Create Virtual Network
Description	For the client side of the API to request creation of a virtual network from a network (maybe physical or virtual network, recursively) provided by the server side of this API, according to the traffic volume between the access points of the client. As a result, the server side of this API will reserve a set of resources to build up the virtual network, over which the client side of the API is allowed to e.g. configure virtual connections (through other transport APIs).
Pre-conditions	The server side of this API should have the topology information of the network under its control.
Inputs	• Access points: The access points of the client to be connected to the virtual network. • Traffic matrix: A matrix to describe the traffic (e.g. bandwidth) between each pair of the client's access points. • VN SLA: The service level agreement of a virtual network, e.g. the capability of recovery from virtual network topology level failures.
Outputs	• Result of the request: Succeed or fail to create the requested virtual network. • VN ID: The identifier of the virtual network. • Description of VN topology: A detailed description of the created virtual topology, which may include: – Virtual node information: the information of the virtual nodes in the created virtual network, e.g., virtual node ID. – Virtual link information: the information of the virtual links in the created virtual network, e.g., virtual link ID, virtual port ID and bandwidth information.

Notifications	• A notification should be sent to the client when virtual network failure happens (e.g., some virtual link or virtual node failures happen). • A notification should be sent to the client when the virtual network has recovered. •
Error-conditions	• There are not enough resources to set up the virtual network that meets the client traffic requirement.
Post-conditions	• The server side of this API reserves a set of resources to build up the virtual network. • The server side of this API maintains the resources and the status of the created virtual networks, as well as the mapping relationship between the created virtual networks and the network under control of the server side. • The client side of this API is allowed to have virtual connection control over the virtual network.
Sources	TBD

TAPI_FR_00xx	**Update Virtual Network**
Description	TThe client side can modify the virtual network which has already beencreated. Such modification may include adding and/or deleting virtual nodes and/or virtual links, modifying link capacity of virtual links, SLA of the virtual network, etc.. As a result, the server side of this API will modify the reservation of resources under its control to form the modified virtual network, according to the new traffic matrix.
Pre-conditions	• The server side of this API has the topology information of the network under its control. • The client side of this API has already requested a virtual network from the server side of this API successfully.
Inputs	• VN ID: The identifier of the virtual network to be modified. • Modified access points: The access points of the client to be connected to the modified virtual network. • Modified traffic matrix: A matrix to describe the modified traffic (e.g. bandwidth) between each pair of client's access points. • Modified VN SLA: The service level agreement of the modified virtual network, e.g. the capability of recovery from virtual network topology level failures.
Outputs	• Result of the modification request: Succeed or fail to modify the virtual network. • VN description: A detailed description of the modified virtual network, which may include: – Virtual node information: the information of the virtual nodes in the created virtual network, e.g., virtual node ID. – Virtual link information: the information of the virtual links in the created virtual network, e.g., virtual link ID, virtual port ID and bandwidth information.
Notifications	• A notification should be sent to the client when virtual network failure happens (e.g., some virtual link or virtual node failures happen). • A notification should be sent to the client when virtual network has recovered.
Error-conditions	• The requested VN (to be modified) ID does not exist at the server side. • There are not enough resources to support modification of the virtual network .

Post-conditions	• The server side of this API modifies the reservation of resource under its control to form the modified virtual network. • The server side of this API maintains the resource and the status of the modified virtual network, as well as the mapping relationship between the modified virtual network and the network under control of the server side. • The client side of this API is allowed to have virtual connection control over the modified virtual network.
Sources	TBD

TAPI_FR_00xx	**Delete Virtual Network**
Description	For the client side of the API to delete the virtual network that it owns. As a result, the server side of this API will release the resources used by this virtual network.
Pre-conditions	• The server side of this API has the topology information of the network under its control. • The client side of this API has requested a virtual network from the server side of this API. • All virtual connections in the virtual network should be deleted by the client before deleting the virtual network.
Inputs	• VN ID: The identifier of the virtual network to be deleted
Outputs	• Result of the request: Succeed or fail to delete the requested virtual network.
Notifications	N/A
Error-conditions	• The requested VN (to be deleted) ID does not exist at the server side. • One or more virtual connections remain in the virtual network.
Post-conditions	• The server side of this API releases the resources used by the virtual network.
Sources	TBD

3.4 Path Computation Service

To define a Path Computation APIs, some assumption would be needed to make clear relationship with the other API, like Connection Control APIs. To do that we cannot avoid to consider how different component can interact together in case a service (i.e. a call) has to be implemented. It is supposed that a connection controller has in charge the management of the connection in terms of real implementation (resource commitment) of the different routes (paths) that a routing controller can provide as output of an APIs of path computation.

If we consider a routing controller as in charge of path computation only of a connection controller request without real implementation of the connection based on that path, some specific APIs can be considered as not part of path computation but in the context of connection control, i.e. connection deletion, connection query, maintenance event management, etc.

The Service ID , we define here as input parameter in the path computation, it is call identifier needed to scope the context of the paths (routes) the APIs will provide as output. The Service with some service characteristics corresponds to the ASON/GMPLS call concept.
A Path is considered as" ordered list of TE-links" that means node IDs + ifindex.
An e2e Connection is composed by a concatenation of link resources (associated with a TE-link) and XCs in the different nodes.

3.4.1 Path Computation Request APIs

TAPI_FR_xxxx	Compute P2P Path
Description	Path computation can be called in the context of service request since path computation is provided in a domain according to the policy which has to refer to specification of service for which the path computation request is required. The client side of the API can request the server side of the API to compute a single path or a batch of paths with consideration of a set of constraints
Pre-conditions	The server side of this API should have the topology information (including TE information) of the network in which the path computation applies. Includes Connectivity matrix,port label restriction (only applicable to optical layer path computation [1]

Inputs	Names/Identifiers • A End-Point (nodes, link end , or end points) • Z End-Point (nodes, link end, or end points) Connectivity Requirements • LayerProtocol • ClientProtocol (G-PID) • Traffic Parameters : e.g.Signal Type (L1), Frequency Range,,Available labels… (L0) • Capacity • shouldComputeConcurrentPaths Routing Constraints: not all constraints will be needed , some will be mandatory other optional, it may happen that path computation will choose to prioritize cost sometime and latency other times. • Latency (max latency permitted) • Cost • Service Level • Protection Type : protection(Unprotected,1+1 protection,1:1 protection,shared-mesh restoration)/restoration • SRLG/Diversity (link, Node) • Explicit Path • Exclude (Node/link) • Include (Node/link) • Optical impairments: e.g. power level, OSNR, CD, PMD, etc. (only applicable to optical layer path computation)Prioritize selection of lambdas that have been used by existing connections (only applicable to optical layer path computation) • Co-routing (e.g. two service must share the same add/drop group in a ROADM) • Multi-layer ? Objective function : • Minimize the cost • Maximize the amount of successfully computed paths (Only for concurrent path computation) • Minimize aggregate Bandwidth Consumption (Only for concurrent path computation) • Minimize the load of the Most Loaded Link (Only for concurrent path computation) • Minimize Cumulative Cost of a set of paths (Only for concurrent path computation) Directionality (unidir/bidir)

Outputs	List of paths computed containing following information (only "one" if shouldComputeConcurrentPaths = false) • Path identifier (identifier of the calculated route) • Routing constrains (Description of routing constraints that are met) • Directionality (bidir, unidir) • Connection ERO (ordered list of strict hopes (nodes, link, link ends) ,and loose hopes/virtual node) Success
Outputs	Failure
Notifications	
Error-conditions	Cause of failure
Post-conditions	
Sources	Oif Onf IETF

TAPI_FR_xxxx	**Optimize P2P Path**
Description	A connection can be reconfigured to meet new constraints and achieve path optimization via this API. [Reconfiguration may involve end-point changes and route changes.-->to be discussed further]
Pre-conditions	The server side of this API should have the topology information (including TE information) of the network in which the path computation applies

Inputs	Names/Identifiers Path ID (identifier of route to be modified) Connection identifier (?) → optional : mainly used to identify resources in use for the connection that could be shared for the new optimized path Connectivity Requirements • Encoding Type, Switching Type, G-PID, Traffic Parameters(e.g.Signal Type) (L1) • Frequency Range, Connectivity matrix,port label restriction ,Available labels… (L0) • Bandwidth (packet) • Routing Constraints (to specify criteria for reconfiguration/optimization) • Latency (max latency permitted) • Cost • CoS (protection/restoration) • SRLG/diversity (link, Node) • Explicit Path • Exclude (Node/link), • Include (Node/link) • Flag resource sharing(max re-usage/min re-usage) Whether new resource can be used or no [Sergio note: it seems the same, I would delete this one and take explicit flag, are you agree?] • Minimum/maximum link utilization value [Sergio note: not sure these two can be considered as constrain for optimization] Optimization Time requirement; - Whether interruption allowed or not ; ; Optimization objective functions：this information indicates the objective of this optimization. Possible objective functions may include but not limited to: • Maximize the amount of successfully computed paths • Minimize aggregate Bandwidth Consumption • Minimize the load of the Most Loaded Link • Minimize Cumulative Cost of a set of paths
Outputs OK	Path identifier (new path id will be linked to the same connection identifier)or Routing constrains (Description of routing constraints that are met) Directionality (bidir, unidir) Connection ERO (ordered list of strict hopes (nodes, link, link ends) ,and loose hopes/virtual node) Success
Outputs NOK	Failure
Notifications	

Error-conditions	Cause of failure: 1、 Optimization fail due to insufficient resources 2、 Cannot readjust resource allocation without interruption of existing services. 3、 Cannot satisfy other constraints, such as timing issue or performance threshold.
Post-conditions	
Sources	Oif Onf IETF

3.5 Notification Service

<TBD>

4 Appendix A: Transport API Concepts Overview

4.1 Node and Topology Aspects of Forwarding Domain

The *Forwarding-Domain* represents the opportunity to enable forwarding between points represented by the *LTPs* available at the boundary of the *Forwarding-Domain*. The *Forwarding-Domain* can hold zero or more instances of *Connections* and provides the context for requesting and instructing the formation, adjustment and removal of *Connections*.

The *Forwarding-Domain* supports a recursive aggregation relationship such that the internal construction of a *Forwarding-Domain* can be exposed as multiple lower level *Forwarding-Domains* and associated *Links* (partitioning). A *Forwarding-Domain* within a Network-Element may represent a switch matrix/fabric as well as those cases where the matrix is itself decomposed so the *Forwarding-Domain* may be smaller than the scope of the matrix. On the other-end of the aggregation relationship spectrum, it is expected that there will always be an all-encompassing top-most Forwarding-Domain that corresponds to the scope of the entire *Network-Control-Domain* (*Context*)

For the purposes of API requirements, the two aspects of the *Forwarding-Domain* have been refactored as separate entities:

- Node – which represents the forwarding-potential between its edge-points (*LTP*s)
- Topology – which represents the topological-aggregation of lower-level *Links* and *Nodes*

Depending on the frame of reference for an API invocation (or the position of an imaginary observer), only the opaque *Node*-aspects of a *Forwarding-Domain* would be visible (placing the observer external to the *Forwarding-Domain*) or the entire *Topology*-structure of a *Forwarding-Domain* would be visible (placing the observer internal to the Forwarding-Domain)

An instance of *Forwarding-Domain* is associated with zero or more *LTP* instances.

The effective adjacency between two or more *Forwarding-Domains* is modeled by a *Link*. In its basic form (i.e., point-to-point Link) it associates a set of *LTP* client layers on one *Forwarding-Domain* with an equivalent set of *LTP* client layers on another *Forwarding-Domain*. A *Link* may offer parameters such as capacity and delay depending on the type of technology that supports the *Link*.

A *Forwarding-Domain* may aggregate *Links* that are wholly within the bounds of the *Forwarding-Domain*. A *Link* with an Off-network end cannot be encompassed by a *Forwarding-Domain.*

The association of the *Link* to *LTPs* is made via the *Link-Ends* which are essentially the ports of the *Link*. In T-API IM, the *Link-Ends* are refactored into the *Link* itself so that the *Link* maintains the *Link-End* attributes (such as link-end *Role*) and has direct association to its terminating *LTPs* (*NodeEdgePoints*).

The *Link* can support multiple transport layers via its associated *LTP* instance.

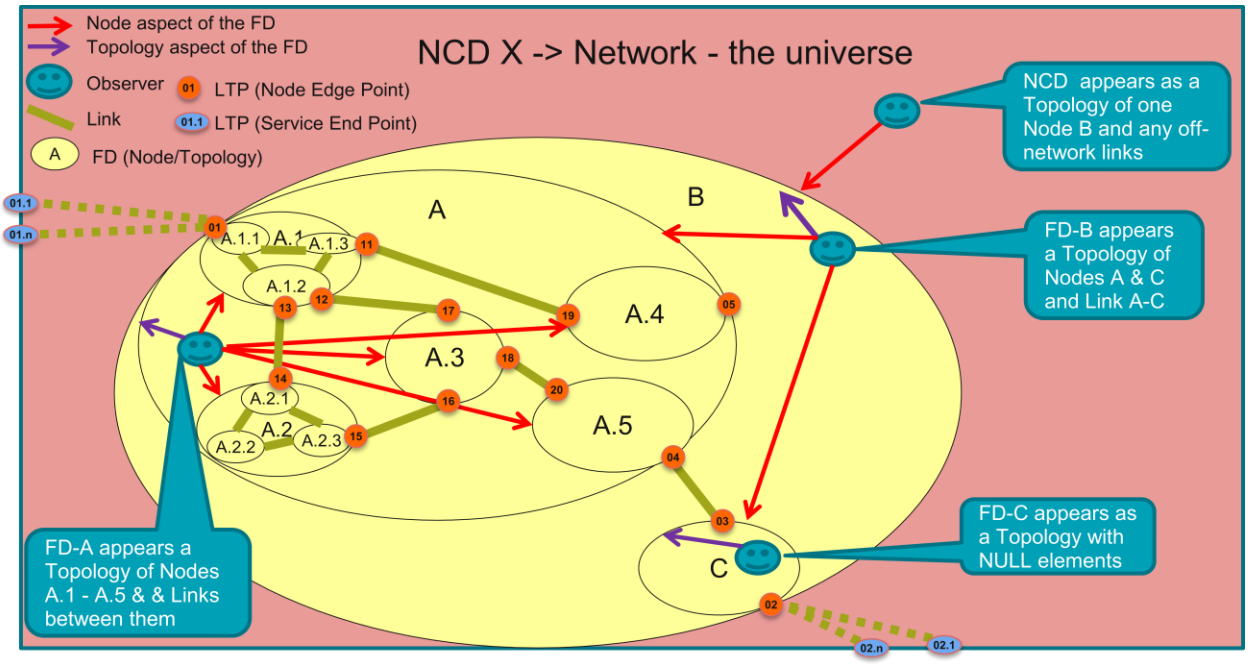

Figure 2: Node/Topology perspectives of recursively partitioned Forwarding Domain (FD)

4.2 Network Control Domain and Contexts

The T-API *Context* is a realization of the *Network-Control-Domain* as defined in the ONF Core Information Model and defines the scope of control and naming that a particular SDN controller or application module has with respect to a specific instance of network abstraction.

In interfaces where an abstracted view of network is offered, e.g. in client/server SDN controller relationship, the *Context* entity defines the scope of control of the client SDN controller on the abstracted/virtual network view that has been provided by the server SDN controller. Thus *Context* relates to an abstraction of the partitioned provider resources allocated to that particular client. In such cases, the *Context* also provides the scope of naming space for identifying objects representing the (virtual) resources within the (virtual) network view.

Often, a provider controller may also create and maintain multiple abstractions of its controlled network for its own purposes and not necessarily to service any particular business clients.

The following figures illustrate few examples of NCD *Contexts* in a hierarchical SDN controller system:

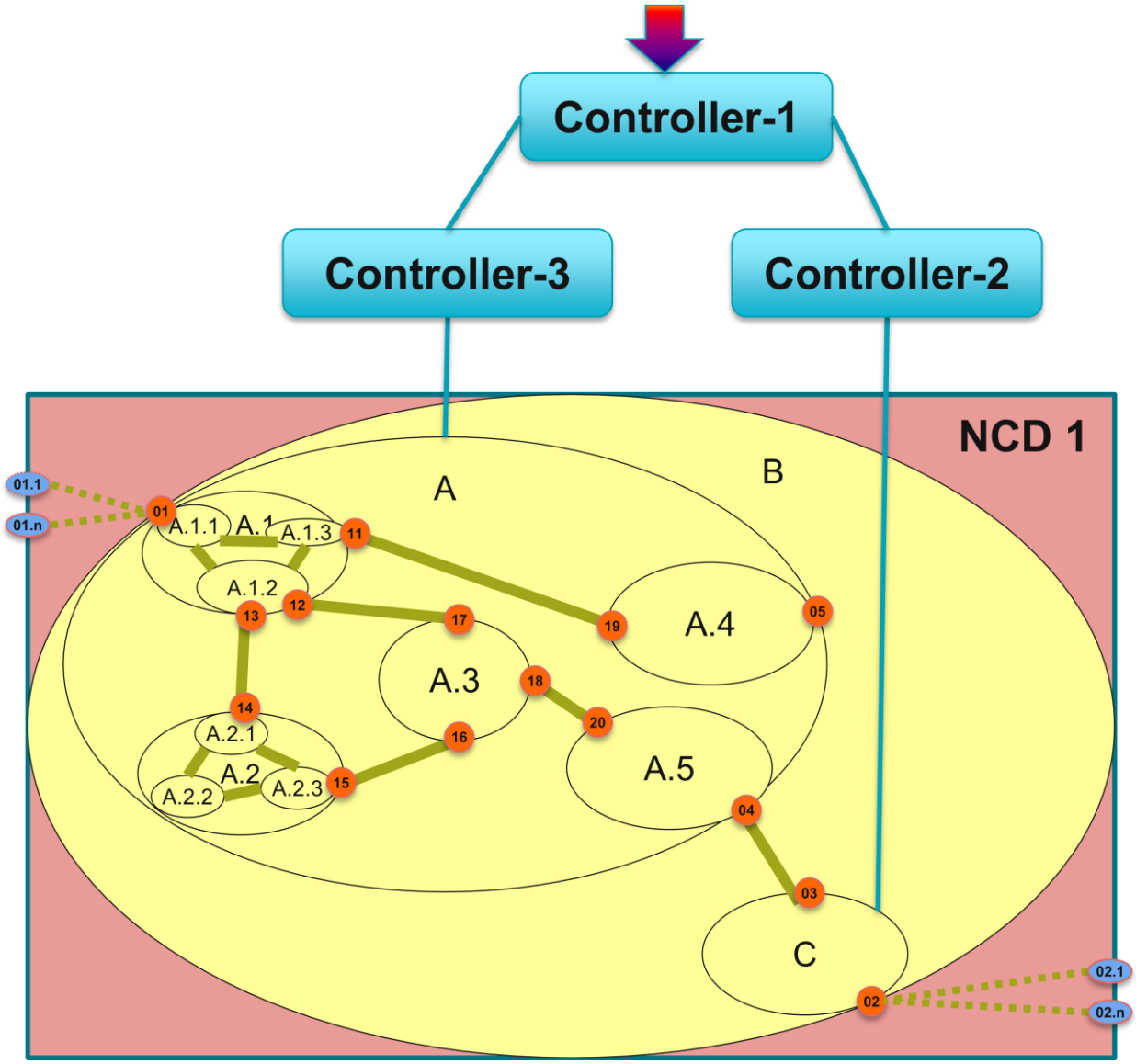

Figure 3: View of Controller-1 NCD based on Views exported by Controllers 2 & 3

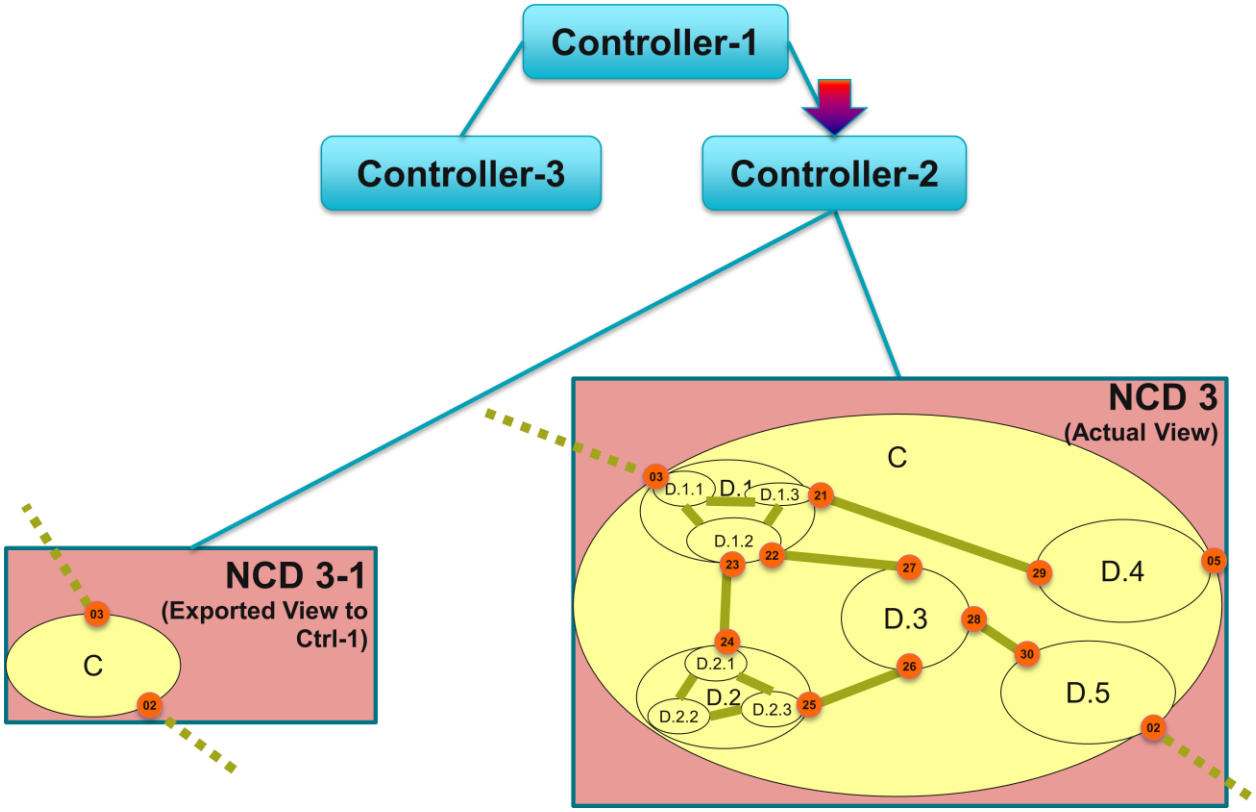

Figure 4: Views of Controller-2 NCDs

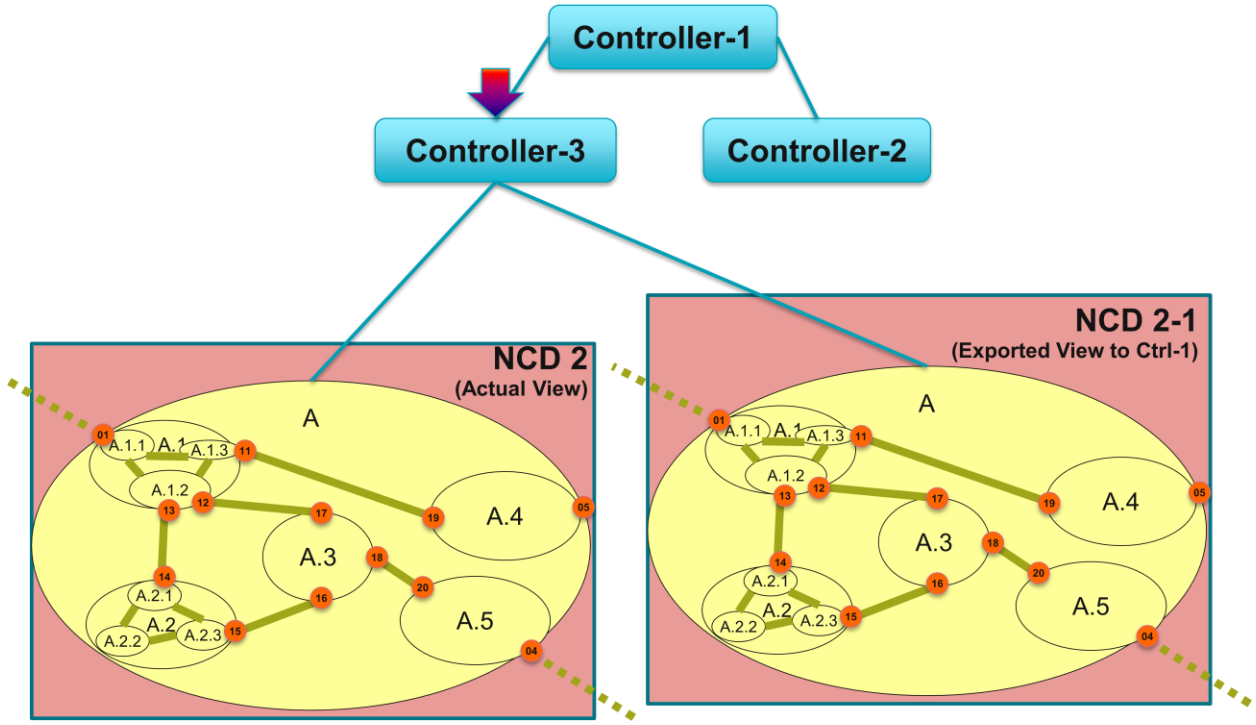

Figure 5: Views of Controller-3 NCDs

4.3 Topology Traversal using APIs

The following figures illustrate few examples of views of a provider topology, that a T-API client application may obtain using the APIs.

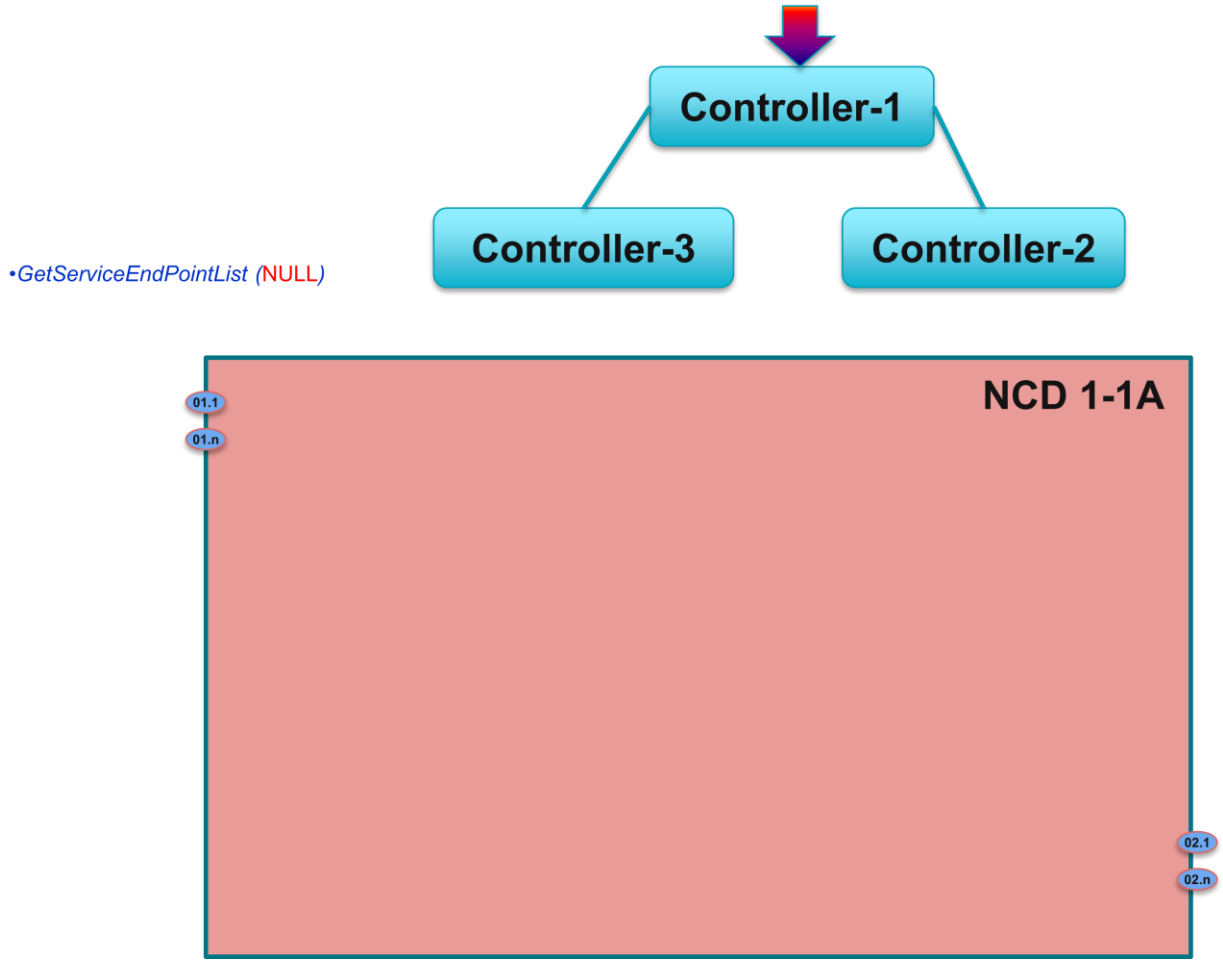

Figure 6: API Client's View of Controller-1 NCD without retrieving Topology details

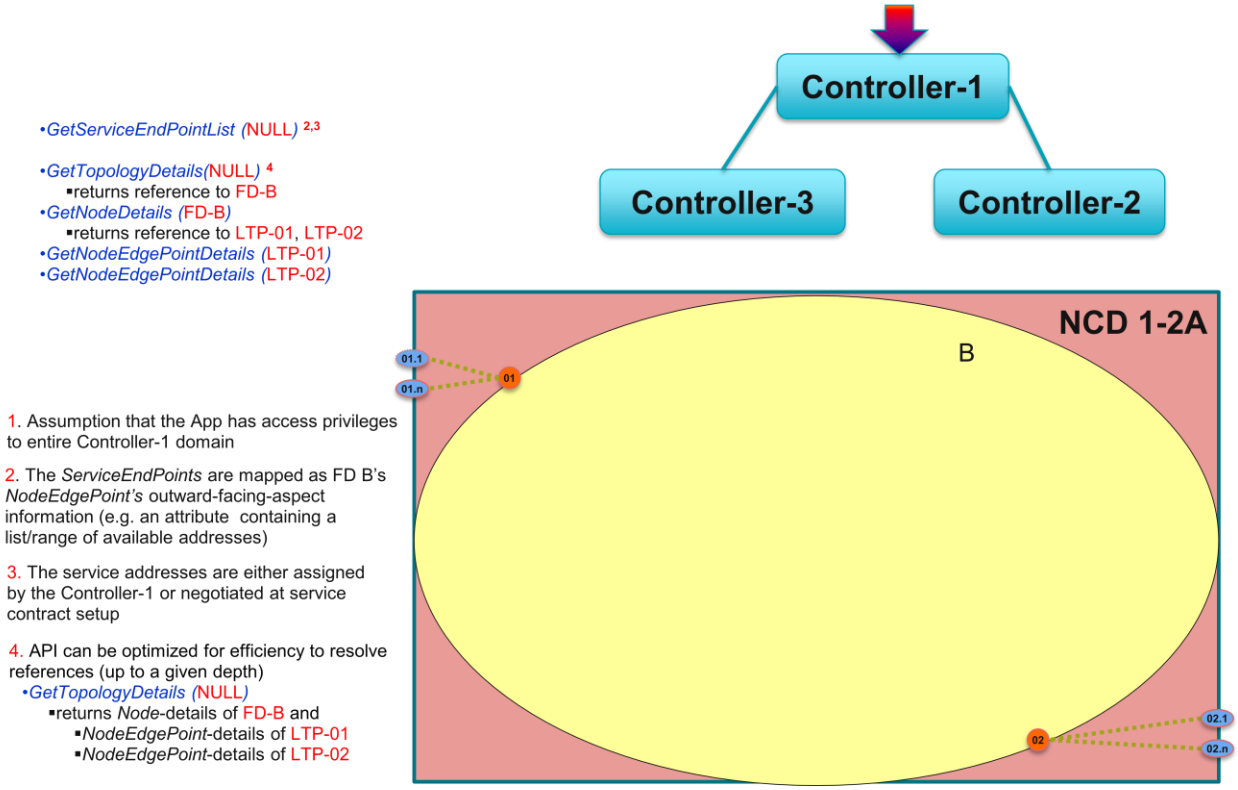

Figure 7: API Client's View of Controller-1 NCD by retrieving top-most level of Topology details

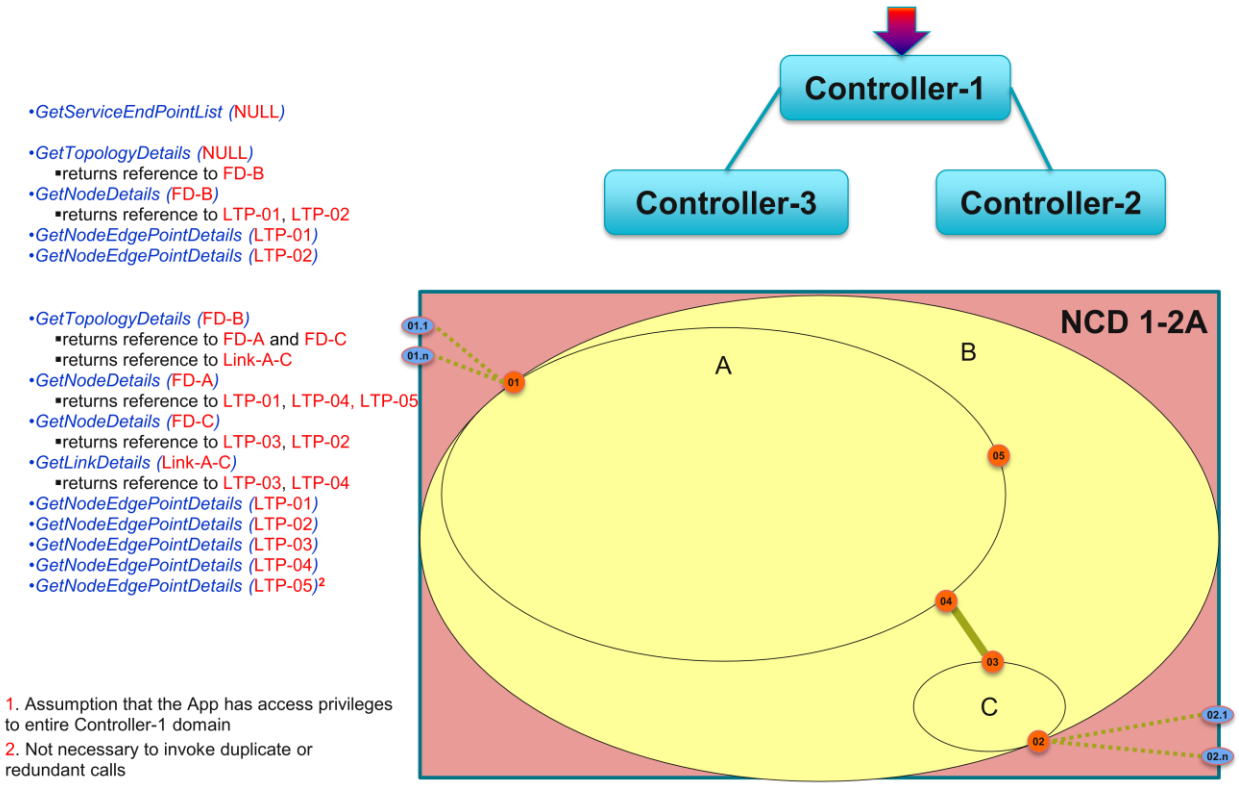

Figure 8: API Client's View of Controller1 NCD by retrieving 2 levels of Topology details

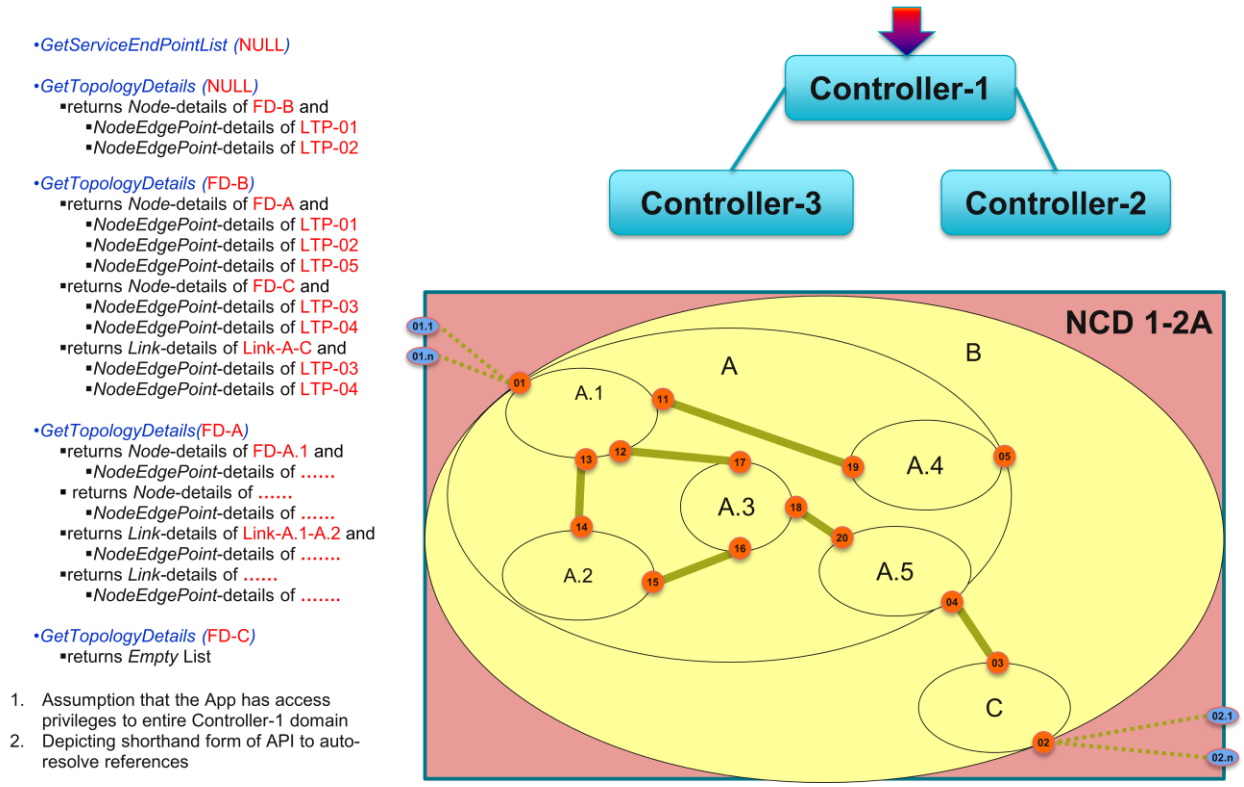

Figure 9: API Client's View of Controller-1 NCD by retrieving 3 levels of Topology details

4.4 Service, Connection and Path

A *Service* represents an "intent-request" for connectivity between two or more *Edge-Points* (*LTPs*) exposed by the all-encompassing top-most *Forwarding-Domain* corresponding to the scope of the entire *Network-Control-Domain*. As such, *Service* is distinct from the *Connection* that realizes the *Service*. The requestor of the *Service* is expected to be able to express their intent using just an "external" view of *Forwarding-Domain* and its advertised *Edge-Points* and not require knowledge of the "internal" topology of the *Forwarding-Domain*.

The association of the *Service* to *LTPs* is made via the *Service-End-Points* which are essentially the ports of the *Service*.

The *Service* is modeled by the *Forwarding-Construct* entity defined in the ONF Core Information Model.

The *Connection* represents an enabled potential for forwarding (including all circuit and packet forms) between two or more Edge-Points (*LTPs)* of a *Forwarding-Domain*. A connection request typically utilizes its knowledge of the "internal" topology view of FD.

The association of the *Connection* to *LTPs* is made via *Connection-End-Points* (essentially the ports of the *Connection*) where each *End-Point* of the *Connection* has a role in the context of the *Connection*. The traffic forwarding between the associated *End-Points* of the *Connection* depends upon the type of *Connection*.

The *Connection* can be used to represent many different structures including point-to-point (P2P), point-to-multipoint (P2MP), rooted-multipoint (RMP) and multipoint-to-multipoint (MP2MP) bridge and selector structure for linear, ring or mesh protection schemes.

A *Connection* can be in only one *Forwarding-Domain*.

A *Connection* supports a recursive aggregation relationship such that the internal construction of a *Connection* can be exposed as multiple lower-level *Connection* instances (partitioning). A *Connection* can have zero or more *Paths*, each of which is defined as a list of lower level *Connection* instances. At the lowest level of recursion, a *Connection* represents a cross-connection within a switch matrix/fabric in a Network Element.

The *Connection* is modeled by the *Forwarding-Construct* entity defined in the ONF Core Information Model.

The *Path* represents the individual routes of a *Connection*. It is represented by a list of *Connections* at a lower level. Note that depending on the *Service* supported by a *Connection*, the *Connection* can have multiple routes.

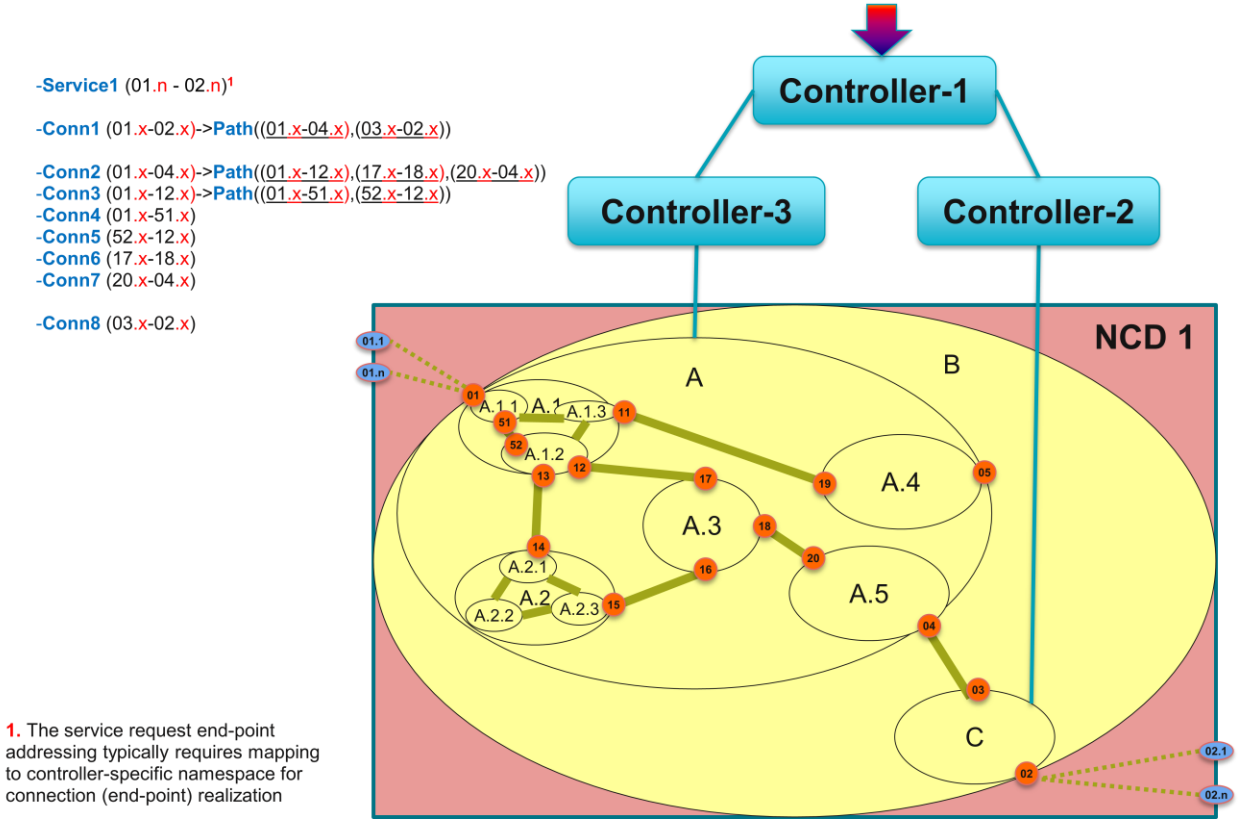

Figure 10: Service & Connections from Controller-1 perspective

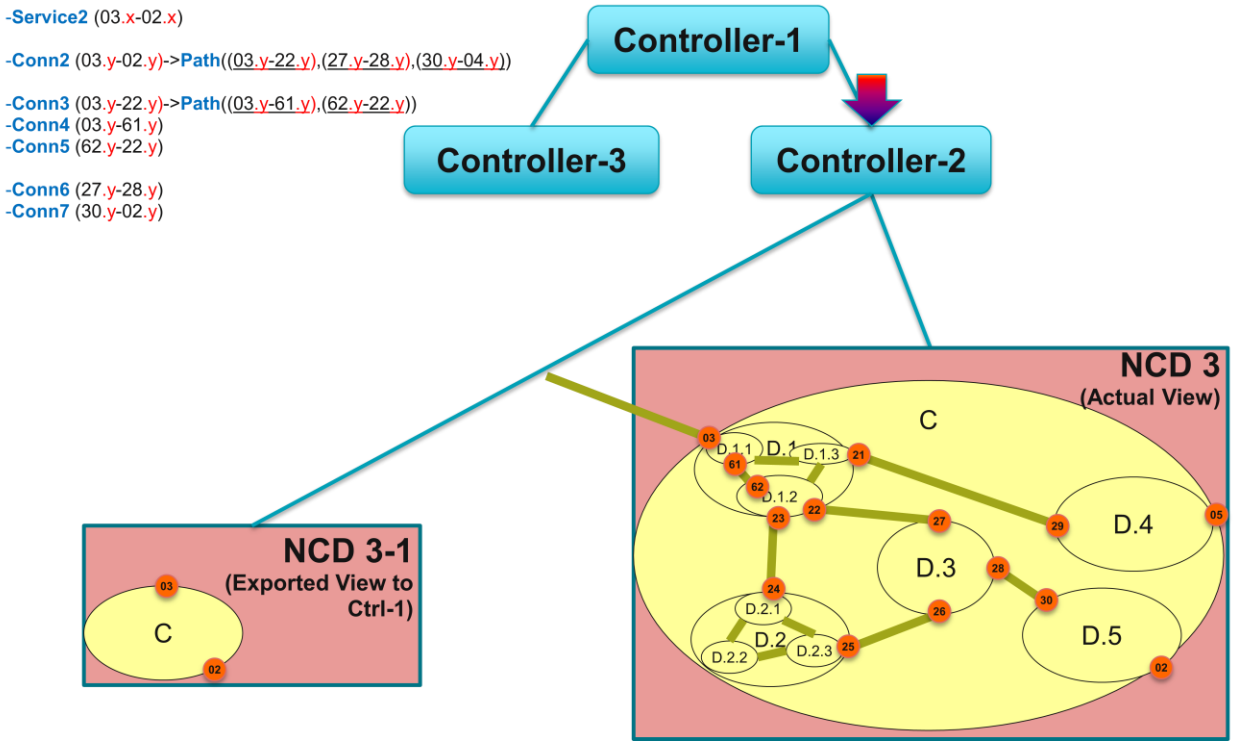

Figure 11: Service & Connections from Controller-2 perspective

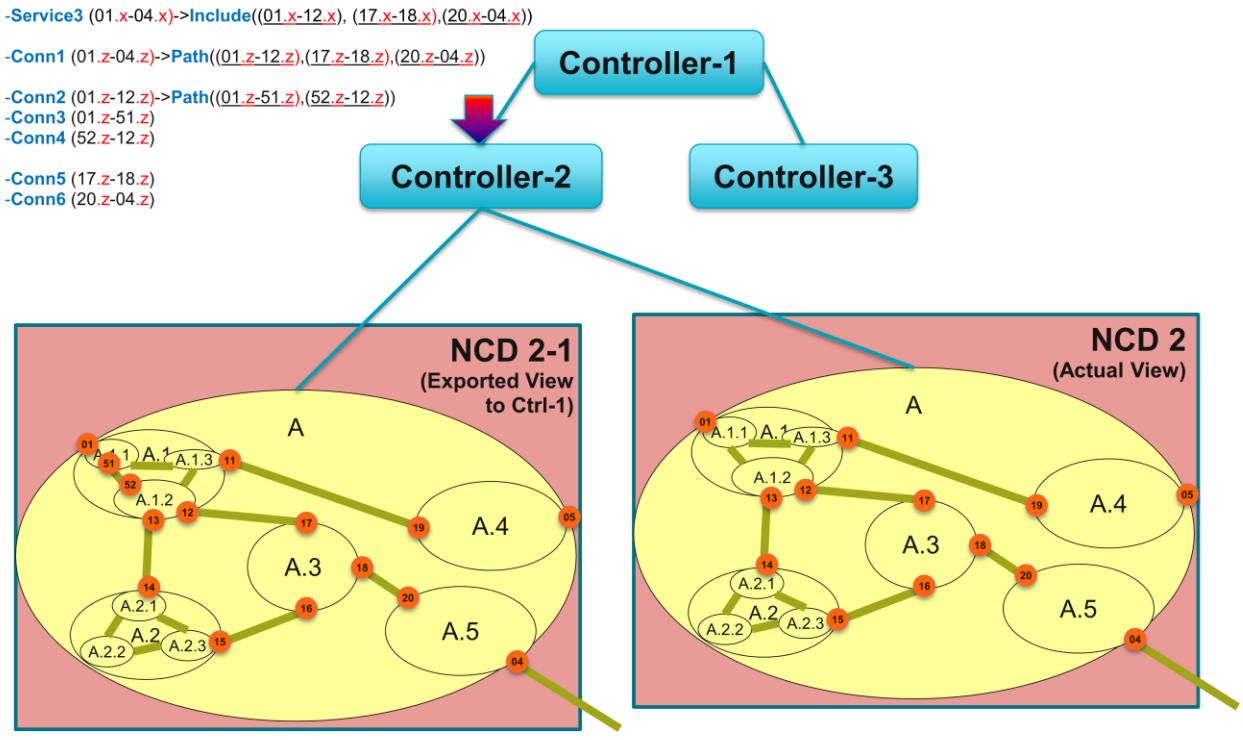

Figure 12: Service & Connections from Controller-3 perspective

4.5 Node Edge Point v/s Service End Point v/s Connection End Point

The *Logical-Termination-Point* (*LTP*) represents encapsulation of the addressing, mapping, termination, adaptation and OAM functions of one or more transport layers (including circuit and packet forms). Where the client – server relationship is fixed 1:1 and immutable, the different layers can be encapsulated in a single *LTP* instance. Where there is a n:1 relationship between client and server, the layers are split over separate instances of *LTP*.

Functions that can be associated/disassociated to/from an *Connection*, such as OAM, protection switching, and performance monitoring are referenced as secondary entities through the associated *LTP* instance.

5 Appendix B: Transport API Examples Use cases

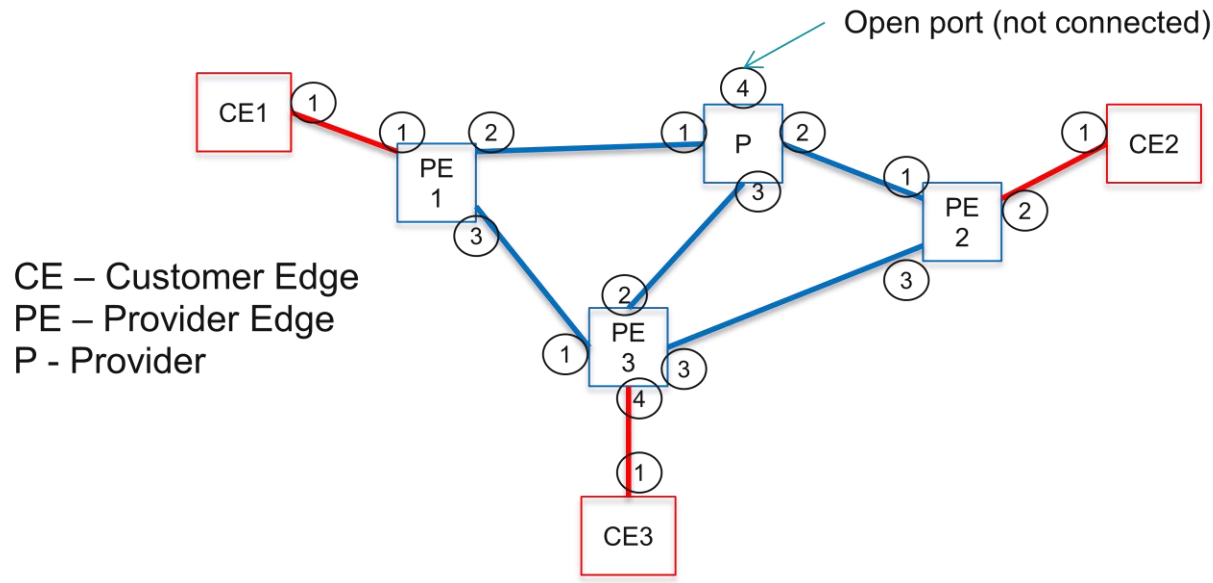

Figure 13: Example Physical Network Topology

5.1 10G EPL Service over ODU2 Connection over 100G OTN network

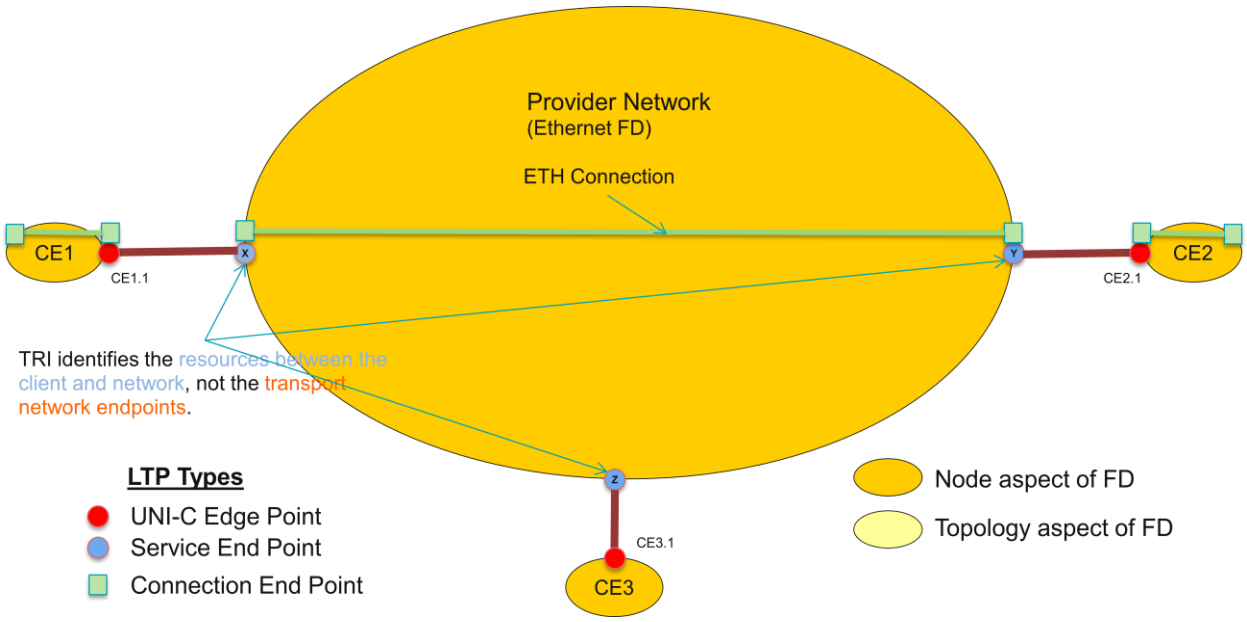

Figure 14: Customer View of Topology and Connectivity

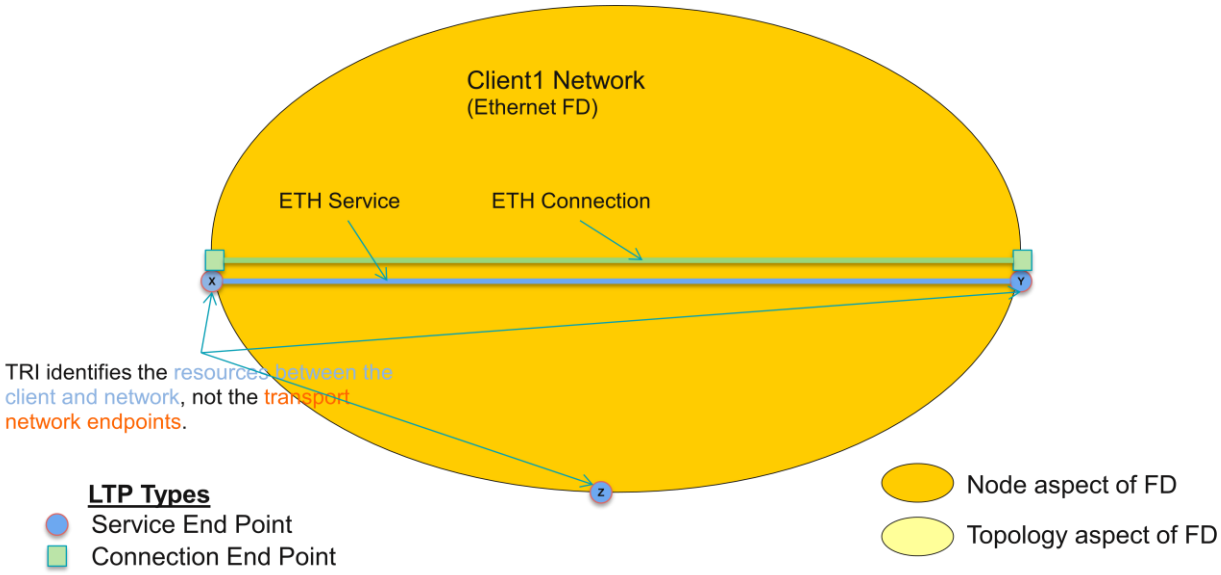

Figure 15: Provider's View of Topology and Service/Connections exported to the Customer

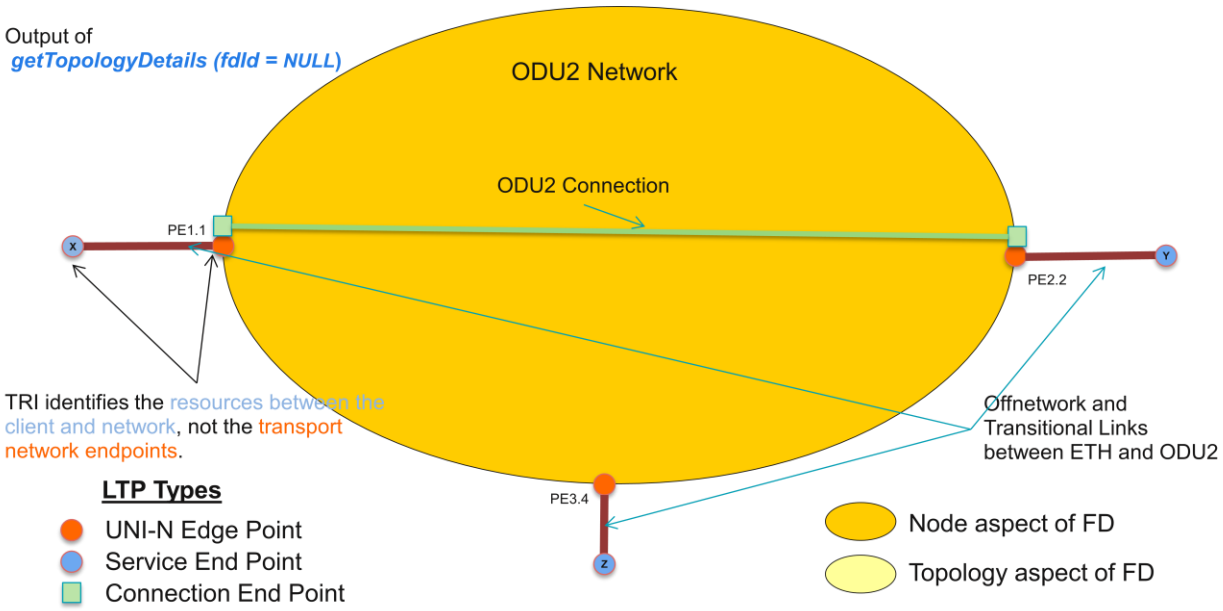

Figure 16: Provider's View of its Internal top-level ODU2 Layer Topology and Connections

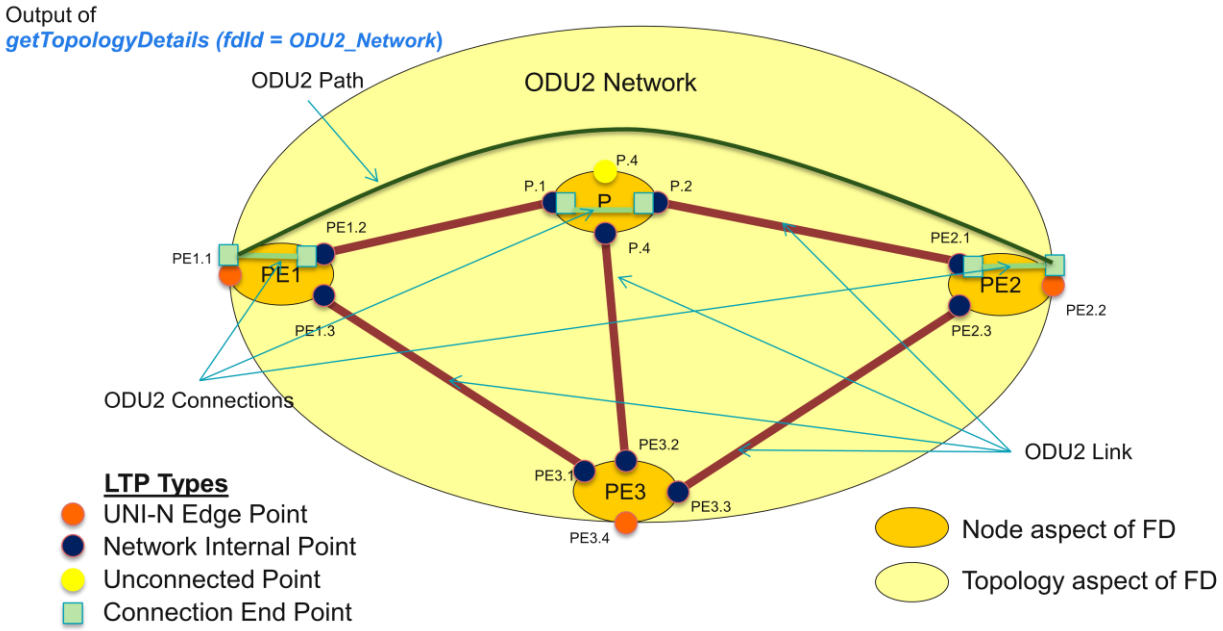

Figure 17: Provider's View of its Internal 2nd-level ODU2 Layer Topology and Connections

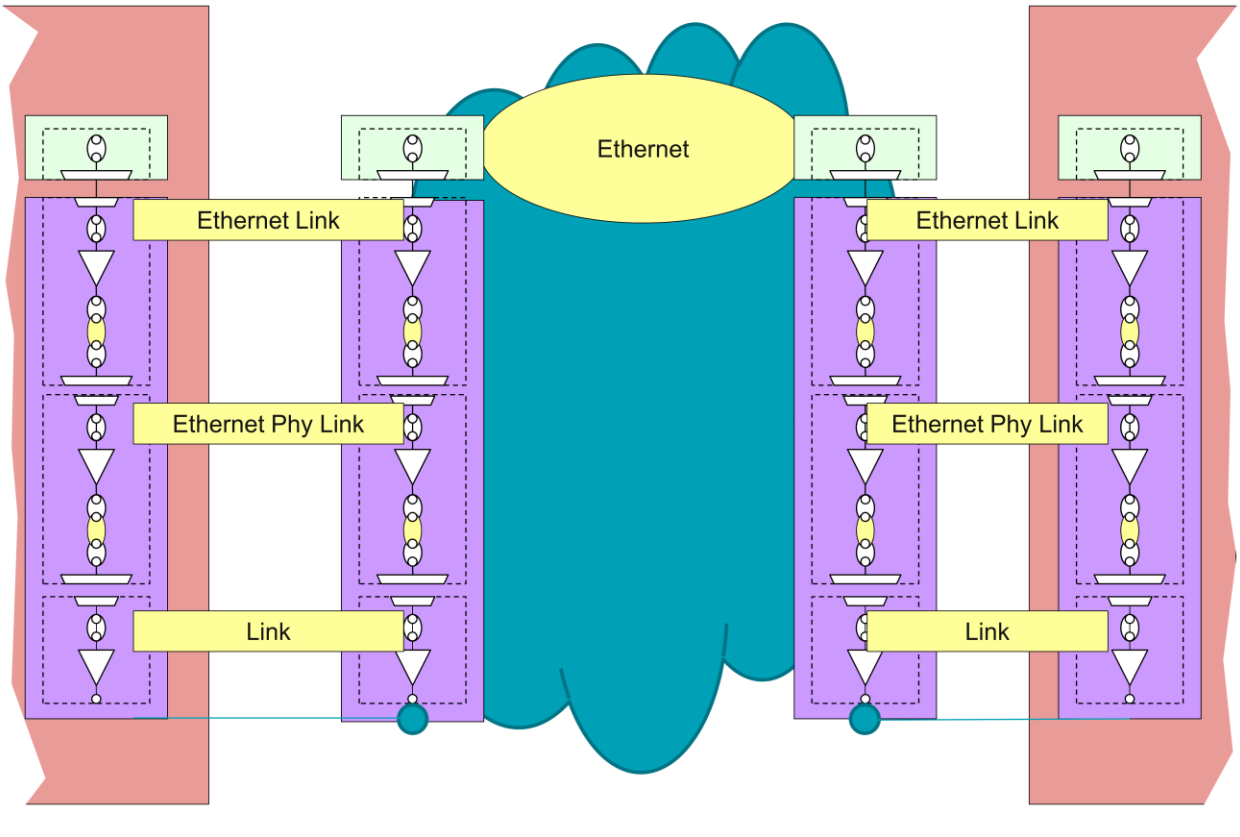

Figure 18: External perspective of UNI Port (NodeEdgePoint) Layers, Links & Switching

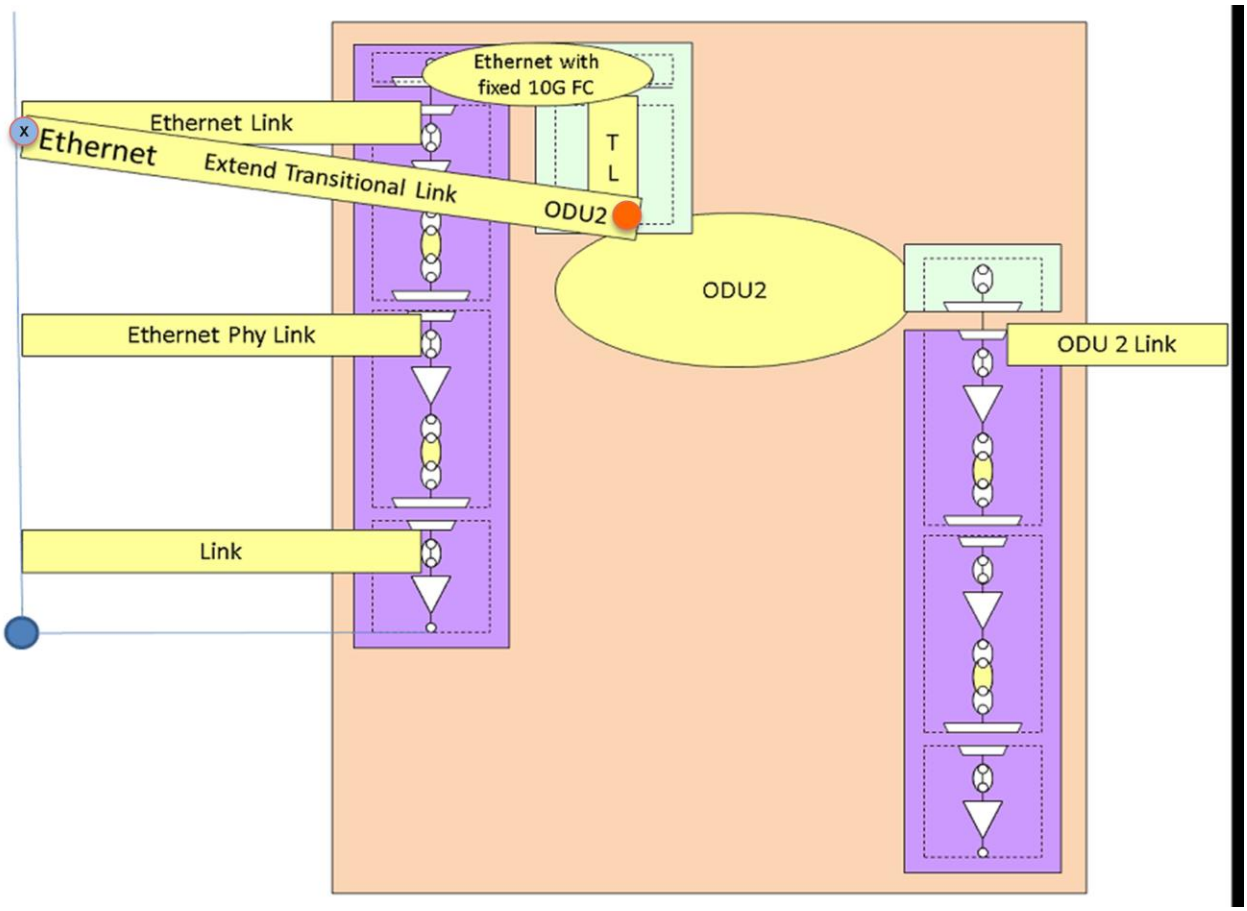

Figure 19: Transitional perspective of UNI-P Port (NodeEdgePoint) Layers, Links & Switching

5.2 1G EVPL Service over ODU0 Connection over 100G OTN network

5.3 Var-rate EVPL Service over EVC Connection over 100G OTN network

6 Appendix C: Transport API Information Model Skeleton

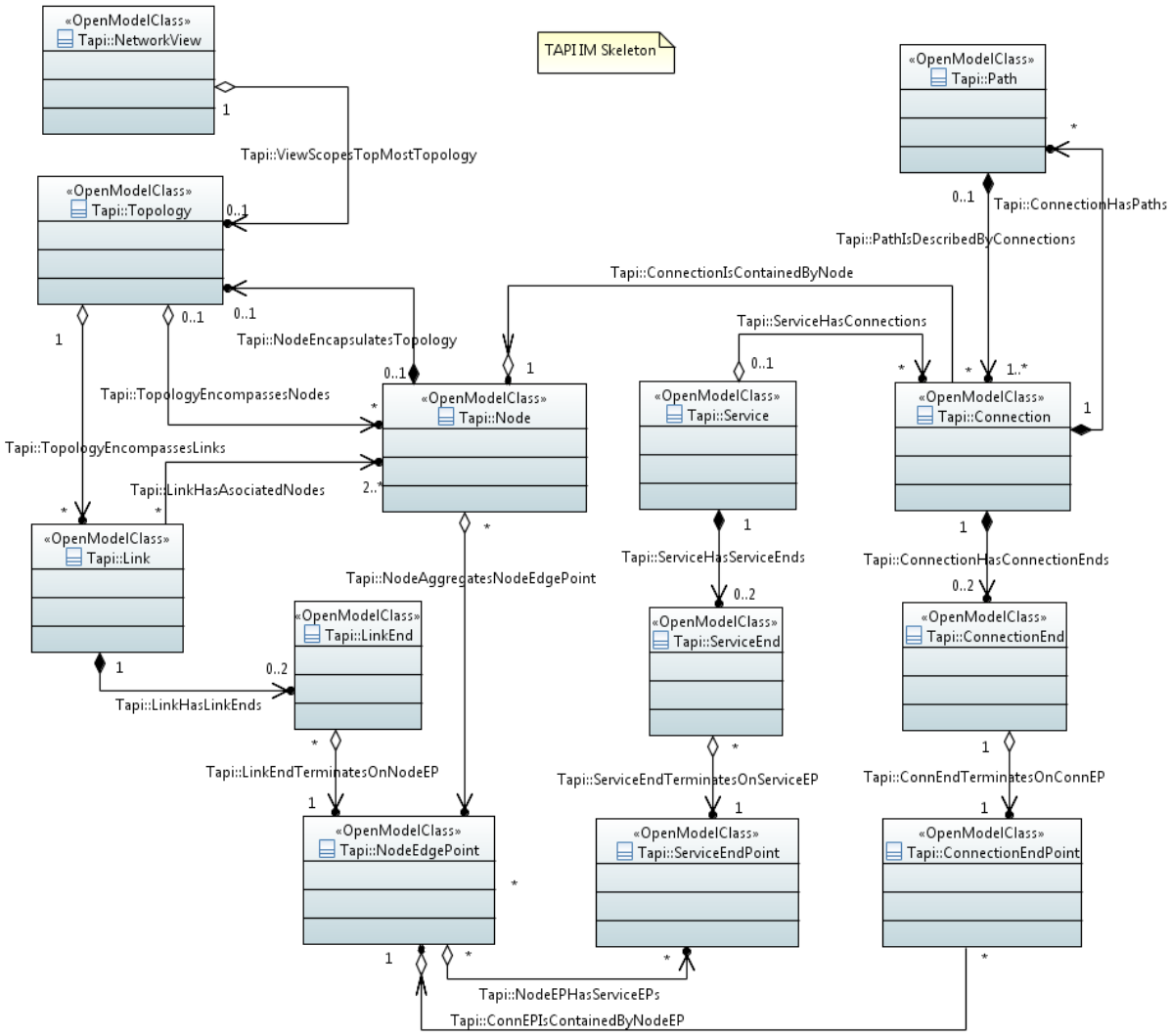

Figure 20: Transport API Information Model Skeleton

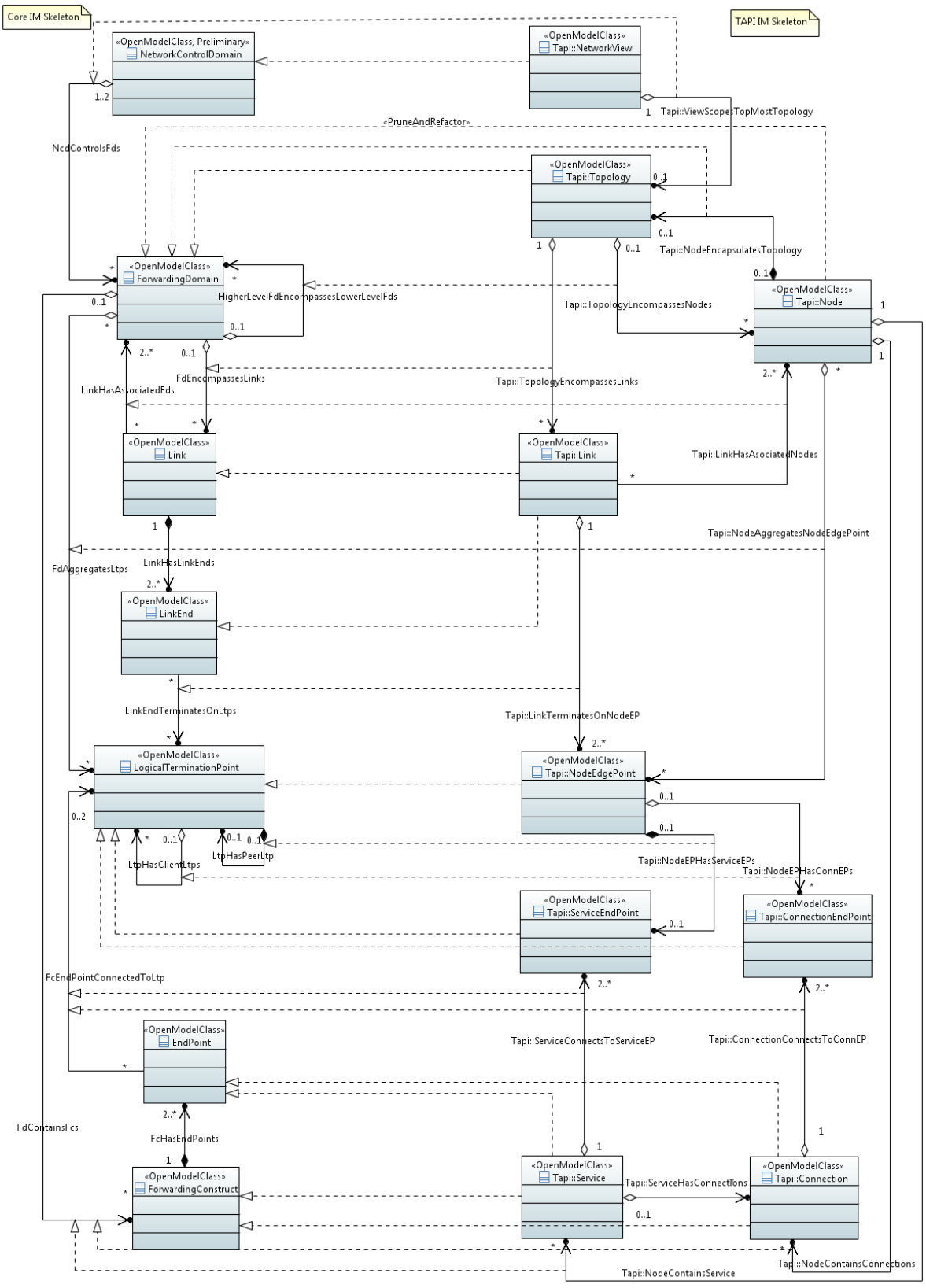

Figure 21: Transport API IM Mapping to Core IM

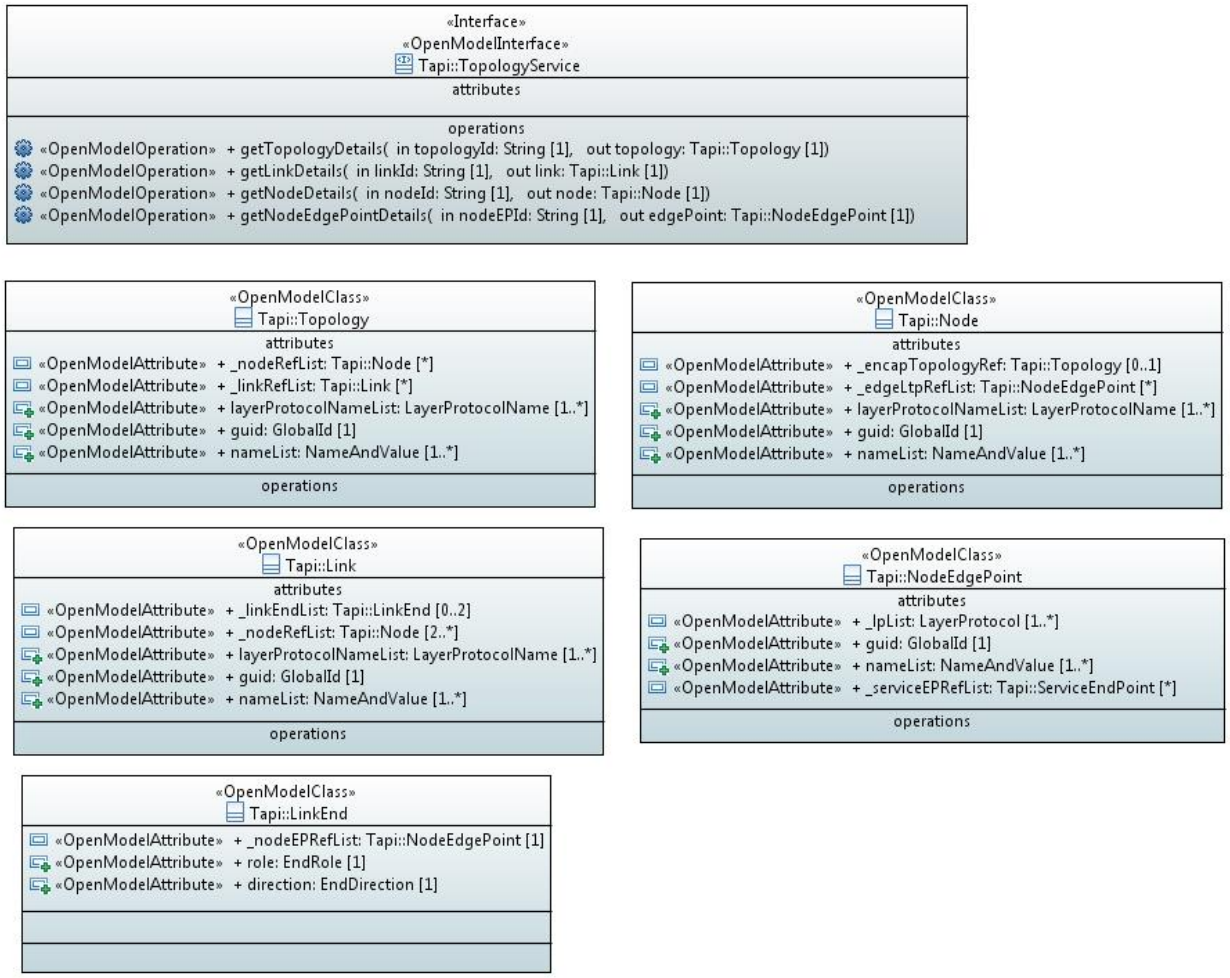

Figure 22: Topology Service API: IM Skeleton

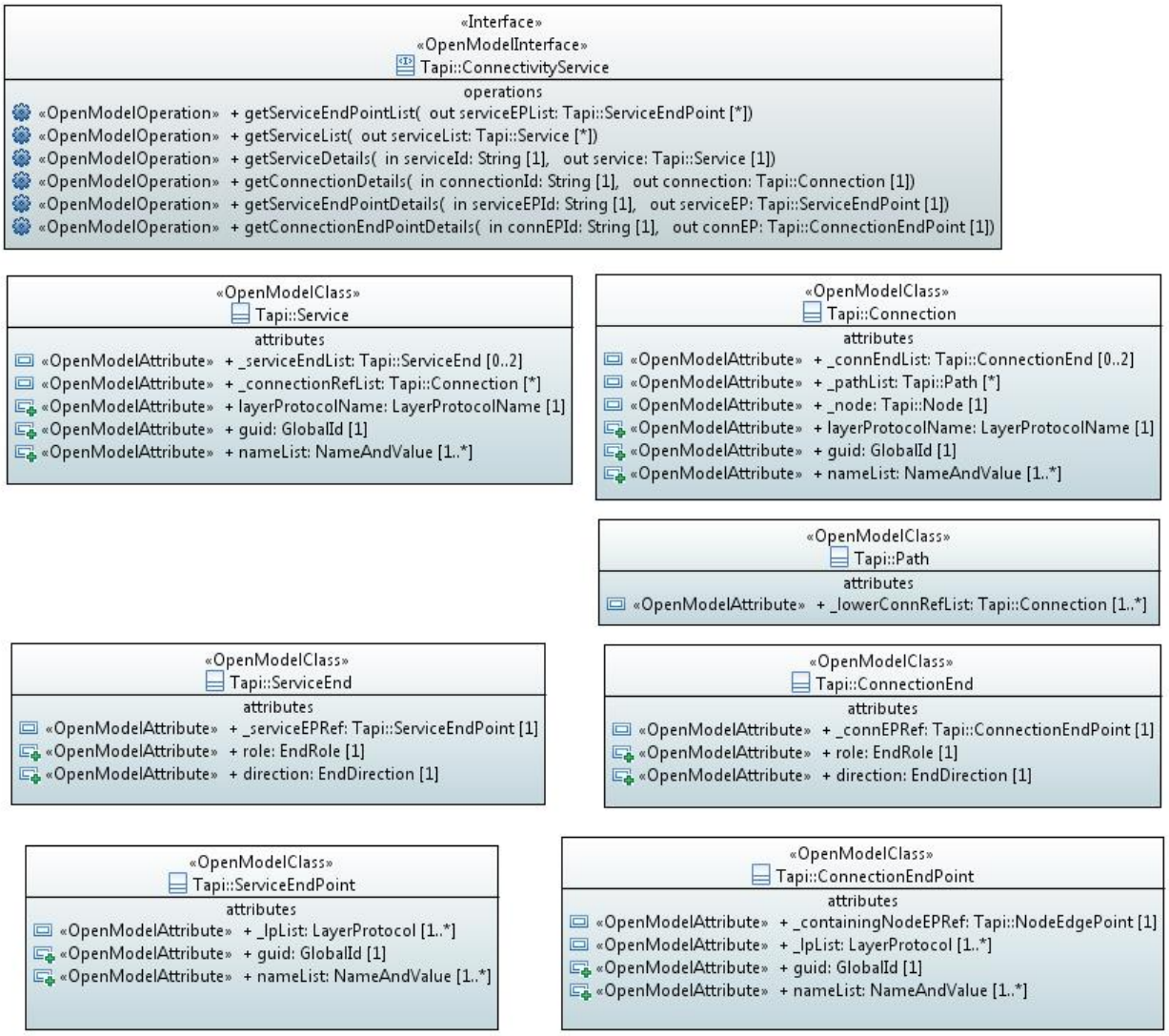

Figure 23: Connectivity Service API: IM Skeleton

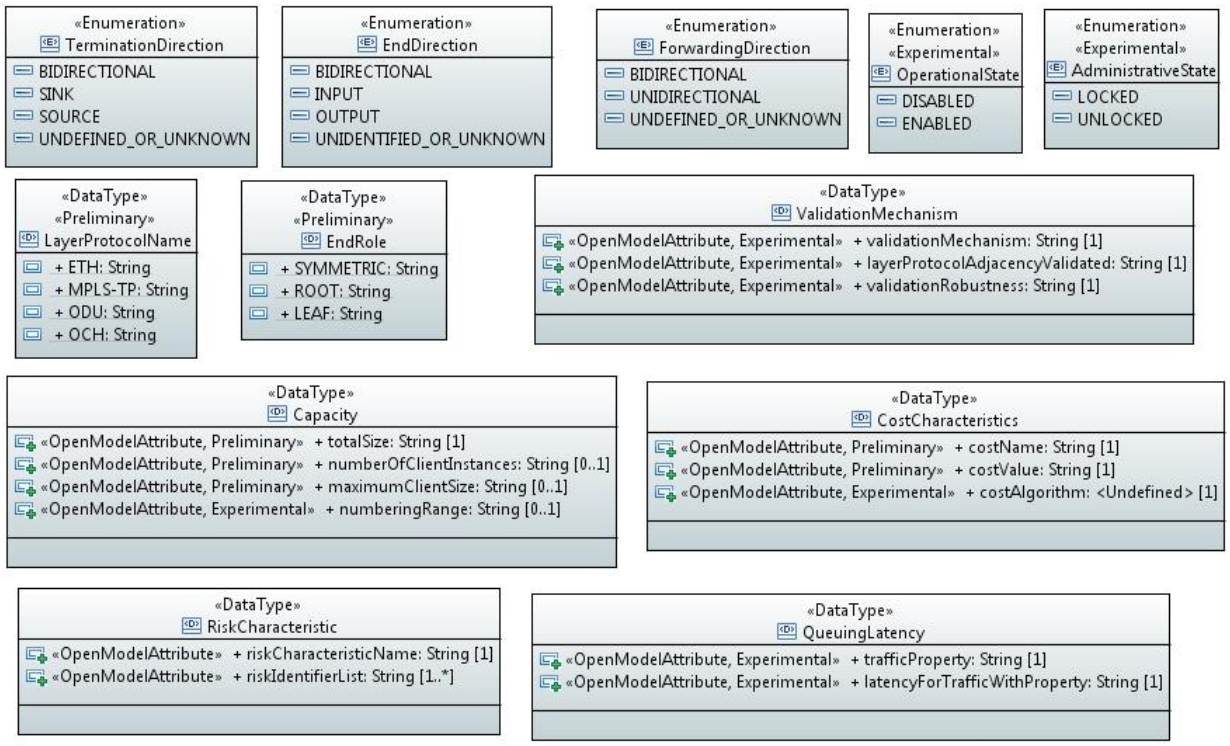

Figure 24: Transport API Data Types

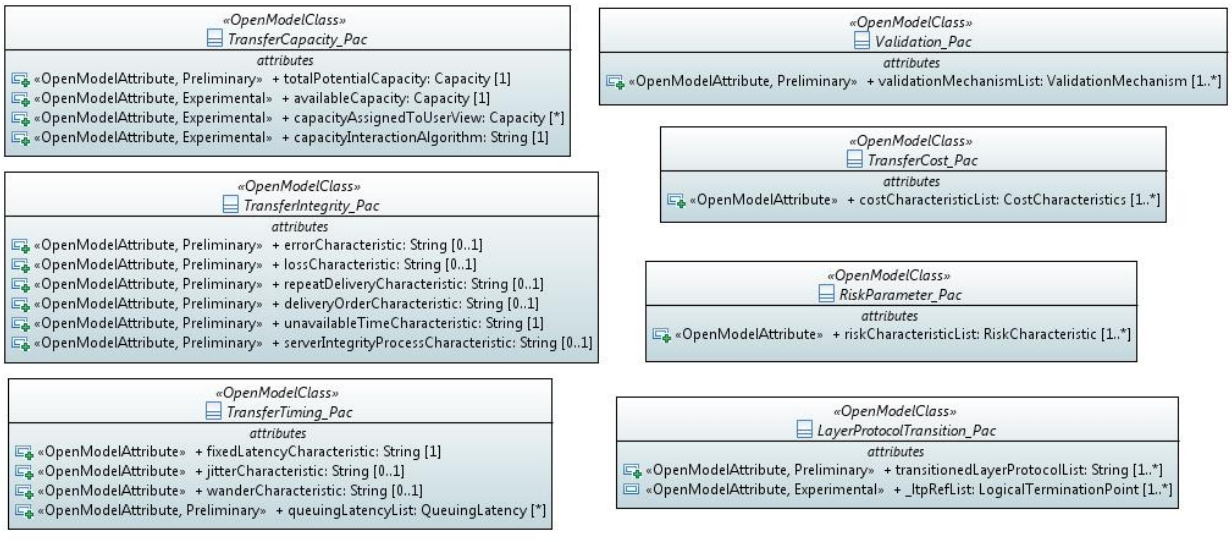

Figure 25: Transport API Topology Pacs

7 Contributors

The Transport API Design team responsible for the writing of this document included:

- Italo Busi, Huawei
- Jia He, Huawei
- Karthik Sethuraman, NEC (Editor)
- Lyndon Ong, Ciena
- Nigel Davis, Ciena
- Sergio Bellotti, ALU

Special thanks to Chen Qiaogang, Erez Segev, Eve Varma, Guoying Zhang, Hui Ding, Ricard Vilalta, Victor Lopez, and others for their input and comments.

8 Version History

ONF2015.087.xx xx=	DATE	VERSION COMMENTS
00	Jan 9, 2015	Initial Draft
01	Feb 10, 2015	Draft Topology, Service and Virtual Network Service requirements from Karthik, Lyndon, Jia
02	Mar 6, 2015	Updates to Service, Draft Connection and Path requirements from Lyndon, Sergio and Shinji
03	Mar 19,2015	Updates and comments to Path Computation sections from Sergio and Jia
04	April 08,2015	Added definitions and some cleanup (resolved comments and changes) by Karthik and Lyndon
05	April 23, 2015	Topology/Service section comments cleanup by Karthik and Path Computation section cleanup by Sergio
06	May 14, 2015	Updated Definitions and Topology sections to reflect the latest design team discussion outcomes and to use Core IM terminology
0.7	June 3, 2015	Merged comments for review at the Darmstadt F2F. Clarified NULL FD concepts
0.8	June 11, 2015	Post Darmstadt F2F version with cleanup and accepting comments and changes. Version liaised to OIF.

0.9	Aug 17,2015	By Karthik: - Added/merged comments from Chen Qiaogang, Hui Ding, Ricard Vilalta, Victor Lopez, Erez Segev, Eve Varma and others. - Added initial draft of appendices for Concepts, Use cases and IM. - Rewrote the Terms and Definitions section. - Reorganized the Requirements section and API headings - Updated the Topology section to use TAPI terminology and removed redundant APIs - Added the Connectivity Retrieval API requirements - Removed the Connection Control section for now – may be added back in future
0.10	Sep 28, 2015	By Karthik - Cleaned up the change bars and addressed comments from version 0.9 - Added additional text to the TAPI Functional Architecture section - Removed any direct dependency of the Service APIs on the Topology API - Renamed TAPI *View* to *Context* -