OPEN NETWORKING FOUNDATION

Core Information Model (CoreModel)

Version 1.0
March 30, 2015

ONF TR-512

OpenFlow™

ONF Document Type: Technical Recommendation
ONF Document Name: Core Information Model version 1.0

Disclaimer

THIS SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Any marks and brands contained herein are the property of their respective owners.

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

©2014 Open Networking Foundation. All rights reserved.

Open Networking Foundation, the ONF symbol, and OpenFlow are registered trademarks of the Open Networking Foundation, in the United States and/or in other countries. All other brands, products, or service names are or may be trademarks or service marks of, and are used to identify, products or services of their respective owners.

© 2015 Open Networking Foundation

Document History

Version	Date	Description of Change
1.0	March 30, 2015	Initial version of the base document of the “Core Information Model” fragment of the ONF Common Information Model (ONF-CIM).

1 Introduction

An information model describes the things in a domain in terms of objects, their properties (represented as attributes), and their relationships. This ONF Technical Recommendation (TR) describes the ONF Common Information Model (ONF-CIM). The ONF-CIM focuses on representation of data plane resources[1] for the purpose of management-control. In accordance with the SDN architecture **[ONF SDN Arch],** this management-control is expected to be achieved by an SDN controller. The controller expresses a view of the network, in terms of the things represented in the ONF-CIM, to each client SDN controllers/applications to meet the needs of that client.

This information model is divided into a number of pieces and is centered on a core fragment that is independent of specific data plane technology. The model includes pieces that provide data plane technology (forwarding technology) specific structures and properties (such as OTN, Ethernet, and MPLS-TP). These can be used to augment the core to provide a data plane technology specific representation. The model also includes application specific information models (e.g. applicable to representation of networking in a storage context). Most importantly the ONF-CIM is modeled independent of the ultimate protocols that may be used in the control interfaces.

The ONF TR document **[ONF CIM Guidelines]** specifies the principles and guidelines for the development and use of the ONF Common IM, including the guidelines for deriving purpose-specific information model views (through pruning and re-factoring on selected subsets of

[1] It is focused on representation of the functions/resources that have the primary purpose of supporting information forwarding (transfer and transform functions), that form a network that realizes virtual adjacency, for the purpose of control of those functions/resources. Those resources are referred to as data plane resources. The information model is not intended to cover resources that have a primary purpose of supporting storage or compute solutions.

© 2015 Open Networking Foundation

artifacts from the Common IM), and mapping into data schema for protocol-specific control interfaces.

The information model defined in this TR is expressed in a formal language called UML (Unified Modeling Language). UML has a number of basic model elements, called UML artifacts. In order to assure consistent modeling, only a selected subset of these UML artifacts is used in the development of the ONF Common IM. The selected subset of UML artifacts is documented in the ONF TR document **[ONF UML Guidelines]**.

The information model defined in this TR is developed using the Papyrus open source UML Tool. The ONF TR document **[ONF Papyrus Guidelines]** specifies the guidelines for using the Papyrus tool. This guidelines document also describes how the Common IM modeling teams can cooperate in the GitHub environment for separate and coordinated development of the ONF-CIM fragments.

Figure 1-1 below, reproduced from Figure 1.1 of **[ONF CIM Guidelines]**, provides an overview of the structure of the ONF Common IM and shows how the purpose and protocol specific interfaces may be derived from the Common IM.

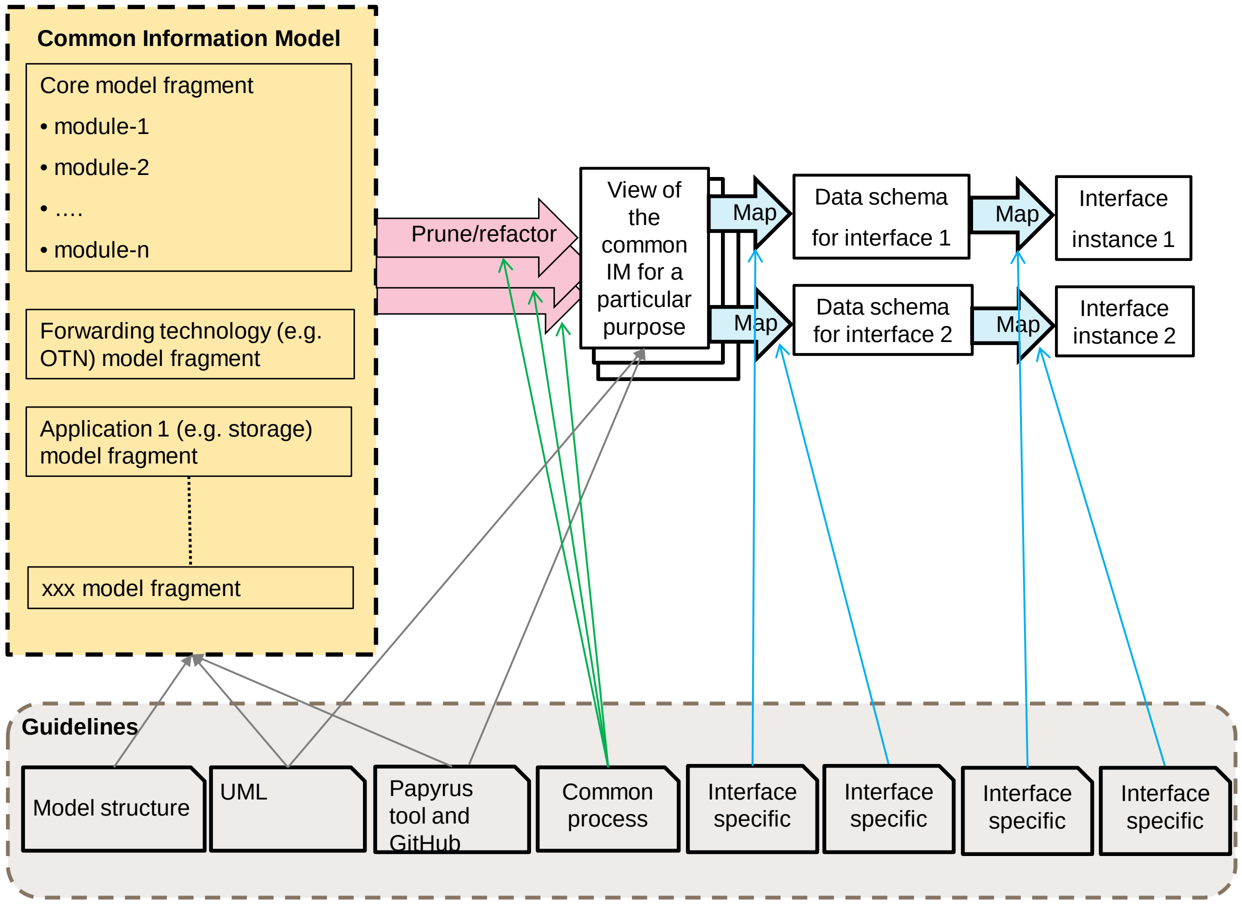


Figure 1-1 Methodology of IM and DS Development

In the development of the Core IM, information models from other SDOs have been taken as input as the bases, including [TMF TR225], [ITU-T G.7711] (ex. G.gim), [ITU-T G.874.1], [ITU-T G.8052], and [ITU-T G.8152].

2 References

[ONF SDN Arch]	ONF SDN Architecture 1.0, June 2014 (https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf)
[ONF CIM Guidelines]	ONF Common Information Model Overview and Guidelines (https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Common_Information_Model_V1.0.pdf)
[ONF UML Guidelines]	ONF UML Model Guidelines (https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/UML_Modeling_Guidelines_V1.0.pdf)
[ONF Papyrus Guidelines]	ONF Papyrus Guidelines (https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Papyrus_Guidelines_V1.0.pdf)
[TMF TR225]	TM Forum TR225, Logical Resource: Network Function Model
[TMF TR215]	TMF TR 215Logical Resource Network Model Advancements and Insights
[ITU-T G.7711]	Recommendation ITU-T G.7711/Y.1702 (ex. G.gim) (draft), *Generic Protocol-Neutral Information Model for Transport Resources*
[ITU-T G.874.1]	Recommendation ITU-T G.874.1 (10/2012), *Optical transport network: Protocol-neutral management information model for the network element view*, plus Amendment 1 (08/2013)
[ITU-T G.8052]	Recommendation ITU-T G.8052/Y.1346 (08/2013), *Protocol-neutral management information model for the Ethernet Transport capable network element*
[ITU-T G.8152]	Recommendation ITU-T G.8152/Y.1375 (draft), *Protocol-neutral management information model for the MPLS-TP network element*
[ITU-T G.852.2]	Recommendation ITU-T G.852.1, *Enterprise viewpoint description of transport network resource model*
[TMF 612]	TM Forum MTOSI
[ISO/IEC 19505]	ISO/IEC 19505:2012 Information technology -- Object Management Group Unified Modeling Language (OMG UML)
[ITU-T G.805]	Recommendation ITU-T G.805, *Generic functional architecture of transport networks*
[ITU-T M.3100]	Recommendation ITU-T M.3100, *Generic network information model*

ITU-T 808.1]	Recommendation ITU-T G.808.1, *Generic protection switching – Linear trail and subnetwork protection*

3 Definitions

3.1 Terms defined elsewhere

This document uses the terms defined elsewhere. These terms are highlighted in section 4 Abbreviations and acronyms below by referring to the definition source document.

3.2 Terms defined in this TR

The primary purpose of this document is to define terms and hence terms are defined throughout the document. Key terms are highlighted in 4 Abbreviations and acronyms below by referring to the section in this document where the term is defined.

4 Abbreviations and acronyms

This TR uses the following abbreviations and acronyms (Note that some cross references are included here rather than in the References section where the cross reference is only relevant for abbreviation/acronym interpretation purposes):

AP	Access Point [ITU-T G.805]
CP	Connection Point [ITU-T G.805]
CTP	Connection Termination Point. Note that definitions differ between TM Forum [TMF 612] and [ITU-T M.3100]. Both usages apply here when referring to legacy cases and the abbreviation is qualified in all cases of use.
ECC	Embedded Communications Channel [ITU-T G.874]
EMS	Element Management System [definition reference ITU-T M.3400 - TMN][2]
EP	EndPoint (see 6.1.5 EndPoint (EP) below)
ETH	Ethernet MAC Layer [definition reference ITU-T G.8011.2]
ETY	Ethernet Physical Layer [definition ITU-T G.8011.2]
FC	ForwardingConstruct (see 6.1.3 ForwardingConstruct (FC) below. Note that at this point the definition is subtly different to that in [TMF TR225]. The aim is to align the terms usage)
FDFr	FlowDomainFragment [TMF 612]
FRE	ForwardingRelationshipEncapsulation [TMF TR215]
FTP	FloatingTerminationPoint [TMF 612]

[2] This term is not intended for use other than in reference to legacy systems.

GitHub	See www.github.com
GUID	Globally Unique IDentifier (see www.wikipedia .org/Globally_unique_identifier)
IM	Information Model (see 1 Introduction above)
IMP	Inverse MultiPlexing [ITU-T G.805]
ISO	International Organization for Standardization (see www.iso.org)
ITU-T	International Telecommunications Union (see www.itu.int)
LP	LayerProtocol (see 6.1.1 LogicalTerminationPoint (LTP) and LayerProtocol (LP) below)
LTP	LogicalTerminationPoint (see 6.1.1 LogicalTerminationPoint (LTP) and LayerProtocol (LP) below)
MAC	Media Access Control
MEP	Maintenance End Point
MFDFr	MatrixFlowDomainFragment
MLSN	MultiLayerSubNetwork [TMF 612]
MP2MP	Multi-Point to Multi-Point
MPLS-TP	Multi-Protocol Label Switching Transport Profile [definition reference RFC6378]
NCD	NetworkControlDomain
NE	NetworkElement
OAM	Operations Administration and Maintenance
OCh	Optical Channel
ODU	Optical Data Unit
OMS	Optical Multiplex Section
ONF-CIM	ONF Common Information Model
OPS	Optical Protection Switch
OS	Operations System (essentially OSS - Operation Support System)
OTN	Optical Transport Network
OTS	Optical Transmission Section
OTU	Optical channel Transport Unit
P2MP	Point to Multi-Point
P2P	Point to Point
PON	Passive Optical Network
PTP	Physical Termination Point [TMF 612]
RMP	Rooted Multi-Point
SDN	Software Defined Networking [ONF]
SDO	Standards Development Organization
SNC	SubNetworkConnection [TMF 612]
TBD	To Be Defined

TCP	Termination Connection Point [ITU-T G.805]
TDM	Time Division Multiplex
TMF	TeleManagement Forum (see www.tmforum.org)
TP	Termination Point [ITU-T M.3100]
TPE	TerminationPointEncapsulation [TMF TR215]
TR	Technical Recommendation [ONF] Technical Report [TM Forum]
TTP	Trail Termination Point [ITU-T M.3100]
UML	Unified Modelling Language (see www.omg.org)
VNE	Virtual Network Element
XC	CrossConnection

5 Conventions

5.1 UML modeling conventions

The information model defined in this TR is expressed in a formal language called UML (Unified Modeling Language), which was developed by the Object Management Group (OMG). It is a general-purpose modeling language in the field of software engineering. In 2000 the Unified Modeling Language was also accepted by the International Organization for Standardization (ISO) as an approved ISO standard [ISO/IEC 19505].

UML defines a number of basic model elements, called UML artifacts. In order to assure consistent modeling, only a selected subset of these artifacts is used in the development of the ONF-CIM. The selected subset of UML artifacts is documented in [ONF UML Guidelines].

5.2 Lifecycle Stereotypes

Stereotypes are applied to entities in the model to indicate the degree of maturity[3]. These are made visible in many of the figures.

The following stereotypes appear in this document:

- «experimental»: Indicates that the entity is at a very early stage of development and will almost certainly change. The entity is NOT mature enough to be used in implementation.
- «preliminary»: Indicates that the entity is at a relatively early stage of development and is likely to change but is mature enough to be used in implementation.

[3] The whole model including all degrees of work in progress has been published to allow the user maximum opportunity to set a most consistent direction with the work at hand. It is considered important to expose work in progress especially where this may have an impact on a choice of implementation. There may be some experimental structure that contains some very stable parts, without that structure those parts might be quite uninterpretable. A user who decides to take a low risk approach can ignore preliminary and experimental parts. A user who is more inclined to take a risk or who is looking for inspiration for their work can take the experimental and preliminary parts, understanding the risk involved.

If no stereotype is shown the entity is mature. There are other lifecycle stereotypes that are not relevant in this document.

5.3 Diagram Keys

The document includes a number of UML diagrams. The UML symbol set is suitably explained in [ONF UML Guidelines].

The symbols highlighted below are used in this document in pictorial representations of the model.

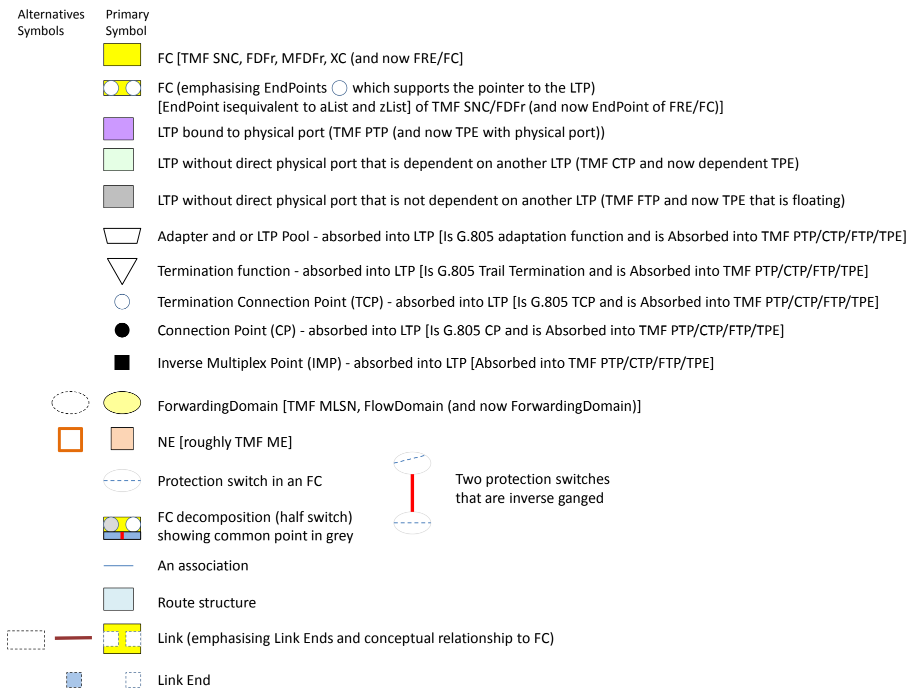

Alternatives Symbols
Primary Symbol
FC [TMF SNC, FDFr, MFDFr, XC (and now FRE/FC]
FC (emphasising EndPoints ◯ which supports the pointer to the LTP) [EndPoint isequivalent to aList and zList] of TMF SNC/FDFr (and now EndPoint of FRE/FC)]
LTP bound to physical port (TMF PTP (and now TPE with physical port))
LTP without direct physical port that is dependent on another LTP (TMF CTP and now dependent TPE)
LTP without direct physical port that is not dependent on another LTP (TMF FTP and now TPE that is floating)
Adapter and or LTP Pool - absorbed into LTP [Is G.805 adaptation function and is Absorbed into TMF PTP/CTP/FTP/TPE]
Termination function - absorbed into LTP [Is G.805 Trail Termination and is Absorbed into TMF PTP/CTP/FTP/TPE]
Termination Connection Point (TCP) - absorbed into LTP [Is G.805 TCP and is Absorbed into TMF PTP/CTP/FTP/TPE]
Connection Point (CP) - absorbed into LTP [Is G.805 CP and is Absorbed into TMF PTP/CTP/FTP/TPE]
Inverse Multiplex Point (IMP) - absorbed into LTP [Absorbed into TMF PTP/CTP/FTP/TPE]
ForwardingDomain [TMF MLSN, FlowDomain (and now ForwardingDomain)]
NE [roughly TMF ME]
Protection switch in an FC
Two protection switches that are inverse ganged
FC decomposition (half switch) showing common point in grey
An association
Route structure
Link (emphasising Link Ends and conceptual relationship to FC)
Link End

Figure 5-1 Illustrative Key[4]

The relationship between the entities in the ONF-CIM and other familiar models are shown in the next figure. The figure also provides a key to some further symbols.

[4] It should be noted that in this version and future versions the terms ForwardingDomain (FD) and ForwardingConstruct (FC) are used in place of SubNetworkConnection (SNC) and SubNetwork (SN) (respectively used in the earlier versions of the ONF Information Model).

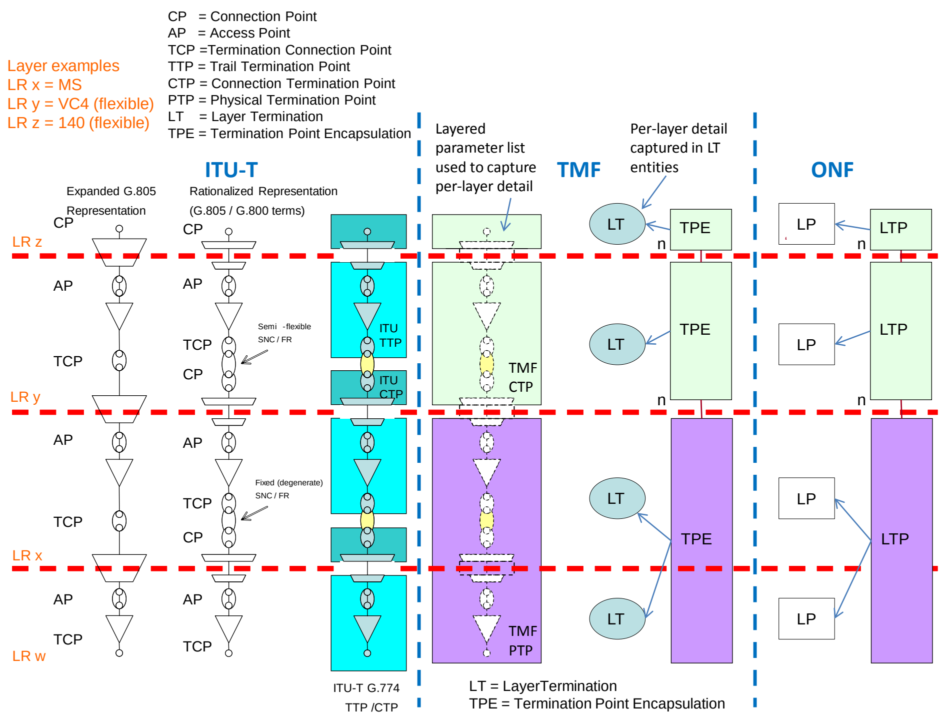

CP = Connection Point
AP = Access Point
TCP =Termination Connection Point
TTP = Trail Termination Point
CTP = Connection Termination Point
PTP = Physical Termination Point
LT = Layer Termination
TPE = Termination Point Encapsulation
Layer examples
LR x = MS
LR y = VC4 (flexible)
LR z = 140 (flexible)
ITU-T
Expanded G.805 Representation
Rationalized Representation (G.805 / G.800 terms)
Layered parameter list used to capture per-layer detail
TMF
Per-layer detail captured in LT entities
ONF
LR z
LR y
LR x
LR w
CP
AP
TCP
Semi -flexible SNC / FR
Fixed (degenerate) SNC / FR
ITU TTP
ITU CTP
TMF CTP
TMF PTP
LT
TPE
LP
LTP
n
ITU-T G.774 TTP /CTP
LT = LayerTermination
TPE = Termination Point Encapsulation

Figure 5-2 Mapping from ITU-T and TM Forum Termination models to the ONF Core[5]

6 Overview of the Core Model Fragment

The focus of this document is the Core Model Fragment of the ONF-CIM. The Core Model Fragment is divided into several UML packages.[6] The following UML packages are discussed in detail in this document:

- CoreNetworkModule: Covers the essentials for modeling of the Network providing an overview of the key classes.
- CoreFoundationModule: Covers aspects common to all classes such as identifiers and naming.

Some of the contents of the CoreModelEnhancements UML package are also discussed.

The CoreNetworkModule encompasses Topology, Termination and Forwarding aspects. These aspects are described in this document as follows:

- Termination Subset of the CoreNetworkModule: Covers the modeling of the processing of transport characteristic information, such as termination, adaptation, OAM, etc.

[5] It should be noted that in this version and future versions the terms ForwardingDomain (FD) and ForwardingConstruct (FC) are used in place of SubNetworkConnection (SNC) and SubNetwork (SN) (respectively used in the earlier versions of the ONF Information Model).

[6] Some UML packages are used to hold model fragments, some to hold modules and other subsets of a module or fragment. The UML packages are labelled accordingly.

  - Note that technology specific details are covered in the ForwardingTechnologyModelFragments of the ONF-CIM (this aspect is not in the scope of this document)
- Forwarding Subest of the CoreNetworkModule: Covers the details of forwarding entities, including:
  - The Basic Forwarding
  - The ForwardingConstructSpec
- Topology Subset of the CoreNetworkModule: Covers the modeling of network topology information in detail[7] and describes the attributes relevant when working with network topology.

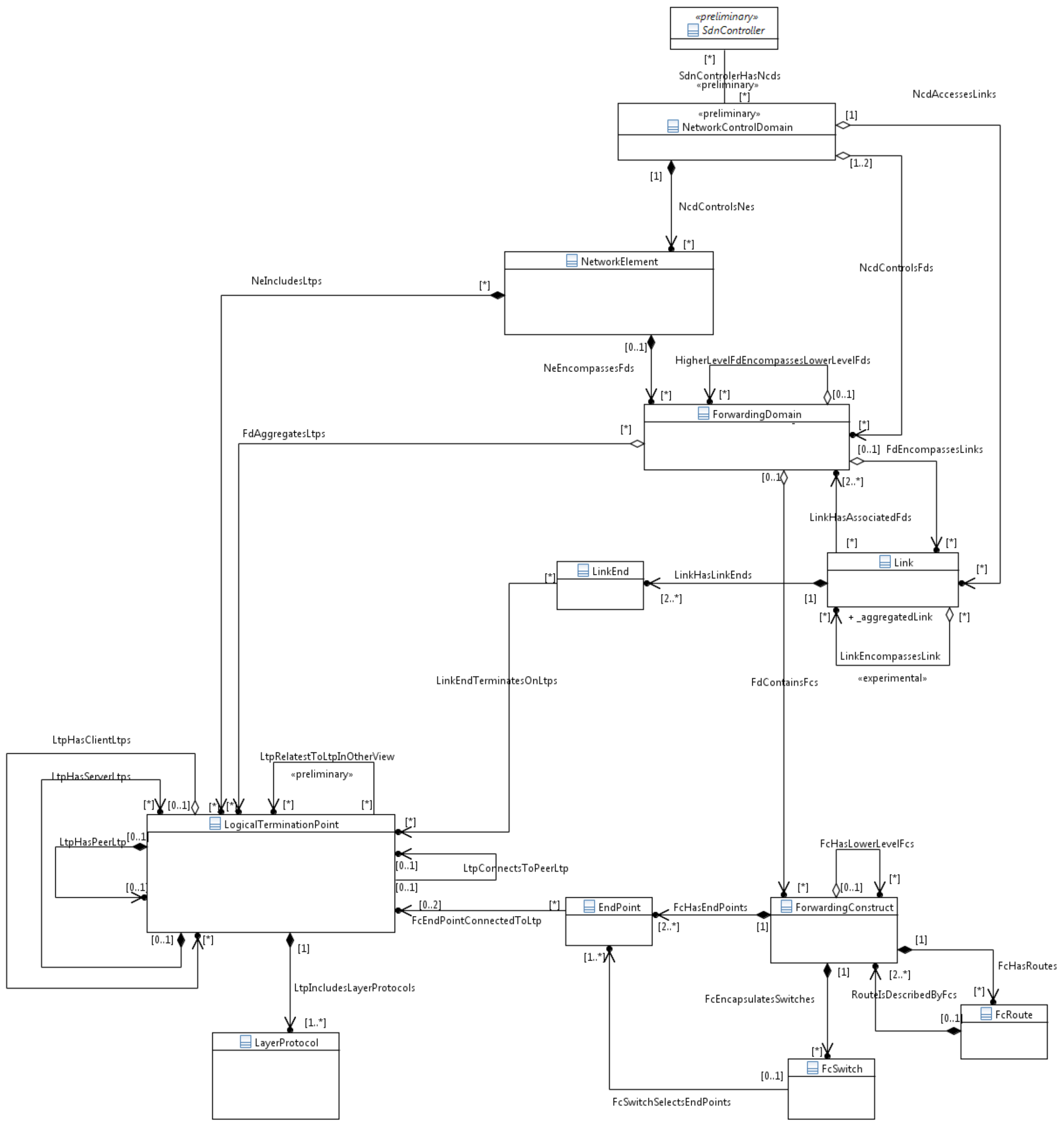
In all sections Stylized Network Element and network views are provided to illustrate the model application.

This clause provides visual views, using the UML class diagrams for the object classes of the information model. These diagrams depict a subset of the relationships among the object classes, such as inheritance (i.e. specialization), association relationships (such as aggregation and composition), and conditional features or capabilities. Some diagrams also show further details of the individual object classes, such as the attributes of the object classes, and the data types used by the attributes. In addition to the class diagrams there are also pictorial representations of stylized network fragments to assist in the understanding of the model.

6.1 Overview of the CoreNetworkModule of the CoreModel

This clause provides a high-level overview of the generic information model. Figure 6-1 below is a skeleton class diagram illustrating the key object classes defined in the CoreNetworkModule of the CoreModel.

[7] The information described in this subset can be used for example for path computation and to provide views of network capacity/capability with information maintained in a topology database.



HighLevelSkeletonOverview.PNG

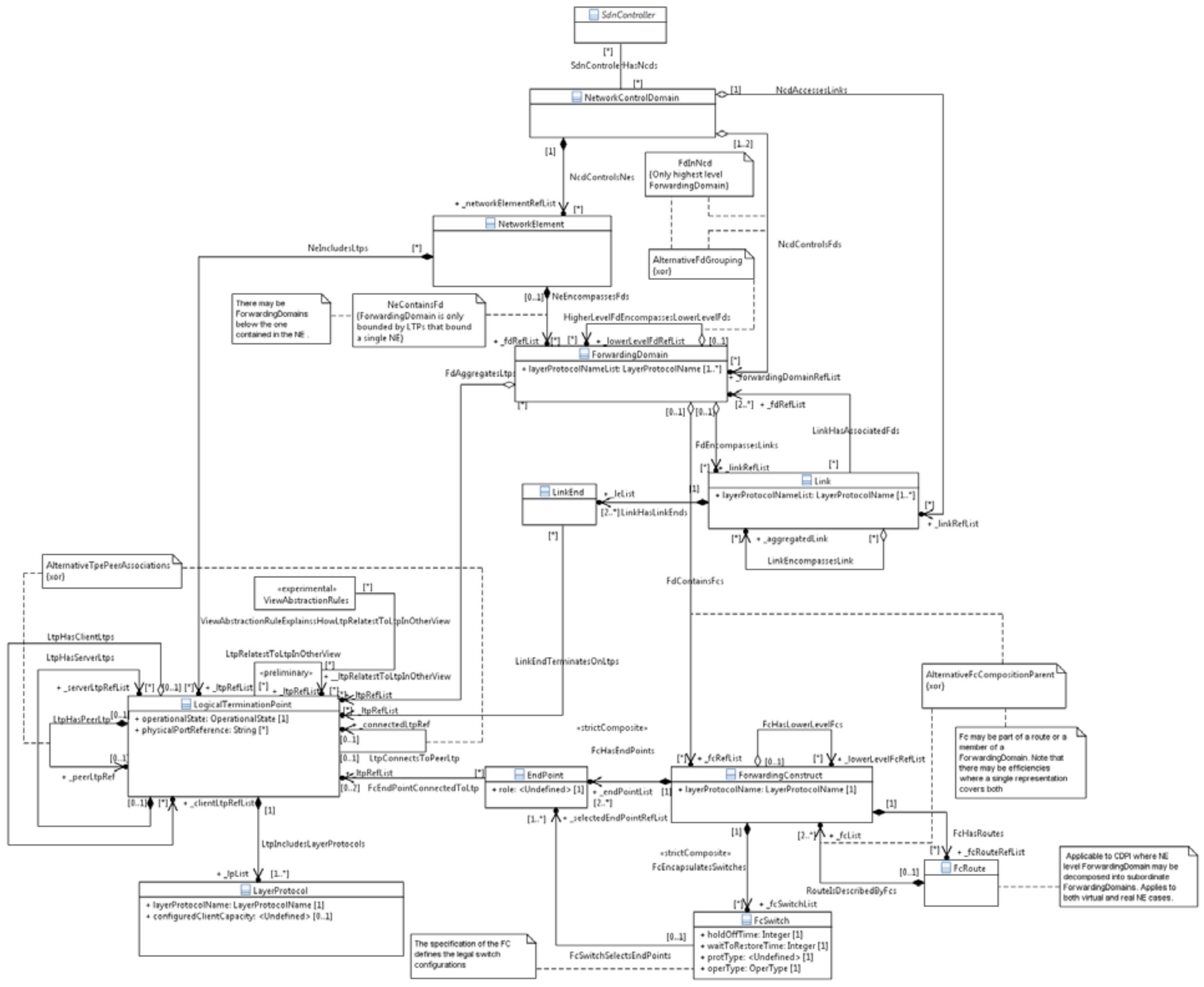
Figure 6-1 Skeleton Class Diagram of key object classes

Note: A more convenient way to view the details of the diagram is by using the companion PNG file, which can be independently zoomed in and out without impacting the viewing of the main document.

© 2015 Open Networking Foundation

When applying the information model to a specific-purpose interface, not all of the information model artifacts need to be considered. That is, only a subset of the overall information model may be needed. Depending on the scope of the interface, pruning of the information model may be necessary, such as excluding a whole object class or part of an object class. For example, in the interface from an SDN controller directly to a Network Element in the infrastructure layer, the NetworkControlDomain object class is beyond the scope of the Network Element[8] interface and thus not needed. In addition, re-factoring of the selected model artifacts may be necessary to meet the specific-purpose needs. However, re-factoring of the model artifacts should not add semantics beyond those defined in the information model.

Figure 6-2 below is a more comprehensive class diagram than the previous one showing some attributes and constraints of the associations in the mode.


SdnController
SdnControllerHasNcds
NetworkControlDomain
NcdAccessesLinks
NcdControlsNes
NcdControlsFds
FdInNcd {Only highest level ForwardingDomain}
NetworkElement
NeIncludesLtps
AlternativeFdGrouping {xor}
NeEncompassesFds
NeContainsFd {ForwardingDomain is only bounded by LTPs that bound a single NE}
There may be ForwardingDomains below the one contained in the NE.
HigherLevelFdEncompassesLowerLevelFds
ForwardingDomain
+ layerProtocolNameList: LayerProtocolName [1..*]
FdAggregatesLtps
LinkHasAssociatedFds
FdEncompassesLinks
LinkEnd
LinkHasLinkEnds
Link
+ layerProtocolNameList: LayerProtocolName [1..*]
LinkEncompassesLink
FdContainsFcs
AlternativeTpePeerAssociations {xor}
«experimental» ViewAbstractionRules
ViewAbstractionRuleExplainssHowLtpRelatestToLtpInOtherView
LtpRelatestToLtpInOtherView
«preliminary»
LinkEndTerminatesOnLtps
LtpHasClientLtps
LtpHasServerLtps
LtpHasPeerLtp
LogicalTerminationPoint
+ operationalState: OperationalState [1]
+ physicalPortReference: String [*]
LtpConnectsToPeerLtp
FcEndPointConnectedToLtp
«strictComposite»
FcHasEndPoints
EndPoint
+ role: <Undefined> [1]
FcHasLowerLevelFcs
ForwardingConstruct
+ layerProtocolName: LayerProtocolName [1]
AlternativeFcCompositionParent {xor}
Fc may be part of a route or a member of a ForwardingDomain. Note that there may be efficiencies where a single representation covers both
FcHasRoutes
FcRoute
RouteIsDescribedByFcs
Applicable to CDPI where NE level ForwardingDomain may be decomposed into subordinate ForwardingDomains. Applies to both virtual and real NE cases.
LtpIncludesLayerProtocols
LayerProtocol
+ layerProtocolName: LayerProtocolName [1]
+ configuredClientCapacity: <Undefined> [0..1]
«strictComposite»
FcEncapsulatesSwitches
FcSwitch
+ holdOffTime: Integer [1]
+ waitToRestoreTime: Integer [1]
+ protType: <Undefined> [1]
+ operType: OperType [1]
FcSwitchSelectsEndPoints
The specification of the FC defines the legal switch configurations

[8] The Network Element scope of the direct interface from a SDN controller to an Network Element in the infrastructure layer is similar to the EMS-to-NE management interface defined in the information models [ITU-T G.874.1] (OTN), [ITU-T G.8052] (Ethernet), and draft [ITU-T G.8152] (MPLS-TP). On the other hand, the network scope of the interface between two SDN controllers is similar to the OS-to-OS management interface defined in the TMF 612 information model.

HighLevelDetail.PNG

Figure 6-2 Class Diagram of all key object classes showing attributes and constraints

6.1.1 LogicalTerminationPoint (LTP) and LayerProtocol (LP)

The LogicalTerminationPoint (LTP) object class encapsulates the termination, adaptation and OAM functions of one or more transport layers. The structure of LTP supports all transport protocols including circuit and packet forms. Each transport layer is represented by a LayerProtocol (LP) instance. The LayerProtocol instances of the LTP can be used for controlling termination and OAM functionality of that layer. It can also be used for controlling the adaptation (i.e. encapsulation and/or multiplexing of client signal). 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 must be split over separate instances of LTP. Rules for forming LTP instances are provided in section 6.3 Termination Subset on page 24.

The LP object class is defined with generic attributes "layerProtocolName" for indicating the supported transport layer protocol. LayerProtocols include:

- Layer 0 (L0): OPS, OTS, OMS, OCh
- Layer 1 (L1): OTU, ODU
- Layer 2 (L2): Carrier Grade Ethernet (ETY, ETH), MPLS-TP (MT)
- Transport layer specific properties (such as layer-specific termination and adaptation properties) are modeled as attributes of conditional packages (called "_Pacs" in the UML notation of the ONF-CIM) associated with the LP object class.[9]

Functions that can be associated/disassociated to/from an LTP instance (and ForwardingConstruct), such as OAM, protection switching, and performance monitoring are modeled as secondary object classes.

6.1.2 ForwardingDomain (FD)

The ForwardingDomain (FD) object class models the topological component which represents the opportunity to enable forwarding between points represented by the LTP in the model. The LTPs available are those defined at the boundary of the FD (see "subnetwork" topological component in G.852.2 and TMF 612). The FD object can hold zero or more instances of ForwardingConstruct (FC) of one or more layer networks; e.g., OCh, ODU, ETH, and MPLS. The FD object provides the context for instructing the formation, adjustment and removal of FCs.

The FD object class supports a recursive aggregation relationship (HigherLevelFdEncompassesLowerLevelFds) such that the internal construction of an FD can be

[9] Note that some implementation languages may allow the addition and removal of _Pacs and attributes from instances of a running system and others may be restricted such that _Pacs/attributes cannot be added/removed once an instance has been created. The model supports both modes of operation.

exposed as multiple lower level FDs and associated Links (partitioning)[10]: see Figure 6-14 in clause 6.5.1. An FD within a NetworkElement may represent a switch matrix/fabric[11]. Note that a Network Element can encompass multiple switch matrixes (represented by FDs): see Figure 6-15 in clause 6.5.1. An instance of FD is associated with zero or more LTP objects via the FdAggregatesLtps aggregation.

6.1.3 ForwardingConstruct (FC)

The ForwardingConstruct (FC) object class <<< is used to effect forwarding of transport characteristic (layer protocol) information and offers the potential to enable forwarding>>> models enabled potential for forwarding between two or more LTPs, and like the LTP, supports any transport protocol including all circuit and packet forms. The association of the FC to LTPs is made via EndPoints (essentially the ports of the FC) where each EndPoint (EP) of the FC has a role in the context of the FC. The traffic forwarding between the associated EPs of the FC depends upon the type of FC and may be associated with FcSwitch object instances.

An FC can be in only one FD. An FC object supports a recursive aggregation relationship such that the internal construction of an FC can be exposed as multiple lower level FC objects (partitioning)[12]. An FC object can have zero or more routes, each of which is defined as a list of lower level FC objects (with the implicit understanding that link connections are interleaved between the lower level FCs).[13] At the lowest level of recursion, a FC represents a cross-connection within a Network Element.[14] Thus the route of an FC may represent the cross-connections in a Network Element.

If an FC provides protection, the FC will have one or more associated FcSwitch objects as described in 6.1.6 FcSwitch on page 17 that have protection configuration related attributes.

The FC object 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.

6.1.4 FcRoute

The FcRoute object class models the individual routes of an FC. The route of an FC object is represented by a list of FCs at a lower level. Note that depending on the service supported by an FC, the FC can have multiple routes.

6.1.5 EndPoint (EP)

The EndPoint (EP) object class models the access to the FC function. Each EP instance has a role (e.g., working, protection, protected, hub, spoke, leaf, root, etc.) with respect to the FC function.

[10] The model actually represents aggregation of lower level FDs into higher level FDs as views rather than FD partition, and supports multiple views. This allow reallocation of capacity from lower level FDs to different higher level FDs as if the network is reorganized (as the association is aggregation not composition).
[11] There are cases where the matrix is itself decomposed so the FD may be smaller than the scope of the matrix.
[12] The LinkConnections associated with the Links that are exposed as part of the internal structure of the FD are not modeled at this point.
[13] The route is an alternative view of the internal structure of the FC. There are cases where a route is the most appropriate representation and cases where the aggregation is the most appropriate form.
[14] Recognizing that as it may be necessary to decompose the matrix into smaller FDs it may also be necessary to decompose the XC into smaller FCs (this is true at a control domain boundary for example).

The EP replaces the Protection Unit of a traditional protection model. It represents a protected (resilient/reliable) point or a protecting (unreliable working or protection) point.

An EP may be associated with a LTP.

6.1.6 FcSwitch

The FcSwitch object class models the switched forwarding of traffic (traffic flow) between EPs and is present where there is protection functionality in the FC. It plays the role of an aspect of the Protection Group of the traditional information model. When supporting the protection function, an FcSwitch object instance associates two or more EPs, each of which is playing the role of a Protection Unit.

It is possible for one or more protection EPs (standby/backup) to provide protection for one or more working (i.e., regular/main/preferred) EPs where either the protection or working EPs can feed one or more protected EPs. The protection system may operate in revertive or non-revertive (symmetric) mode. The waitToRestore attribute defines the revertive timer for the revertive mode. The protection function of the FcSwitch may operate in one of several modes including source switched, destination switched, source and destination switched, etc. (covering cases such as 1+1 and 1:1). It may be lockout (prevented from switching), force switched or manual switched. It will indicate switch state and notify change of state.

A specific instance of FC may not contain any FcSwitch instances when there is no protection capability, and may contain many FcSwitch intances in complex cases. The arrangement of switches for a particular instance is described by a referenced FcSpec[15] (see 6.4.2 Forwarding Construct Specification and other details of Forwarding on page 27). The approach supports all forms of protection described in [ITU-T 808.1].

6.1.7 Link and LinkEnd

The Link object class models effective adjacency between two or more[16] ForwardingDomains (FD).[17] In its basic form (i.e., point-to-point Link) it associates a set of LTP clients on one FD with an equivalent set of LTP clients on another FD. Like the FC, the Link has endpoints (LinkEnd) which take roles relevant to the constraints on flows offered by the Link (e.g., Root role or leaf role for a Link that has a constrained Tree configuration). A Link may offer parameters such as capacity and delay (see section 6.5.3 Detailed properties of Topology on page 38). These parameters depend on the type of technology that supports the link. An FD may aggregate Links: see Figure 6-14 in clause 6.5.1.The FdEncompassesLinks association is modeled to collect links that are wholly within the bounds of the FD.[18] [19]

[15] Many instances of FC may reference the same FcSpec.
[16] At this point the model supports point to point links fully. The model allows multi-point but anything above 2 is essentially (i.e., 3..*) is preliminary
.[17] The model supports an experimental attribute, offNetworkAddress, in the LinkEnd to cover cases where the FD that the Link ends on is outside the visibility (and hence off network).
[18] This association can also be inferred from the higherLevelFdEncompassesLowerLevelFd association together with the linkHasAssociatedFds association. Note that Link decomposition can also be represented using the LinkEncompassesLink association (similar to the FdEncompassesFd usage for the ForwardingDomain (this association is experimental).
[19] A Link with an Off-network end cannot be encompassed by an FD.

The Link can support multiple transport layers via the associated LTP object. Instance of Link can be formed with the necessary properties according to the degree of virtualization. For implementation optimization, where appropriate, multiple layer-specific links can be merged and represented as a single Link instance as the Link can represent a list of layer protocols (identified via the layerProtocolNameList attribute).

6.1.8 NetworkElement

The NetworkElement object class represents a Network Element[20] in the data plane or a virtual network element visible in the interface where virtualization is needed.

In the direct interface from an SDN controller to a Network Element in the data plane, the NetworkElement object defines the scope of control for the resources within the network element, e.g., internal transfer of user information between the external terminations (ports), encapsulation, multiplexing/demultiplexing, and OAM functions, etc. The NetworkElement provides the scope of the naming space for identifying objects representing the resources within the Network Element.

The NeEncompassesFd association occurs for FDs that are within the bounds of the NetworkElement definition such that the FD is bounded by LTPs, all of which are on the boundary of the NetworkElement or are within the NetworkElement.[21]

Where virtualization is employed, theNetworkElementobject represents a VirtualNetwork Element(VNE). The mapping of the VNE to the Network Elements is the internal matter of the SDN controller that offers the view of the VNE. Via the CPI interface between hierarchical SDN controllers,NetworkElementinstances can be created (or deleted) for providing (or removing) virtual views of the combination of slices of network elements in the data plane.

6.1.9 NetworkControlDomain (NCD)

The NetworkControlDomain (NCD) object class represents the scope of control that a particular SDN controller has with respect to a particular network, i.e., encompassing a designated set of interconnected (virtual) network elements.

In the interfaces between SDN controllers where virtualization is necessary, e.g., in client/server SDN controller relationship, the NCD object defines the scope of control of the client SDN controller on the virtual network that has been provided by the server SDN controller (i.e., the scope of control relates to the partitioned provider resources allocated to that particular client). The NCD provides the scope of naming space for identifying objects representing the virtual resources within the virtual network.

[20] The Network Element concept is well known in the industry and it is normal practice to represent it as in this model. However there would appear to be a number of potential issues with this traditional representation. These potential issues will be explored in a future release and there may be changes made to this entity. Because of the familiarity it has NOT been marked preliminary.

[21] Where an FD is referenced by the NeEncompassesFd association, any FDs that it encompasses (i.e., that are associated with it by HigherLevelFdEncompassesLowerLevelFds), must also be encompassed by the NE and hence must have the NeEncompassesFd association.

6.2 CoreFoundationModule

To communicate about a thing it is important to have some way of referring to that thing, i.e., to have some reference. Terms such as name and identifier are often used when describing the reference. Unfortunately these terms in general usage have ambiguity in their definition that leads to erroneous system behavior. With the aim to ensure that the controller system behavior is not erroneous, the model will adopt the following (hopefully suitably rigorous) principal definitions:

- Entity: A thing with an identity, defined boundary, properties, functionality and life.
  - Examples: A circuit pack, an LTP
- Feature of an Entity: A thing that is an inseparable part of an entity but is a distinct surface characteristic of the entity.
  - Examples: A pin on an integrated circuit, the endpoint on a FC, a face of a cube, the handle of a cup.
  - Note that this is important from a modeling perspective as the representation appears similar to that of an Entity
- Object Class: The representation of a thing that may be an entity or an inseparable "Feature of an Entity".
- Role: A specific structure of responsibilities, knowledge, skills, and attitudes in the context of some activity or greater structure. The role has an identity and identifier.
- Identifier: A property of an entity/role with a value that is unique within an identifier space, where the identifier space is itself globally unique, and immutable. An identifier carries no semantics with respect to the purpose of the entity.
- Globally Unique Identifier (GUID): An identifier that is globally unique.
- Local ID: An identifier that is unique in the context of some scope that is less than the global scope.
- Name: A property of an entity with a value that is unique in some namespace but may change during the life of the entity. A name carries no semantics with respect to the purpose of the entity.
- Label: A property of an entity with a value that is not expected to be unique and is allowed to change. A label carries no semantics with respect to the purpose of the entity and has no effect on the entity behavior or state.
  - A label can be used to carry a freeform text string for any operator purpose. The contents of a label in one view may happen to be the value of a name or identifier in another view. From the perspective of the view with the label there is no expectation other than the value is a string.
- Address: A structure of named values[22] in some address space that defines a location (a volume in that address space) where the structure is a nested hierarchy.
  - A named value may be a name or identifier, the name of the value may be a name or identifier
- Route: the way (via specified intermediate locations and paths) to get to one location from another.
- Property: A quality associated with a thing, structure or location.

[22] A named value is simply a tuple with two terms, one being a value and the other being the name of that value. For example in a street address a value may be "London" and the name of that value would be "City".

- Semantics: Meaning.
- Reference: Data in a communication between two applications that allows a shared understanding of the individual things.
    - This could be an identifier (including a GUID), a name, an address, or a route, depending upon the needs

Note:

- An entity may be known to be at a place in some functional or physical structure.
- A role may be known to be at a place in some process or behavioral structure.

Figure 6-3 below illustrates the naming/identifier-related attributes defined in the ONF-CIM. They are Global Unique ID (GUID), Local ID, Name and Label.

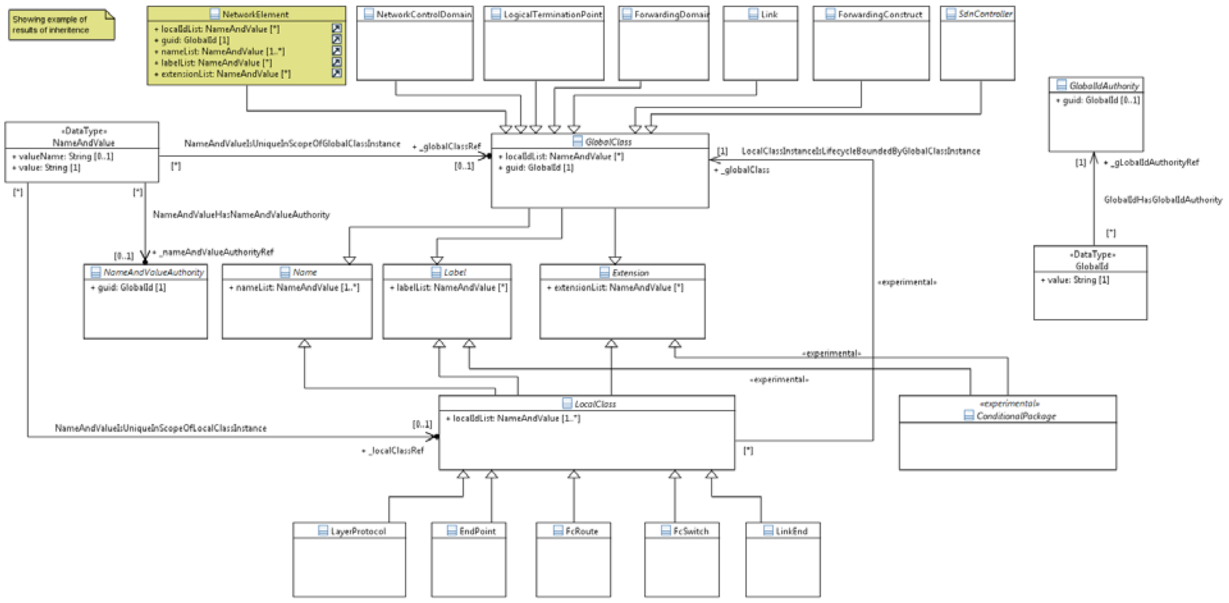
The model includes two abstract classes that provide names and identifiers, the GlobalClass and the LocalClass.[23] A GlobalClass represents a type of thing that has instances which can exist in their own right (independently of any others). A LocalClass represents a type of thing that is inseparable from a GlobalClass, but that is a distinct feature of that GlobalClass such that the instances of LocalClass are able to have associations with other instances. The mandatory LocalId of the LocalClass instance is unique in the context of the GlobalClass instance, from which it is inseparable.

The model also includes Extension which is not related to naming/identification. Extension provides an opportunity to define properties not declared in the class that extend the class enabling a realization with simple ad-hoc extension of standard classes to be conformant.

Note that the GUID is applicable only to global type object classes (i.e., subclass of GlobalClass) that their instances can exist on their own right, i.e., NCD, NetworkElement, LTP, FD, Link, FC, and SdnController. The other naming/identifier-related attributes are applicable to both global type object classes and local type object classes (i.e., subclass of LocalClass).[24]

[23] The model also provides ConditionalPackage to supply names and identifiers to _Pac classes but this is currently experimental.

[24] The intention is that only classes from the Core Model are shown in the figure. The classes shown are essentially illustrative. There is another figure in the model that captures Core Model inheritance in detail. All classes from all fragments should inherit from GlobalClass, LocalClass or ConditionalPackage. There is no issue with model dependency as the inheritance association is maintained with the class that is inheriting properties. Although not mandatory, it would seem advisable to maintain a figure per fragment that shows all classes from that fragment and their inheritance.


Showing example of results of inheritence
NetworkElement
+ localIdList: NameAndValue [*]
+ guid: GlobalId [1]
+ nameList: NameAndValue [1..*]
+ labelList: NameAndValue [*]
+ extensionList: NameAndValue [*]
NetworkControlDomain
LogicalTerminationPoint
ForwardingDomain
Link
ForwardingConstruct
SdnController
GlobalIdAuthority
+ guid: GlobalId [0..1]
«DataType» NameAndValue
+ valueName: String [0..1]
+ value: String [1]
NameAndValueIsUniqueInScopeOfGlobalClassInstance
+ _globalClassRef
[*]
[0..1]
GlobalClass
+ localIdList: NameAndValue [*]
+ guid: GlobalId [1]
[1]
LocalClassInstanceIsLifecycleBoundedByGlobalClassInstance
+ _globalClass
[1]
+ _gLobalIdAuthorityRef
GlobalIdHasGlobalIdAuthority
[*]
NameAndValueHasNameAndValueAuthority
[0..1]
+ _nameAndValueAuthorityRef
NameAndValueAuthority
+ guid: GlobalId [1]
Name
+ nameList: NameAndValue [1..*]
Label
+ labelList: NameAndValue [*]
Extension
+ extensionList: NameAndValue [*]
«experimental»
«DataType» GlobalId
+ value: String [1]
«experimental»
«experimental»
LocalClass
+ localIdList: NameAndValue [1..*]
«experimental» ConditionalPackage
NameAndValueIsUniqueInScopeOfLocalClassInstance
[0..1]
+ _localClassRef
[*]
LayerProtocol
EndPoint
FcRoute
FcSwitch
LinkEnd

CommonPackages-NoNotes.PNG

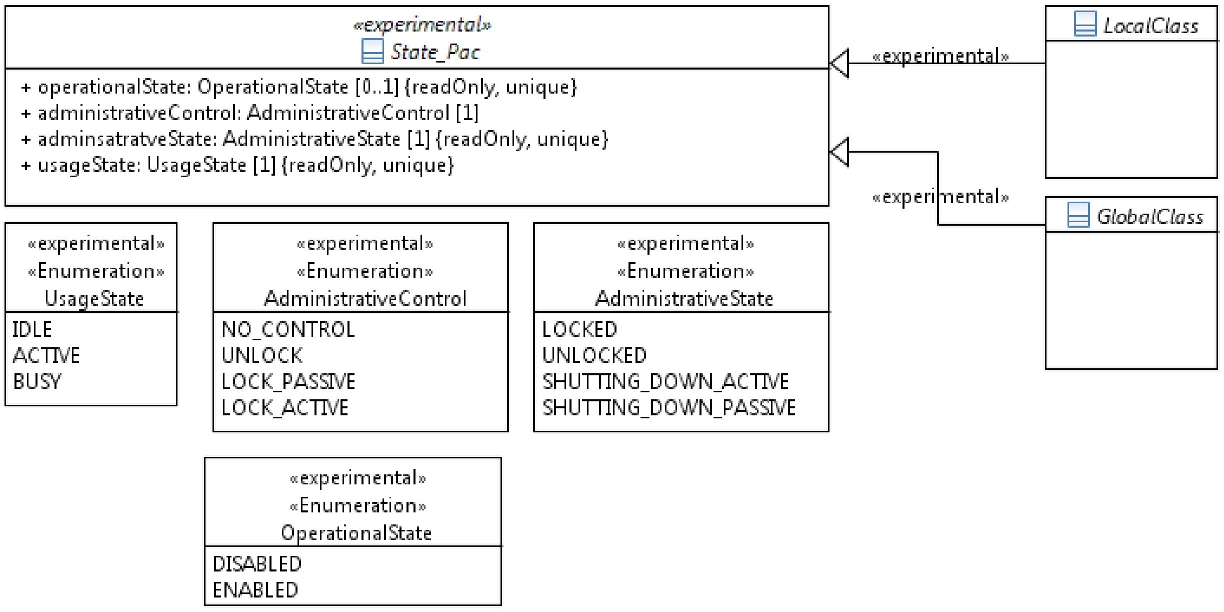
Figure 6-3 Class Diagram for Naming and Identifier of Objects

The Core Foundation module also defines a State_Pac artifact, which is a package of state attributes. The work on states is experimental at this stage (it is derived from ITU-T X.731). The State_Pac is inherited by GlobalClass and LocalClass object classes. The State_Pac consists of the following state-related attributes:

- Operational State:
    - Indicates the operability of the entity.
    - Read-only with values:
        - DISABLED: The entity is totally inoperable and unable to provide service to its users(s)
        - ENABLED: The entity is partially or fully operable and available for use.
- AdministrativeControl (not derived from X.731):
    - Reflects the current control action when the entity is not in the desired state.
    - Read/Write with values:
        - NO_CONTROL: There is no current control action active as the entity is in the desired state.

        - UNLOCK: The intention is for the entity to become unlocked and the entity is not UNLOCKED.
        - LOCK_PASSIVE: The intention is for the entity to become locked but no effort is expected to move to the Locked state (the state will be achieved once all users stop using the resource). The entity is not LOCKED.
        - LOCK_ACTIVE: The intention is for the entity to become locked and it is expected that an effort will be made to move to the Locked state (users will be actively removed). The entity is not LOCKED.

- Administrative State (derived from X.731 and extended):
    - Indicates the degree to which the capabilities of the entity are allowed for use.
    - Read-only with values:
        - LOCKED: The entity is administratively prohibited from performing services for its users.
        - UNLOCKED: The entity is administratively permitted to perform services for its users. This is independent of its inherent operability.
        - SHUTTING_DOWN_PASSIVE: The entity is administratively restricted to existing instances of use only. There may be no new instances of use enabled. This corresponds to a control of LOCK_PASSIVE.
        - SHUTTING_DOWN_ACTIVE: The entity is administratively restricted to existing instances of use only. There are specific actions to remove existing uses. There may be no new instances of use enabled. This corresponds to a control of LOCK_ACTIVE.
- Usage State:
    - Indicates the degree to which the entity is used.
    - Read-only with values:
        - IDLE: The entity is not currently in use.
        - ACTIVE: The entity is in use and has sufficient spare operating capacity to provide for additional simultaneous uses.
        - BUSY: The entity is in use but has no spare operating capacity to provide for any further simultaneous uses.



State.PNG

Figure 6-4 States for all Objects

6.3 Termination Subset

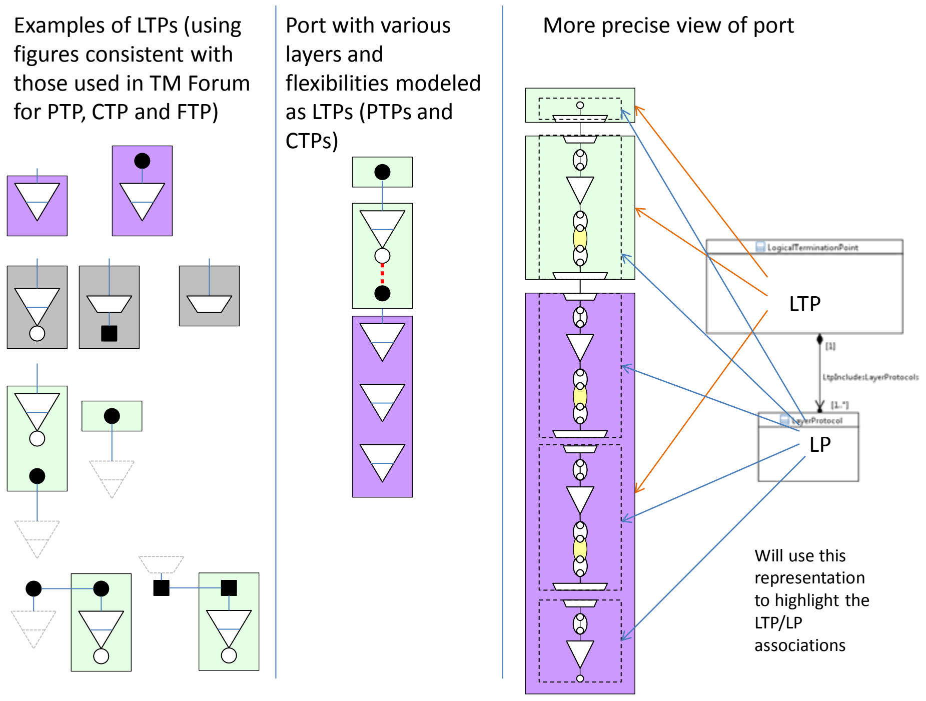

Examples of LTPs (using figures consistent with those used in TM Forum for PTP, CTP and FTP)
Port with various layers and flexibilities modeled as LTPs (PTPs and CTPs)
More precise view of port
LogicalTerminationPoint
LTP
[1]
LtpIncludesLayerProtocols
[1..*]
LayerProtocol
LP
Will use this representation to highlight the LTP/LP associations

Figure 6-5 Representations of LTPs

In Figure 6-5 above the pictorial form shows a number of representations of LTPs (purple, grey and green) representing the layering associated with physical ports (purple), their connectable clients (green) and floating LTPs (grey). The right most pictorial form shows the relationship between the LTP and the LP in terms of a detailed symbol derived from work by TM Forum and ITU-T.[25] An LP instance represents all aspects of termination of a single LayerProtocol. An LTP is composed on 1 or more LPs where the LPs represent the stack of terminations relevant to the LTP as depicted in the pictorial view. A termination stack may spread across several LTPs. The reason for this split includes multiplicity and connection flexibility transitions (see also Figure 5-2 in section 5.3 Pictorial diagram Key).

[25] The work has been liaised by TM Forum and related to Recommendation ITU-T G.805.

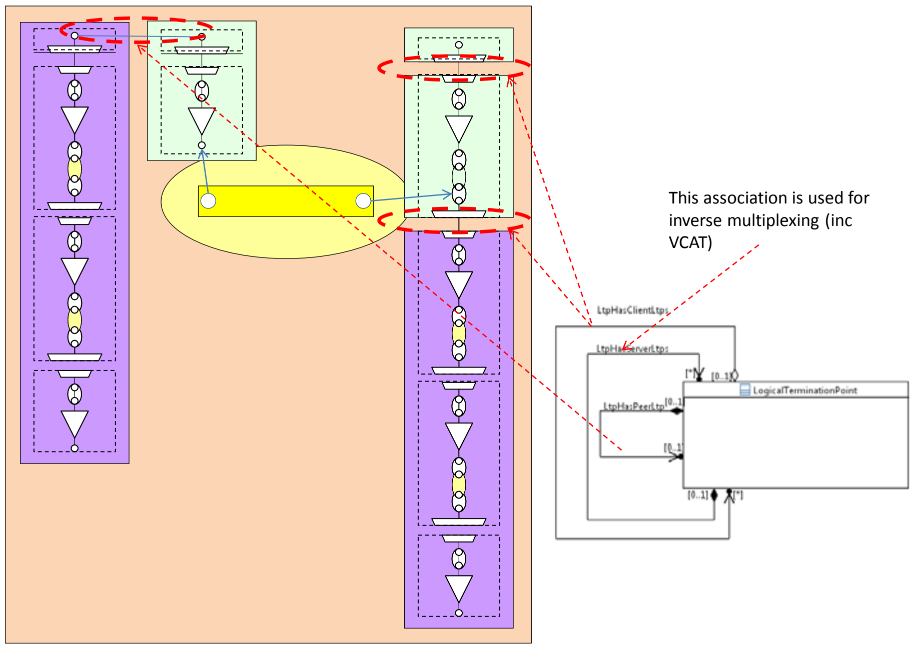


Figure 6-6 LTP relationships illustrated in a simple Network Element context

In Figure 6-6 above the pictorial form shows a number of LTPs (purple and green) representing the layering associated with physical ports (purple) and their connectable clients (green) as described in the previous section. This figure shows in more detail the partitioning of the layer stack between LTPs. Several different relationships are used at the split, depending upon the orientation of traffic flow.

Considering the left most LTP pair in the pictorial form and a signal entering the bottom of the purple LTP (at a physical port), the signal would be de-multiplexed up to the top of the purple LTP and then re-multiplexed as it travels down the green LTP. The association between the two is essentially a degenerate 1:1 FC. The LTPs are split because of the change in flow multiplexing orientation. The association supporting this relationship is shown in the UML fragment.

Considering the right most LTPs in the pictorial form and a signal entering the bottom of the purple LTP (at a physical port), the signal would be de-multiplexed up to the top of the purple LTP and then further de-multiplexed in the client LTPs. The LTPs are split because of a change in multiplicity or the opportunity to connect with an FC. The association supporting this relationship is shown in the UML fragment.

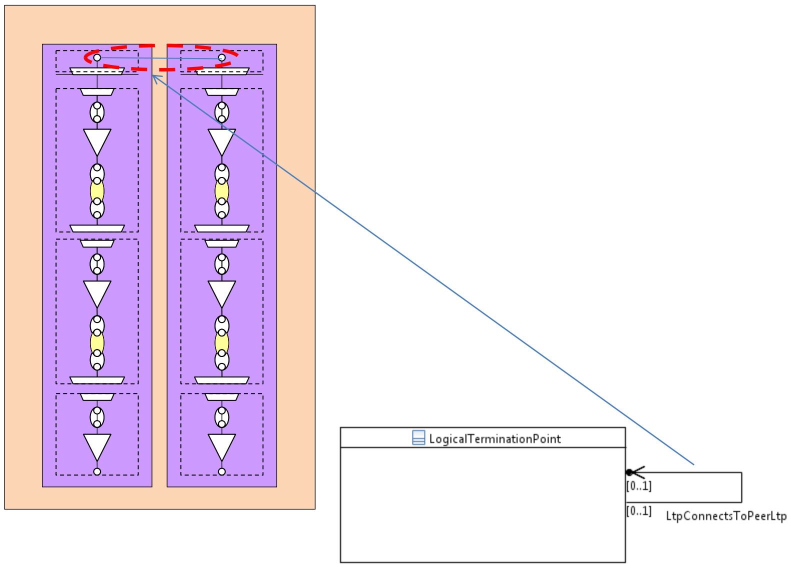


Figure 6-7 LtpConnectsToPeerLtp illustrated in an Amplifier/Regenerator context

In the simple Figure 6-7 above the final LTP to LTP association is highlighted. This allows two LTPs that are associated with physical ports without the need for an FC. This is only allowed in a case when the relationship between the LTPs is such that the whole signal from one LTP must flow to the other with no flexibility. The association effectively represents a degenerate FC.

6.4 Forwarding Subset

6.4.1 Basic Forwarding

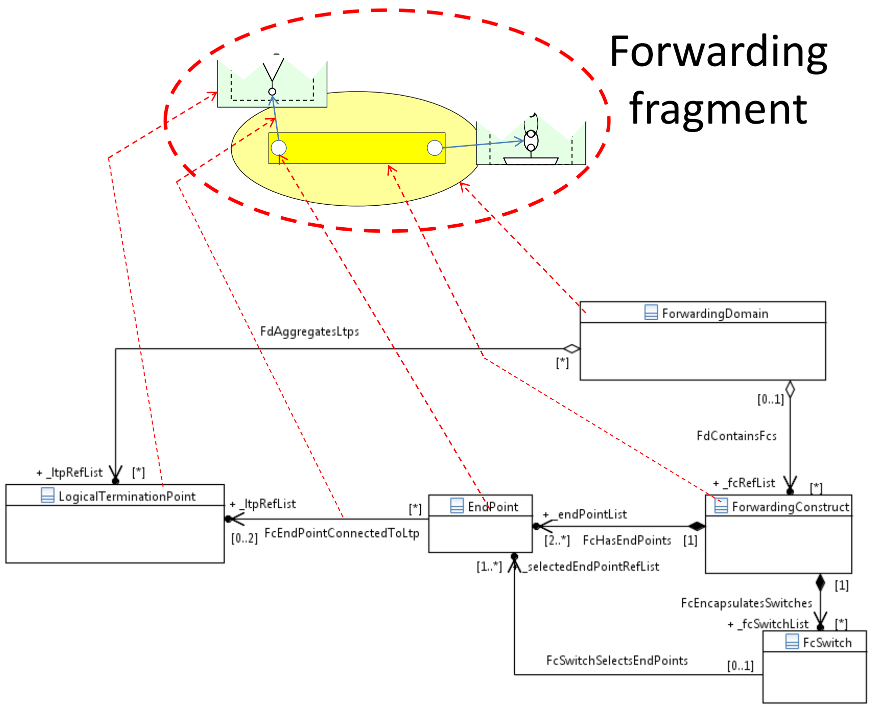


Figure 6-8 Forwarding fragment

The pictorial form in Figure 6-8 above shows the ForwardingConstruct (FC) in the context of two LTPs (a fragment of an earlier figure). The EndPoint (EP) of the FC is depicted as within the FC emphasizing the strict part-whole relationship and lifecycle dependency of the EP on the FC. The EPs are effectively ports on the FC component. The FC shown has two EPs but the model allows for two or more EPs [2..*] where in some cases the EP could be selected as a source or destination for switching. The protection switching capability is explained elsewhere in this document.

The [0..2] multiplicity of _ltpRefList allows for a bidirectional FC end to associate with two unidirectional LTPs.

6.4.2 Forwarding Construct Specification and other details of Forwarding

Prior to embarking on a brief description of the FC specification and associated classes it is important to explain the concept of specification classes in general. In this model the specification classes provide a mechanism to express the restrictions of a particular case of application of a specific class or set of classes. For example an FC may in general have [2..*] endpoints while a specific case of FC may have exactly 4. This case may also be such that it has 2 switches and such that these switches affect specific flows in the FC. The FcSpec is designed to allow the expression of cases of this sort.

At this point only limited work has been done on specification in general with a focus on the FcSpec and associated classes. It is anticipated that in general specification classes would be developed for all entities in the model.

In the diagram below the FcSpec and supporting EndpointSetSpec describe the capabilities of the FC in terms of MultiSwitchedUniFlows, each of which has [1..*] IngressEndpointSets and [1..*]

EgressEndpointSets. Each MultiSwitchedUniFlow may have [0..1] ingress switches and [0..1] egress switches where the ingress switch may select only one set member from one set and the egress switch may select [1..*] set members from the egress set. The ingress and egress switch selections are controlled by the ConfigurationAndSwitchControl element that may be:

- embedded in the switch when there is no coordination of switches required
- embedded in the FC when there is coordination of switches in the scope of the FC but no wilder
- independent of the FC and described by the ConfigurationGroupSpec where there is multi-FC coordination required

The behavior of the ConfigurationAndSwitchControl element is described by ControlRules.

The model has been exercised for a number of different cases (not detailed here). Figure 6-9 below provides the class diagram of the FC specification fragment.

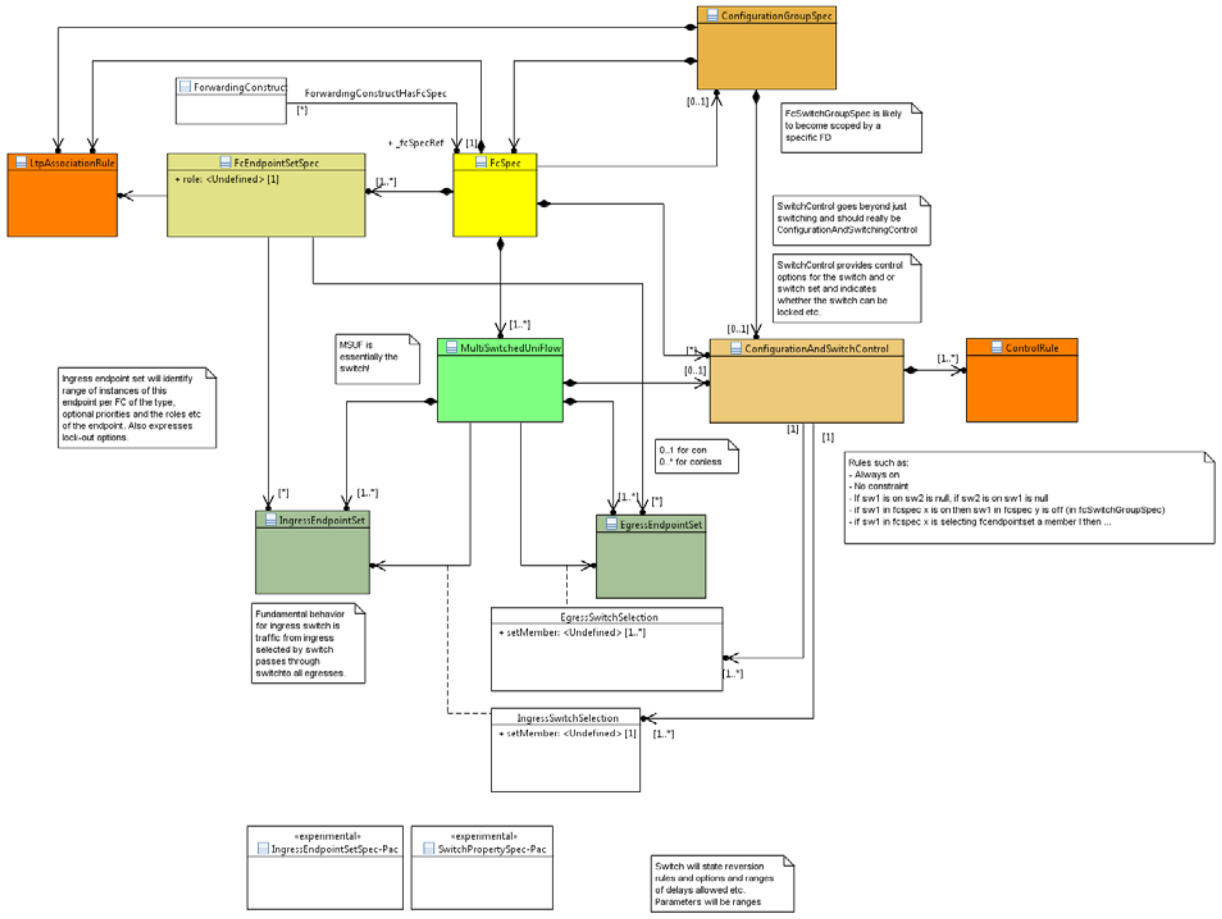

ConfigurationGroupSpec
ForwardingConstruct
ForwardingConstructHasFcSpec
[*]
+ _fcSpecRef
[1]
[0..1]
FcSwitchGroupSpec is likely to become scoped by a specific FD
LtpAssociationRule
FcEndpointSetSpec
+ role: <Undefined> [1]
[1..*]
FcSpec
SwitchControl goes beyond just switching and should really be ConfigurationAndSwitchingControl
SwitchControl provides control options for the switch and or switch set and indicates whether the switch can be locked etc.
[1..*]
[0..1]
MSUF is essentially the switch!
MultiSwitchedUniFlow
[*]
ConfigurationAndSwitchControl
[1..*]
ControlRule
[0..1]
Ingress endpoint set will identify range of instances of this endpoint per FC of the type, optional priorities and the roles etc of the endpoint. Also expresses lock-out options.
[1]
[1]
0..1 for con
0..* for conless
Rules such as:
- Always on
- No constraint
- If sw1 is on sw2 is null, if sw2 is on sw1 is null
- if sw1 in fcspec x is on then sw1 in fcspec y is off (in fcSwitchGroupSpec)
- if sw1 in fcspec x is selecting fcendpointset a member I then ...
[*]
[1..*]
IngressEndpointSet
[1..*]
[*]
EgressEndpointSet
Fundamental behavior for ingress switch is traffic from ingress selected by switch passes through switchto all egresses.
EgressSwitchSelection
+ setMember: <Undefined> [1..*]
[1..*]
IngressSwitchSelection
+ setMember: <Undefined> [1]
[1..*]
«experimental»
IngressEndpointSetSpec-Pac
«experimental»
SwitchPropertySpec-Pac
Switch will state reversion rules and options and ranges of delays allowed etc.
Parameters will be ranges

Figure 6-9 Class Diagram of the Spec Model of Connection Control

FcCapabilitySpec.PN
G

The diagrams below show a pictorial view of some of the classes above (the colors used in the figure are consistent with those used in the model above).

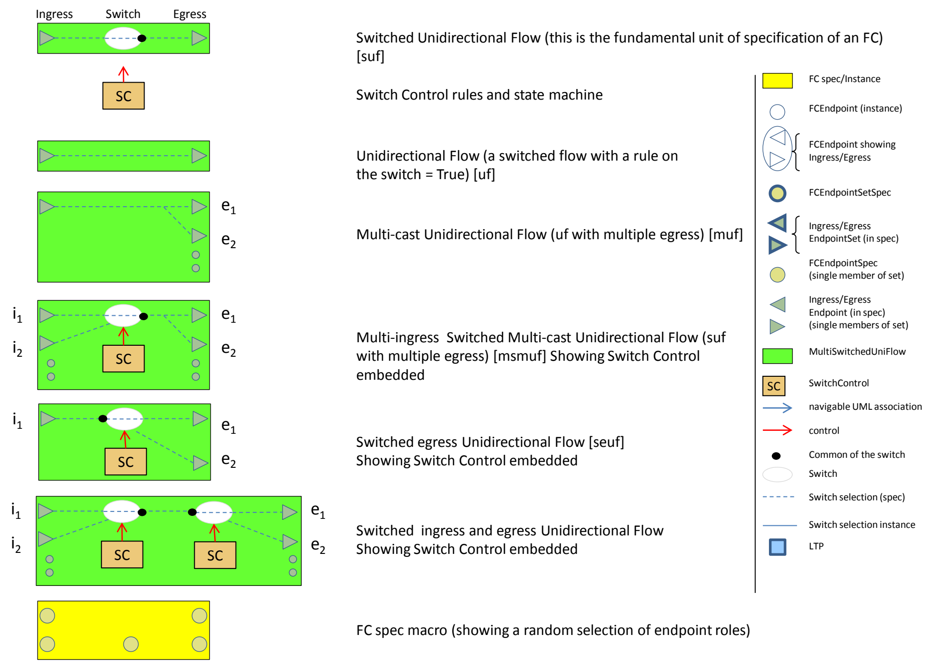

Ingress
Switch
Egress
Switched Unidirectional Flow (this is the fundamental unit of specification of an FC) [suf]
SC
Switch Control rules and state machine
Unidirectional Flow (a switched flow with a rule on the switch = True) [uf]
e1
e2
Multi-cast Unidirectional Flow (uf with multiple egress) [muf]
i1
i2
Multi-ingress Switched Multi-cast Unidirectional Flow (suf with multiple egress) [msmuf] Showing Switch Control embedded
Switched egress Unidirectional Flow [seuf] Showing Switch Control embedded
Switched ingress and egress Unidirectional Flow Showing Switch Control embedded
FC spec macro (showing a random selection of endpoint roles)
FC spec/Instance
FCEndpoint (instance)
FCEndpoint showing Ingress/Egress
FCEndpointSetSpec
Ingress/Egress EndpointSet (in spec)
FCEndpointSpec (single member of set)
Ingress/Egress Endpoint (in spec) (single members of set)
MultiSwitchedUniFlow
SwitchControl
navigable UML association
control
Common of the switch
Switch
Switch selection (spec)
Switch selection instance
LTP

Figure 6-10 Pictorial view of the Spec Model of Configuration Control

The diagrams below show a pictorial view of a case of FcSpec. The lower element of the diagram shows specification class instances and the upper element shows an instance of FC abiding by the spec.

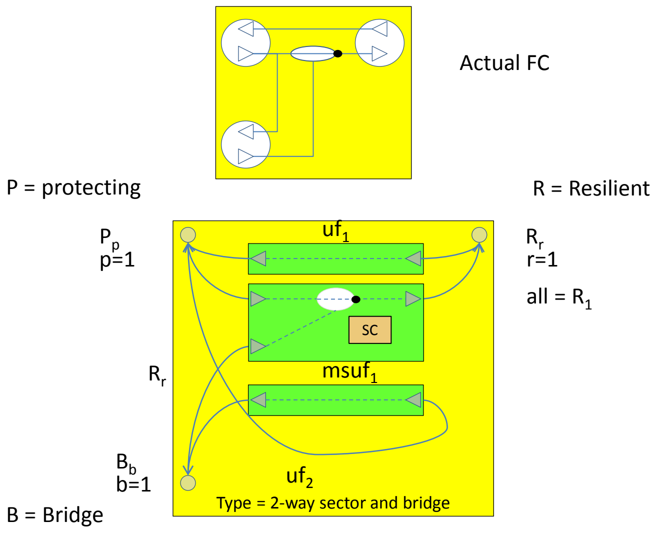
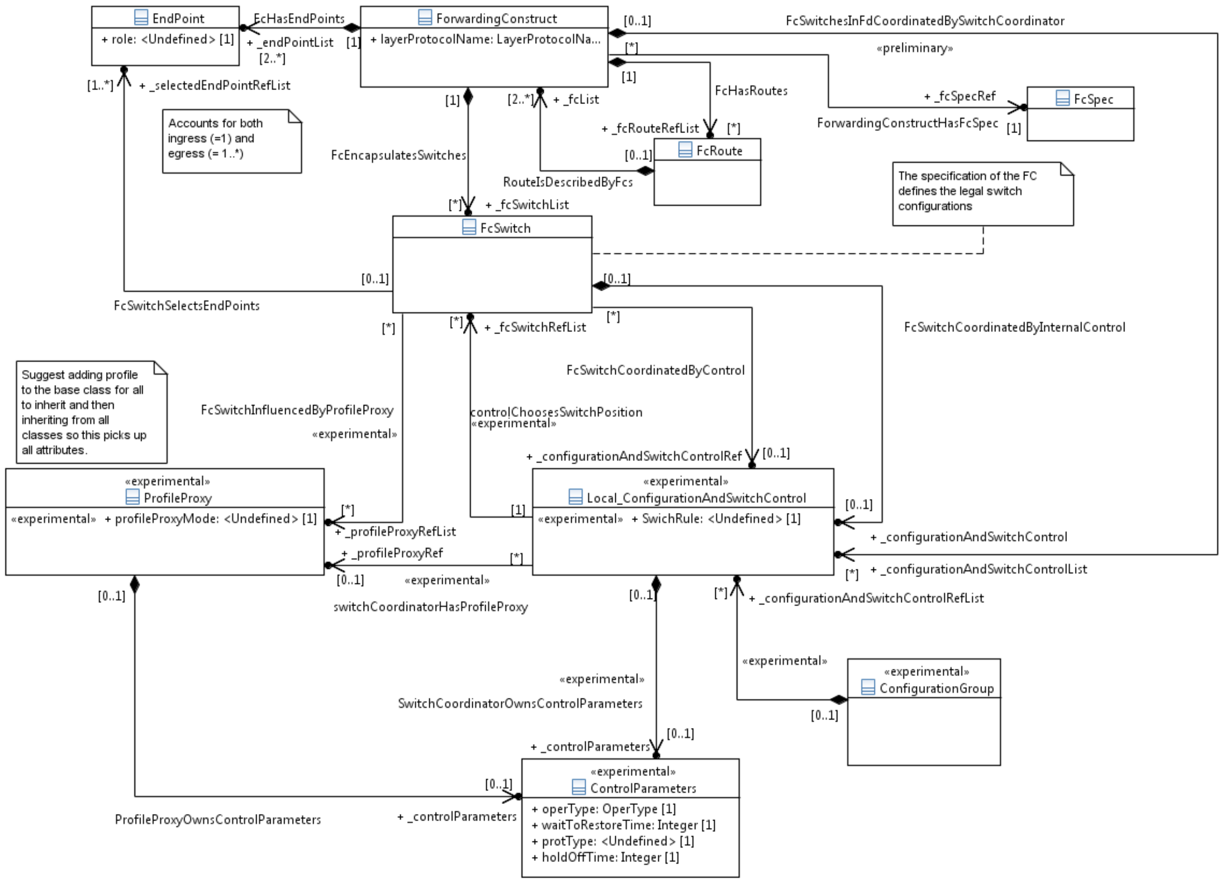


Figure 6-11 –Pictorial view of spec model and resulting FC instance

Figure **6-12** below provides the class diagram of further detailed FC and protection switching related object classes. The figure shows development of the controller of the FcSwitch. This area of model is experimental work in progress as highlighted by the «experimental» stereotypes.

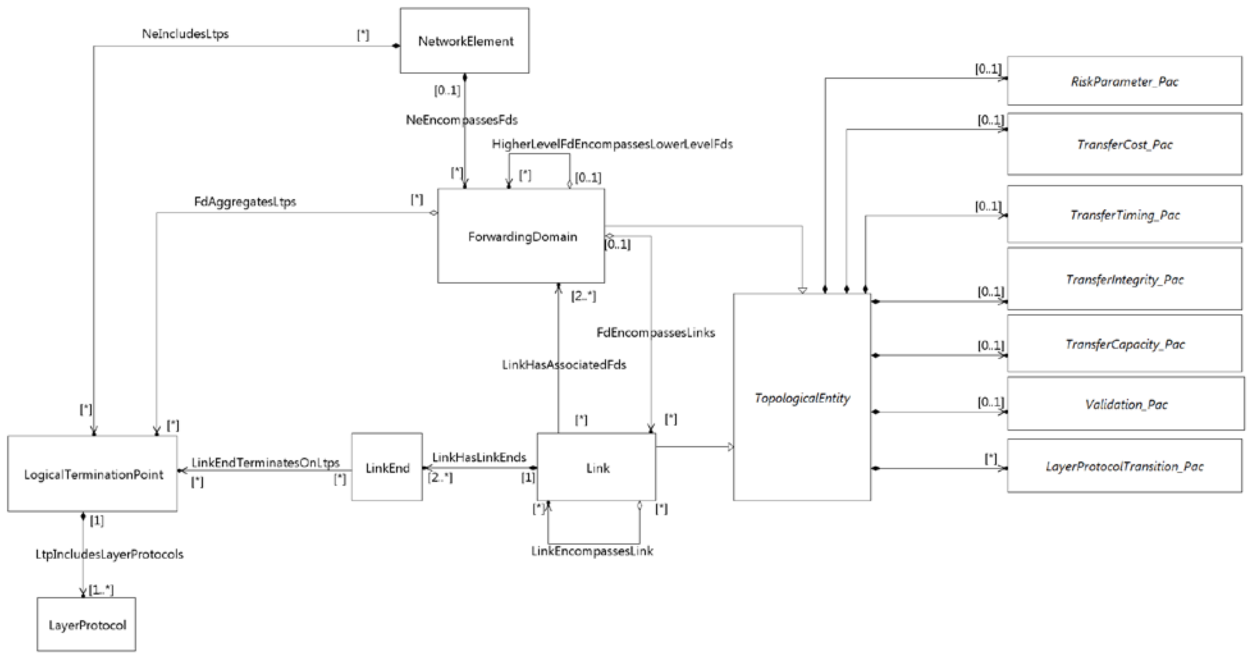

EndPoint
+ role: <Undefined> [1]
FcHasEndPoints
+ _endPointList
[2..*]
[1]
ForwardingConstruct
+ layerProtocolName: LayerProtocolNa...
[0..1]
FcSwitchesInFdCoordinatedBySwitchCoordinator
[*]
«preliminary»
[1]
FcHasRoutes
+ _fcSpecRef
FcSpec
ForwardingConstructHasFcSpec
[1]
[1..*]
+ _selectedEndPointRefList
Accounts for both ingress (=1) and egress (= 1..*)
[1]
[2..*]
+ _fcList
+ _fcRouteRefList
[*]
[0..1]
FcRoute
FcEncapsulatesSwitches
RouteIsDescribedByFcs
The specification of the FC defines the legal switch configurations
[*]
+ _fcSwitchList
FcSwitch
[0..1]
[0..1]
FcSwitchSelectsEndPoints
[*]
[*]
+ _fcSwitchRefList
[*]
FcSwitchCoordinatedByInternalControl
FcSwitchCoordinatedByControl
Suggest adding profile to the base class for all to inherit and then inheriting from all classes so this picks up all attributes.
FcSwitchInfluencedByProfileProxy
«experimental»
controlChoosesSwitchPosition
«experimental»
+ _configurationAndSwitchControlRef
[0..1]
«experimental»
ProfileProxy
«experimental» + profileProxyMode: <Undefined> [1]
[*]
+ _profileProxyRefList
+ _profileProxyRef
[0..1]
«experimental»
switchCoordinatorHasProfileProxy
[1]
[*]
«experimental»
Local_ConfigurationAndSwitchControl
«experimental» + SwichRule: <Undefined> [1]
[0..1]
+ _configurationAndSwitchControl
+ _configurationAndSwitchControlList
[*]
[0..1]
[0..1]
[*]
+ _configurationAndSwitchControlRefList
«experimental»
«experimental»
ConfigurationGroup
[0..1]
«experimental»
SwitchCoordinatorOwnsControlParameters
[0..1]
+ _controlParameters
«experimental»
ControlParameters
[0..1]
+ operType: OperType [1]
+ waitToRestoreTime: Integer [1]
+ protType: <Undefined> [1]
+ holdOffTime: Integer [1]
ProfileProxyOwnsControlParameters
+ _controlParameters

SketchOfSwitchCoor
dinatorAndProfile.PN

Figure 6-12 Class Diagram of Connection related Object Classes

6.5 Topology Subset

The topology subset is summarized in the following figure.



Topology-HighLevelO
verviewOfStructure-I

Figure 6-13 Classes of the Topology Subset

The figure above shows a lightweight view of the model omitting the attributes (where appropriate these will be described later in this section). The figure focuses on interrelationships and shows that:

– An FD may be a subordinate part of a NetworkElement, may coincide with anNetworkElement or may be larger than, and independent of, any NetworkElement (See for example FDs A.1 and A.3 in Figure 6-15).

– An FD may encompass lower level FDs. This may be such that:

  o An FD directly contained in a NetworkElement is divided into smaller parts

  o An FD not encompassed by a NetworkElement is divided into smaller parts some of which may be encompassed by NetworkElements

  o The FD represents the whole network

  Note that an FD at the lowest level of abstraction (i.e., a fabric) does not encompass FDs while an FD at the highest level of abstraction (i.e., the FD representing the whole network) is not encompassed by any higher level FDs.

– An FD encompasses Links that interconnect any FDs encompassed by the FD

Note that Offnet Links are not encompassed by any FD. All other Links are always encompassed by one FD which may be the FD representing the whole network. As a consequence, the FD representing the whole network shall always be instantiated.

- A Link may aggregate Links in several ways
    - In parallel where several links are considered as one
    - In series where Links chain to form a Link of a greater span
        - Note that this case requires further development in the model
- A Link has associated FDs that it interconnects
    - A Link may interconnect 2 or more FDs[26]
        - Note that it is usual for a Link to interconnect 2 FDs but there are cases where many may be interconnected by a Link
- A Link has LinkEnds (LE) that represent the ports of the Link itself
    - LEs are especially relevant for multi-ended asymmetric Link
- An FD aggregates LogicalTerminationPoints (LTPs) that bound it. The LTP represent a stack LayerProtocol terminations where the details of each is held in the LayerProtocol (LP). The LTP may be:
    - Part of a NetworkElement
    - Conceptually independent from any NetworkElement
- An LE references LTPs on which the Link associated to the LE terminates

Both the Link and FD are TopologicalEntities (an abstract class, i.e., a class that will never instantiate) and hence they can acquire contents from the conditional packages (_Pacs). The conditional packages provide all key topology properties.

6.5.1 Basic Topology

The first two figures focus on the ForwardingDomain class and the recursive aggregation relationship as well as the relationship between the ForwardingDomain, Link and the NetworkElement.

[26] An off-network link with two ends does not interconnect any FDs in the view.

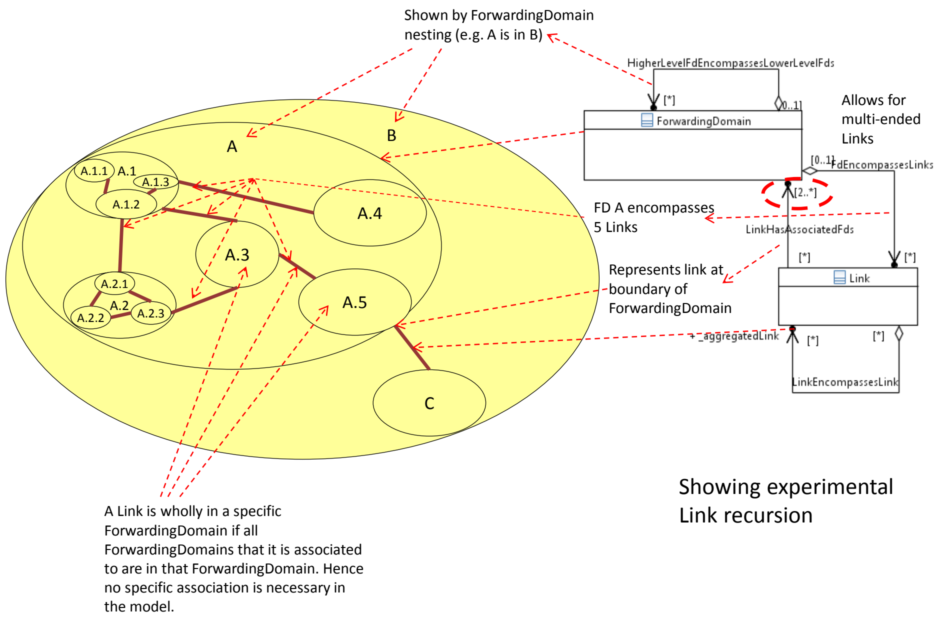

Shown by ForwardingDomain nesting (e.g. A is in B)
HigherLevelFdEncompassesLowerLevelFds
[*]
[0..1]
ForwardingDomain
Allows for multi-ended Links
[0..1]
FdEncompassesLinks
[2..*]
FD A encompasses 5 Links
LinkHasAssociatedFds
[*]
[*]
Link
Represents link at boundary of ForwardingDomain
+ _aggregatedLink
[*]
[*]
LinkEncompassesLink
A
B
A.1.1
A.1
A.1.3
A.1.2
A.4
A.3
A.2.1
A.2
A.2.3
A.2.2
A.5
C
Showing experimental Link recursion
A Link is wholly in a specific ForwardingDomain if all ForwardingDomains that it is associated to are in that ForwardingDomain. Hence no specific association is necessary in the model.

Figure 6-14 ForwardingDomain recursion with Link

Figure 6-14 shows a UML fragment including the Link and ForwardingDomain (FD). For simplicity it is assumed here that the Links and FDs are for a single LayerProtocol (LP) although an FD can support a list of LPs.

The pictorial form shows a number of instances of FD interconnected by Links and shows nesting of FDs. The recursive aggregation HigherLevelFdEncompassesLowerLevelFds relationship (aggregation is represented by an open diamond) supports the ForwardingDomain nesting, but it should be noted that this is intentionally showing no lifecycle dependency between the lower ForwardingDomains and the higher ones that nest them (to do this composition, a black diamond would have been used instead of an open diamond). This is to allow for rearrangements of the ForwardingDomain hierarchy (e.g., when regions of a network are split or merged) and to emphasize that the nesting is an abstraction rather than decomposition. The underlying network still operates regardless of how it is perceived in terms of aggregating ForwardingDomains. The model allows for only one hierarchy.

In the example of Figure 6-14, there are fourteen FD instances with the following instances of the "HigherLevelFdEncompassesLowerLevelFds" relationships:

- B encompasses two FDs: A and C
- A encompasses five FDs: A.1, A.2, A.3, A.4 and A.5
- A.1 encompasses three FDs: A.1.1, A.1.2 and A.1.3
- A.2 encompasses three FDs: A.2.1, A.2.2 and A.2.3

When one FD is removed, the “HigherLevelFdEncompassesLowerLevelFds” relationships are modified. For example, if FD A.1 in Figure 4-2 is removed, the instances of the “HigherLevelFdEncompassesLowerLevelFds” relationships will be modified as follows:

- B encompasses two FDs: A and C
- A encompasses seven FDs: A.1.1, A.1.2, A.1.3, A.2, A.3, A.4 and A.5[27]
- A.2 encompasses three FDs: A.2.1, A.2.2 and A.2.3

An FD can also be added. Initially it will have no associated lower level FDs. Existing FDs can be moved as appropriate to form the new hierarchy.

The association between Link and FD allows a Link to be terminated on two or more FDs (see section 6.1.7 Link and LinkEnd on page 17). Through this the model supports point to point Links as well as cases where the server ForwardingConstruct is multi-point terminated giving rise to a multi-pointed Link. Multi-pointed links occur in PON and Layer 2 MAC in MAC.[28]

It should be noted that the model includes LinkEnd which further details the relationship between FD and Link. This is explained below.

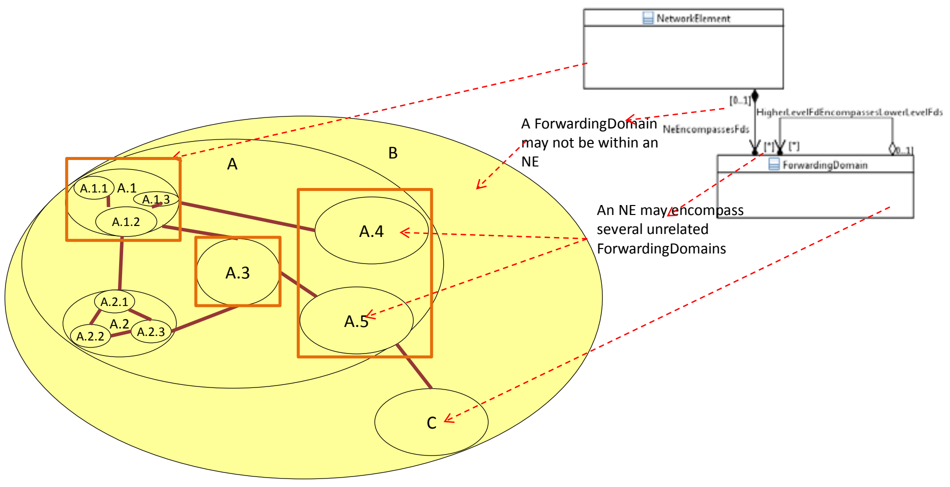

NetworkElement
[0..1]
HigherLevelFdEncompassesLowerLevelFds
NeEncompassesFds
[*]
[*]
[0..1]
ForwardingDomain
A ForwardingDomain may not be within an NE
An NE may encompass several unrelated ForwardingDomains
A
B
A.1.1
A.1
A.1.3
A.1.2
A.3
A.4
A.5
A.2.1
A.2
A.2.2
A.2.3
C

Figure 6-15 ForwardingDomain recursion with link and NetworkElement

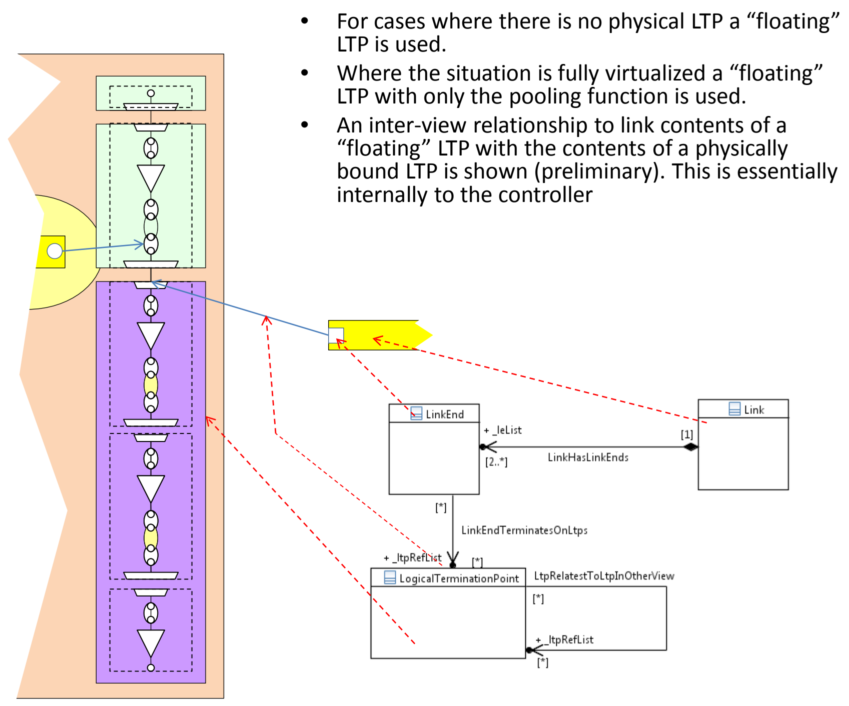
[27] Clearly the FD naming in the figure is for ease of reading the diagram and does not represent hierarchy.
[28] Work supporting this was liaised from TM Forum.

In Figure 6-15 above the pictorial form shows an overlay of NetworkElement on the ForwardingDomains and a corresponding fragment of UML showing only the ForwardingDomain and NetworkElement classes.

The figure emphasizes that at and below one particular level of abstraction of ForwardingDomain, the ForwardingDomains are all bounded by a specific NetworkElement. This is represented in the UML fragment by the composition association (black diamond) that explains that there is a lifecycle dependency in that the ForwardingDomain at this level cannot exist without the NetworkElement. The figure also shows that a ForwardingDomain need not be bounded by anNetworkElement(as explained in the UML fragment by the 0..1 composition), and that a ForwardingDomain may have a smaller scope than the wholeNetworkElement(even when considering only a single LayerProtocol as noted earlier). In one case depicted (e.g., the right hand side NetworkElement encompassing two FDs), the two ForwardingDomains in the NetworkElement are completely independent. In the other cases depicted (e.g., the left hand side NetworkElement encompassing three FDs), the subordinate ForwardingDomains are themselves joined by Links emphasizing that the NetworkElement does not necessarily represent the lowest level of relevant network decomposition.

The figure also emphasizes that just because one ForwardingDomain at a particular level of decomposition of the network happens to be the one bounded by a NetworkElement does not mean that all ForwardingDomains at that level are also bounded by NetworkElements.[29]

6.5.2 Advanced Topology



- For cases where there is no physical LTP a "floating" LTP is used.
- Where the situation is fully virtualized a "floating" LTP with only the pooling function is used.
- An inter-view relationship to link contents of a "floating" LTP with the contents of a physically bound LTP is shown (preliminary). This is essentially internally to the controller

[29] It should be noted that a NetworkElement is never within the bounds of an FD. The NetworkElement is associated with levels in the FD hierarchy.

Figure 6-16 LTP “pooling” client LTPs

Figure 6-16 above shows how the Link terminates on the LTP via the LinkEnd (LE).

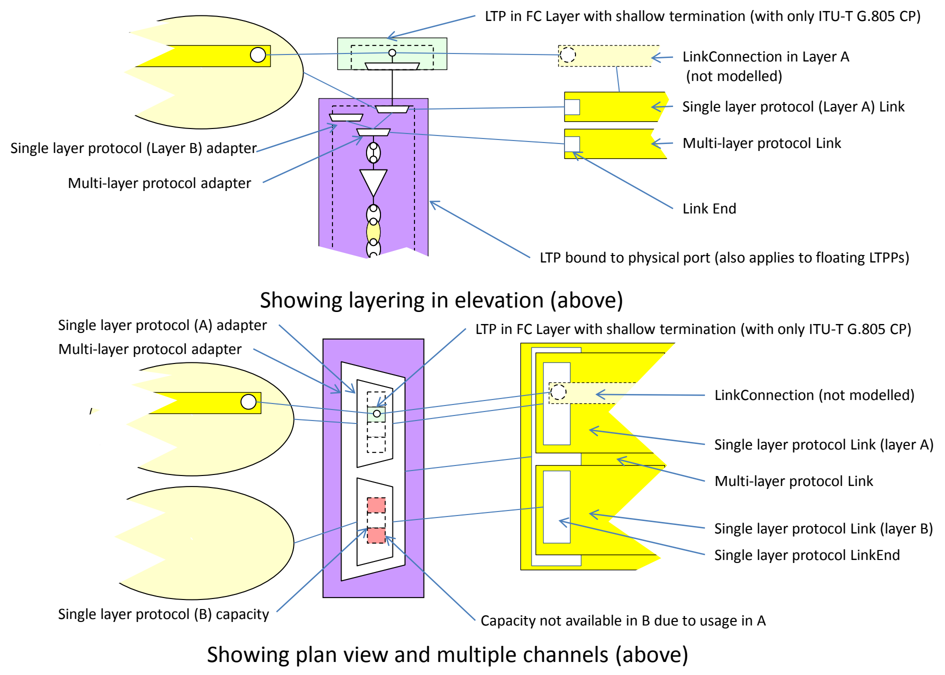

LTP in FC Layer with shallow termination (with only ITU-T G.805 CP)
LinkConnection in Layer A (not modelled)
Single layer protocol (Layer A) Link
Multi-layer protocol Link
Single layer protocol (Layer B) adapter
Multi-layer protocol adapter
Link End
LTP bound to physical port (also applies to floating LTPPs)
Showing layering in elevation (above)
Single layer protocol (A) adapter
Multi-layer protocol adapter
LTP in FC Layer with shallow termination (with only ITU-T G.805 CP)
LinkConnection (not modelled)
Single layer protocol Link (layer A)
Multi-layer protocol Link
Single layer protocol Link (layer B)
Single layer protocol LinkEnd
Single layer protocol (B) capacity
Capacity not available in B due to usage in A
Showing plan view and multiple channels (above)

Figure 6-17 Views of Link, LinkEnd and LTP showing LTP pooling

The LTP may have the capability[30] to map to multiple client layer protocols where there is an interaction between the client mappings (e.g., if capacity/channel x of client layer protocol A is used then capacity/channel set y of client layer protocol B is no longer available). The capacity of the Link is determined by evaluating the “intersection” of capabilities of the LTPs at the ends (which is complex in a multi-ended case).

The used capacity is determined by considering which client LTPs exist as a result of their being FCs.

A Link may be multi-layered and hence may represent the whole client capacity of an LTP or it may be single layered.

[30] This capability of the LTP is not currently modeled but work is under way to construct an LTP specification model

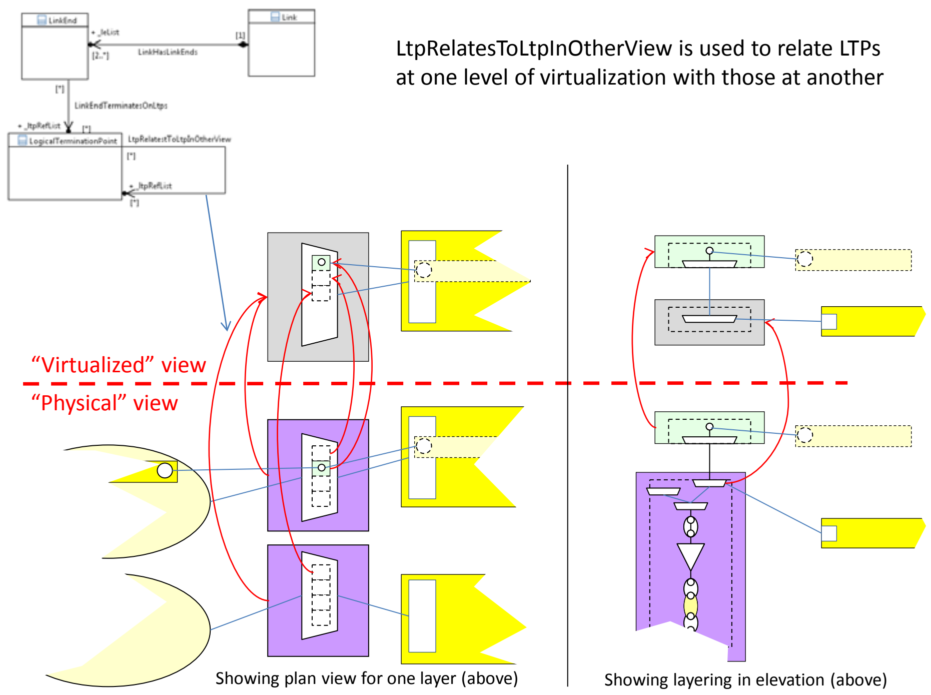
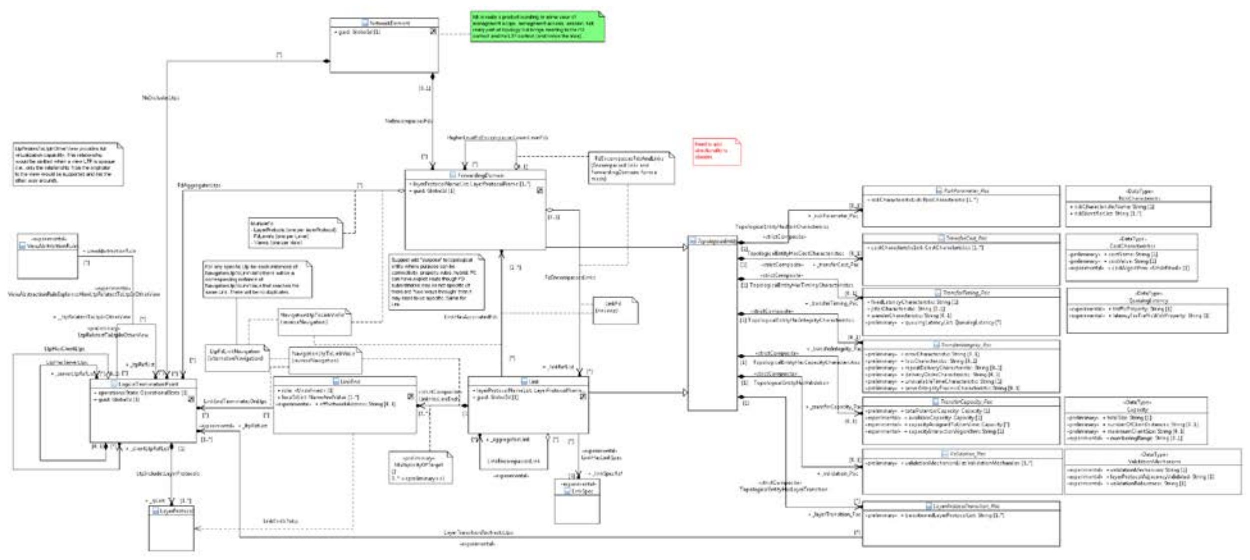

LinkEnd
Link
_leList
[2..*]
[1]
LinkHasLinkEnds
[*]
LinkEndTerminatesOnLtps
_ltpRefList
[*]
LogicalTerminationPoint
LtpRelatesToLtpInOtherView
[*]
_ltpRefList
[*]
LtpRelatesToLtpInOtherView is used to relate LTPs at one level of virtualization with those at another
“Virtualized” view
“Physical” view
Showing plan view for one layer (above)
Showing layering in elevation (above)

Figure 6-18 Views of “virtualization” of LTP

Some capacity may be taken from each of a number of Links supporting a particular layer protocol and offered in a “virtualized” view perhaps for use in a particular application etc. The “virtualized” view will normally be referenced in a different name space. The rules for grouping capacity into Links in the “virtualized” view have not yet been documented. The same model is used for Links and LTPs in the “virtualized” view as is used in the “physical” view.

6.5.3 Detailed properties of Topology

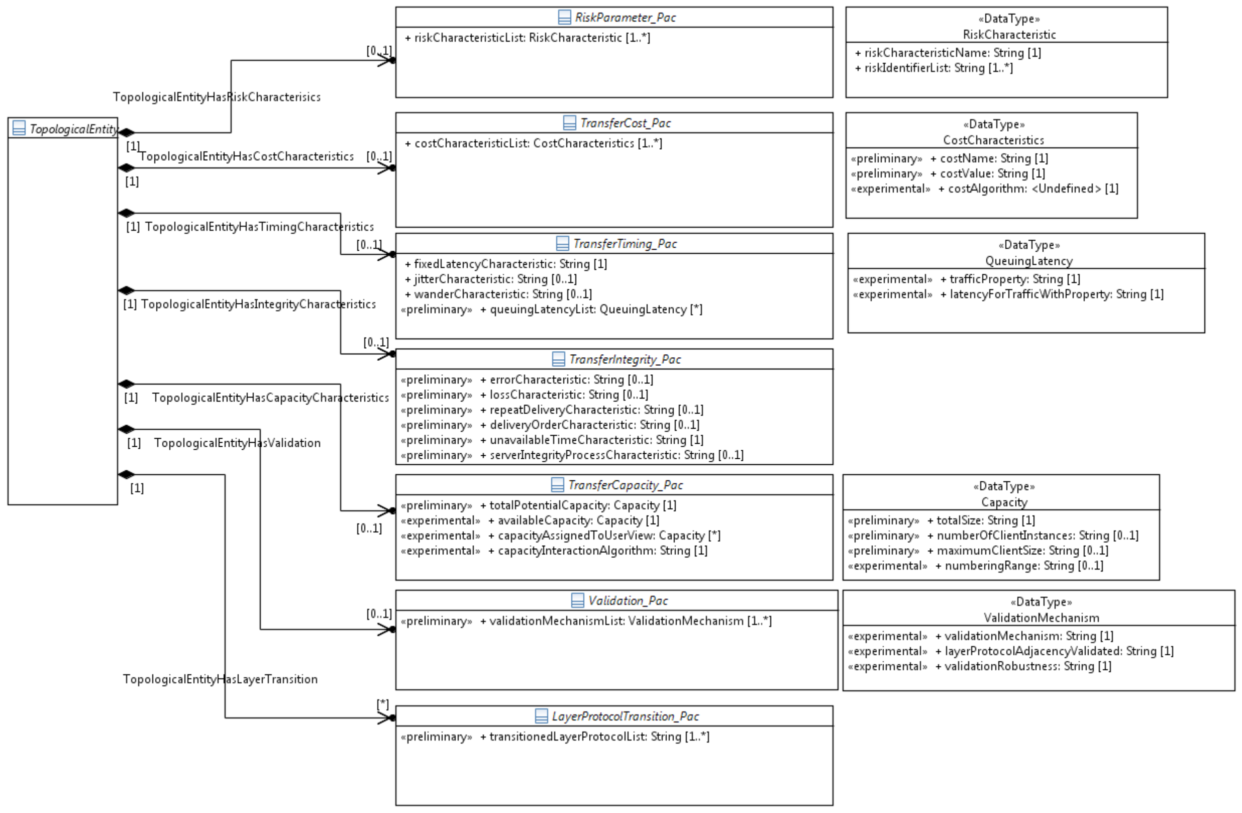


TopologyOverview.PNG

Figure 6-19 Topology detail

The figure above shows finalized, preliminary and experimental extensions of the Topology model. The model recognizes that both ForwardingDomain and Link share topological properties (the TopologicalEntity, which is abstract and hence not intended to be instantiated, provides the linkage[31]). The classes related to TopologicalEntity, the _Pacs, are « strictComposition » and hence are essentially part of the ForwardingDomain and of the Link. The _Pacs are optional as in some cases of Link/ForwardingDomain they are essentially not relevant.

The figure below shows the _Pacs in more detail.


TopologicalEntity
TopologicalEntityHasRiskCharacterisics
[1]
[0..1]
TopologicalEntityHasCostCharacteristics
[1]
[0..1]
[1] TopologicalEntityHasTimingCharacteristics
[0..1]
[1] TopologicalEntityHasIntegrityCharacteristics
[0..1]
[1] TopologicalEntityHasCapacityCharacteristics
[0..1]
[1] TopologicalEntityHasValidation
[0..1]
[1]
TopologicalEntityHasLayerTransition
[*]
RiskParameter_Pac
+ riskCharacteristicList: RiskCharacteristic [1..*]
«DataType» RiskCharacteristic
+ riskCharacteristicName: String [1]
+ riskIdentifierList: String [1..*]
TransferCost_Pac
+ costCharacteristicList: CostCharacteristics [1..*]
«DataType» CostCharacteristics
«preliminary» + costName: String [1]
«preliminary» + costValue: String [1]
«experimental» + costAlgorithm: <Undefined> [1]
TransferTiming_Pac
+ fixedLatencyCharacteristic: String [1]
+ jitterCharacteristic: String [0..1]
+ wanderCharacteristic: String [0..1]
«preliminary» + queuingLatencyList: QueuingLatency [*]
«DataType» QueuingLatency
«experimental» + trafficProperty: String [1]
«experimental» + latencyForTrafficWithProperty: String [1]
TransferIntegrity_Pac
«preliminary» + errorCharacteristic: String [0..1]
«preliminary» + lossCharacteristic: String [0..1]
«preliminary» + repeatDeliveryCharacteristic: String [0..1]
«preliminary» + deliveryOrderCharacteristic: String [0..1]
«preliminary» + unavailableTimeCharacteristic: String [1]
«preliminary» + serverIntegrityProcessCharacteristic: String [0..1]
TransferCapacity_Pac
«preliminary» + totalPotentialCapacity: Capacity [1]
«experimental» + availableCapacity: Capacity [1]
«experimental» + capacityAssignedToUserView: Capacity [*]
«experimental» + capacityInteractionAlgorithm: String [1]
«DataType» Capacity
«preliminary» + totalSize: String [1]
«preliminary» + numberOfClientInstances: String [0..1]
«preliminary» + maximumClientSize: String [0..1]
«experimental» + numberingRange: String [0..1]
Validation_Pac
«preliminary» + validationMechanismList: ValidationMechanism [1..*]
«DataType» ValidationMechanism
«experimental» + validationMechanism: String [1]
«experimental» + layerProtocolAdjacencyValidated: String [1]
«experimental» + validationRobustness: String [1]
LayerProtocolTransition_Pac
«preliminary» + transitionedLayerProtocolList: String [1..*]

Topology_Pacs-DetailedProperties.PNG

Figure 6-20 Topology _Pac detail

[31] TopologicalEntity has no direct attributes and only relationships that the ForwardingDomain and Link inherit.

© 2015 Open Networking Foundation

As shown in the figure an object class “TopologicalEntity” has been defined to collect topology-related properties (characteristics, etc.) that are common for FD and Link.

A TopologicalEntity is an abstract representation of the emergent effect of the combined functioning of an arrangement of components (running hardware, software running on hardware, etc.). The effect can be considered as the realization of the potential for apparent communication adjacency for entities that are bound to the terminations at the boundary of the TopologicalEntity.

The TopologicalEntity enables the creation of constrained forwarding to achieve the apparent adjacency. The apparent adjacency has intended performance degraded from perfect adjacency and a statement of that degradation is conveyed via the attributes of the packages associated with this class. In the model both ForwardingDomain and Link are TopologicalEntities.

This abstract class is used as a modeling approach to apply packages of attributes to both Link and ForwardingDomain. Link and ForwardingDomain are the key TopologicalEntities.

The _Pacs are detailed in the following sections.

6.5.3.1 RiskParameter_Pac

The risk characteristics of a TopologicalEntity come directly from the underlying physical realization.

The risk characteristics propagate from the physical realization to the client and from the server layer to the client layer, this propagation may be modified by protection.

A TopologicalEntity may suffer degradation or failure as a result of a problem in a part of the underlying realization.

The realization can be partitioned into segments which have some relevant common failure modes.

There is a risk of failure/degradation of each segment of the underlying realization.

Each segment is a part of a larger physical/geographical unit that behaves as one with respect to failure (i.e., a failure will have a high probability of impacting the whole unit (e.g.. all cables in the same duct).

Disruptions to that larger physical/geographical unit will impact (cause failure/errors to) all TopologicalEntities that use any part of that larger physical/geographical entity.

Any TopologicalEntity that uses any part of that larger physical/geographical unit will suffer impact and hence each TopologicalEntity shares risk.

The identifier of each physical/geographical unit that is involved in the realization of each segment of a Topological entity can be listed in the RiskParameter_Pac of that TopologicalEntity.

A segment has one or more risk characteristics.

Shared risk between two TopologicalEntities compromises the integrity of any solution that uses one of those TopologicalEntities as a backup for the other.

Where two TopologicalEntities have a common risk characteristic, they have an elevated probability of failing simultaneously, compared to two TopologicalEntities that do not share risk characteristics.

- riskCharacteristicList: A list of risk characteristics (RiskCharacteristic) for consideration in an analysis of shared risk. Each element of the list represents a specific risk consideration.
- RiskCharacteristic: The information for a particular risk characteristic where there is a list of risk identifiers related to that characteristic. It includes:
    - riskCharacteristicName: The name of the risk characteristic. The characteristic may be related to a specific degree of closeness. For example a particular characteristic may apply to failures that are localized (e.g., to one side of a road) where as another characteristic may relate to failures that have a broader impact (e.g., both sides of a road that crosses a bridge). Depending upon the importance of the traffic being routed, different risk characteristics will be evaluated.
    - riskIdentifierList: A list of the identifiers of each physical/geographic unit (with the specific risk characteristic) that is related to a segment of the TopologicalEntity.

6.5.3.2 TransferCost_Pac

The cost characteristics of a TopologicalEntity are not necessarily correlated to the cost of the underlying physical realization.

They may be quite specific to the individual TopologicalEntity, e.g., opportunity cost. Relates to layer capacity.

There may be many perspectives from which cost may be considered for a particular TopologicalEntity and hence many specific costs and potentially cost algorithms.

Using an entity will incur a cost.

- costCharacteristicList: The list of costs (CostCharacteristic) where each cost relates to some aspect of the TopologicalEntity.
    - CostCharcteristic: The information for a particular cost characteristic
        - costName: The cost characteristic will related to some aspect of the TopologicalEntity (e.g. $ cost, routing weight). This aspect will be conveyed by the costName
        - costValue: The specific cost.
        - costAlgorithm: The cost may vary based upon some properties of the TopologicalEntity. The rules for the variation are conveyed by the costAlgorithm.

6.5.3.3 TransferTiming_Pac

A link will suffer effects from the underlying physical realization related to the timing of the information passed by the link.

- fixedLatencyCharacteristic: A TopologicalEntity suffers delay caused by the realization of the servers (e.g., distance related; FEC encoding etc.), along with some client specific processing. This is the total average latency effect of the TopologicalEntity.
- jitterCharacteristic: High frequency deviation from true periodicity of a signal and therefore a small high rate of change of transfer latency. Applies to TDM systems (and not packet).
- wanderCharacteristics: Low frequency deviation from true periodicity of a signal and therefore a small low rate of change of transfer latency. Applies to TDM systems (and not packet).
- queuingLatencyList: The effect on the latency of a queuing process. This only has significant effect for packet based systems and has a complex characteristic (QueuingLatency).
    - QueuingLatency: Provides information on latency characteristic for a particular stated trafficProperty.

6.5.3.4 *TransferIntegrity_Pac*

Transfer integrity characteristic covers expected (specified) error, loss and duplication signal content as well as any damage of any form to total link and to the client signals.

- errorCharacteristic: describes the degree to which the signal propagated can be errored. Applies to TDM systems, as the errored signal will be propagated, but not to packet systems, as errored packets will be discarded.
- lossCharacteristic: Describes the acceptable characteristic of lost packets where loss may result from discard due to errors or overflow. Applies to packet systems and not TDM (as for TDM errored signals are propagated unless grossly errored and overflow/underflow turns into timing slips).
- repeatDeliveryCharacteristic: Primarily applies to packet systems where a packet may be delivered more than once (in fault recovery for example). It can also apply to TDM where several frames may be received twice due to switching in a system with a large differential propagation delay.
- deliveryOrderCharacteristic: Describes the degree to which packets will be delivered out of sequence. Does not apply to TDM as the TDM protocols maintain strict order.
- unavailableTimeCharacteristic: Describes the duration for which there may be no valid signal propagated.
- serverIntegrityProcessCharacteristic: Describes the effect of any server integrity enhancement process on the characteristics of the TopologicalEntity.

6.5.3.5 *TransferIntegrity_Pac*

Transfer integrity characteristic covers expected (specified) error, loss and duplication signal content as well as any damage of any form to total link and to the client signals.

- errorCharacteristic: describes the degree to which the signal propagated can be errored. Applies to TDM systems, as the errored signal will be propagated, butnot to packet systems, as errored packets will be discarded.
- lossCharacteristic: Describes the acceptable characteristic of lost packets where loss may result from discard due to errors or overflow. Applies to packet systems and not TDM systems (as in TDM systems errored signals are propagated, unless grossly errored, and overflow/underflow turns into timing slips).
- repeatDeliveryCharacteristic: Primarily applies to packet systems where a packet may be delivered more than once (in fault recovery for example). It can also apply to TDM where several frames may be received twice due to switching in a system with a large differential propagation delay.
- deliveryOrderCharacteristic: Describes the degree to which packets will be delivered out of sequence. Does not apply to TDM as the TDM protocols maintain strict order.
- unavailableTimeCharacteristic: Describes the duration for which there may be no valid signal propagated.
- serverIntegrityProcessCharacteristic: Describes the effect of any server integrity enhancement process on the characteristics of the TopologicalEntity.

6.5.3.6 TransferCapcity_Pac

The TopologicalEntity derives capacity from the underlying realization.

A TopologicalEntity may be an abstraction and virtualization of a subset of the underlying capability offered in a view or may be directly reflecting the underlying realization.

A TopologicalEntity may be directly used in the view or may be assigned to another view for use.

The clients supported by a multi-layer TopologicalEntity may interact such that the resources used by one client may impact those available to another. This is derived from the LTP spec details.

A TopologicalEntity represents the capacity available to user (client) along with client interaction and usage.

A TopologicalEntity may reflect one or more client protocols and one or more members for each profile.

- totalPotentialCapacity: An optimistic view of the capacity of the TopologicalEntity assuming that any shared capacity is available to be taken.

Note that this area is still under development to cover concepts such as:

- exclusiveCapacityList: The capacity allocated to this TopologicalEntity for its exclusive use.
- sharedCapacityList: The capacity allocated to this TopologicalEntity that is not exclusively available as it is shared with others.
- assignedAsExclusiveCapacityList: The capacity assigned from this TopologicalEnity to another TopologicalEntity for its exclusive use.

© 2015 Open Networking Foundation

- assignedAsSharedCapacityList: The capacity assigned to one or more other TopologicalEntities for shared use where the interaction follows some stated algorithm.
- Capacity which includes:
  - totalSize
  - numberOfUsageInstances
  - maximumUsageSize
  - numberingRange

6.5.3.7 Validation_Pac

Validation covers the various adjacent discovery and reachability verification protocols. Also may cover Information source and degree of integrity.

- validationMechanismList: Provides details of the specific validation mechanism(s) used to confirm the presence of an intended topologicalEntity.

6.5.3.8 LayerProtocolTransition_Pac

Relevant for a Link that is formed by abstracting one or more LTPs (in a stack) to focus on the flow and deemphasize the protocol transformation.

This abstraction is relevant when considering multi-layer routing.

The layer protocols of the LTP and the order of their application to the signal is still relevant and need to be accounted for. This is derived from the LTP spec details.

This Pac provides the relevant abstractions of the LTPs and provides the necessary association to the LTPs involved.

Links that included details in this Pac are often referred to as Transitional Links.

- transitionedLayerProtocolList: Provides the ordered structure of layer protocol transitions encapsulated in the TopologicalEntity. The ordering relates to the LinkEnd role.

7 Future CoreModel areas

Potential future areas of work in the CoreModel include

- Profiles, Templates and Specifications Module
- Management-Control Component Module
- Assurance Module
- Synchronization (frequency and time/phase) Subset
- ECC Subset
- Policy Module

- Physical Equipment Module
- Generalized OAM functions e.g., generalized MEPs

8 UML model files

8.1 Papyrus File

This section provides the link to the information model file and the companion Open Model Profile file specified using the "Papyrus" modeling tool and also some model sketches to assist the understanding of the model.

Link of the Core Model files: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/CoreInformationModel.V1.0.zip

- Model: consists of four files:
  - .project,
  - CoreModel.di,
  - CoreModel.notation
  - CoreModel.uml
- Profiles:
  - OpenModelProfiles folder

In order to view and further extend or modify the information model, one will need to install the open source Eclipse software and the Papyrus tool. The installation guide for Eclipse and Papyrus can be found at https://www.eclipse.org/papyrus/updates/index.php.

8.2 Data Dictionary File

A data dictionary format of the information model in MS WORD document will be generated in the future. The data dictionary includes the description and properties of the object classes and their attributes and associations etc.

- Core Network Module data dictionary
  << To be provided>>

- Core Foundation Module data dictionary
  << To be provided>>

9 Back matter

9.1 Editors

Kam LAM, Alcatel-Lucent

Nigel DAVIS, Ciena

9.2 Contributors

Dieter BELLER,	Alcatel-Lucent
Sergio BELOTTI,	Alcatel-Lucent
Kam LAM,	Alcatel-Lucent
Eve VARMA,	Alcatel-Lucent
Nigel DAVIS,	Ciena
Jonathan SADLER,	Corian
Bernd ZEUNER,	Deutsch Telekom
Dave HOOD,	Ericsson
Meral SHIRAZIPOUR,	Ericsson
Scott MANSFIELF,	Ericsson
Xiang YUN,	FiberHome
Yuji TOCHIO,	Fujitsu
Italo BUSI,	Huawei
Maarten VISSERS,	Huawei
Raymond CHEH,	Juniper
Karthik SETHURAMAN,	NEC
Malcolm BETTS,	ZTE