Core Information Model (CoreModel)

TR-512.4

Topology

Version 1.2
September 20, 2016

ONF Document Type: Technical Recommendation
ONF Document Name: Core Information Model version 1.2

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

©2016 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.

Table of Contents

© 2016 Open Networking Foundation

List of Figures

© 2016 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).
1.1	November 24, 2015	Version 1.1
1.2	September 20, 2016	Version 1.2 [Note Version 1.1 was a single document whereas 1.2 is broken into a number of separate parts]

1 Introduction

This document is an addendum to the TR-512_v1.2 ONF Core Information Model and forms part of the description of the ONF-CIM. For general overview material and references to the other parts refer to TR-512.1 ONF Core IM - Overview.

1.1 References

For a full list of references see TR-512.1.

1.2 Definitions

For a full list of definition see TR-512.1.

1.3 Conventions

See TR-512.1 for an explanation of:

- UML conventions
- Lifecycle Stereotypes
- Diagram symbol set

1.4 Viewing UML diagrams

Some of the UML diagrams are very dense. To view them either zoom (sometimes to 400%), open the associated image file (and zoom appropriately) or open the corresponding UML diagram via Papyrus (for each figure with a UML diagram the UML model diagram name is provided under the figure or within the figure).

1.5 Understanding the figures

Figures showing fragments of the model using standard UML symbols and also figures illustrating application of the model are provided throughout this document. Many of the application-oriented figures also provide UML class diagrams for the corresponding model fragments (see TR-512.1 for diagram symbol sets). All UML diagrams depict a subset of the relationships between the classes, such as inheritance (i.e. specialization), association relationships (such as aggregation and composition), and conditional features or capabilities. Some UML diagrams also show further details of the individual classes, such as their attributes and the data types used by the attributes.

2 Introduction to the Topology Model

The focus of this document is the parts of Core Network Model of the ONF-CIM that deal with Topology.

The Core Network Model encompasses all aspects of Topology. The focus of this document is:

- The basic topology model

© 2016 Open Networking Foundation

- Specific generalized forwarding properties of topology
- Recursive aggregation (reverse of partitioning) of topology
- Client-server layering of topology
- Views of topology and inter-view relationships[1]
- Abstraction of topology
- Off-network Links

Topology builds on aspects of the Core Network Model related to Termination and Forwarding described in TR-512.2 ONF Core IM - Forwarding and Termination. Topology capability and other specification considerations are covered in TR-512.7 ONF Core IM - Specification.

A data dictionary that sets out the details of all classes, data types and attributes is also provided (TR-512.8).

3 Topology model

3.1 Topology model overview

This section provides a high-level overview of the Topology model subset of the Core Network Model. The figure below provides a basic view of topology as a lightweight class diagram illustrating the key classes and focuses on the basic topology pattern. To avoid cluttering the figure, not all associations have been shown and all of the attributes were omitted.

The key object classes of the Topology model are ForwardingDomain (FD), Link, LinkPort and LogicalTerminationPoint (LTP). These entities are described in detail in section 3.2 Topology model detail on page 11.

---

[1] A topology view will normally be in a separate name space from the topology it is a view of. The view will normally not be associated with physical components (other than at its extreme edges via an association from the LTP to entities in the Physical Model). A topology view is essentially a view of virtual things. However, it should be noted that all entities in the Core Network Model (Link, FD, FC, LTP etc) are essentially representations of virtual things. By their very nature functional things are essentially emergent and virtual. Clearly to function they need underlying physical things but they do not need to have a known or a fixed association to the physical world. The Network Model supports associations to the Physical Model that need not be populated and that are NOT invariant and hence can change through the life of the entity.

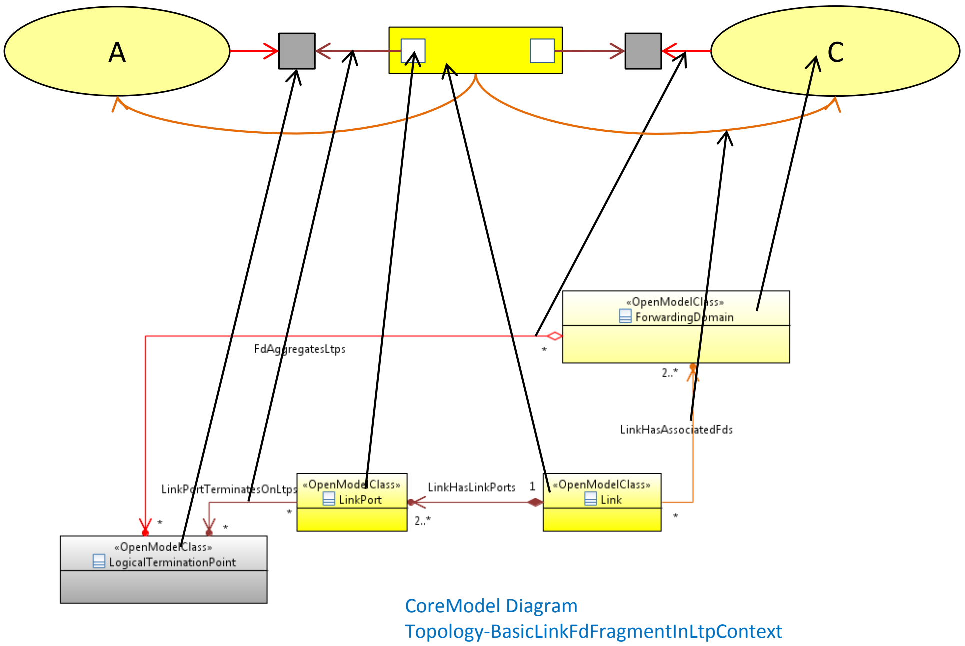
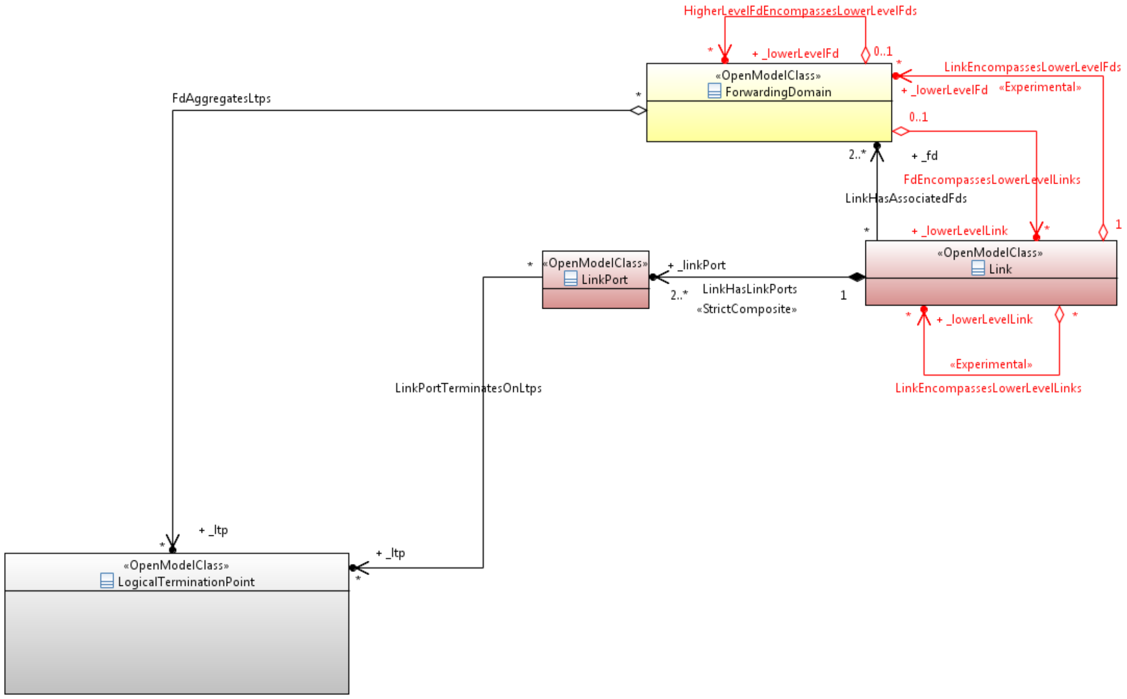


Figure 3-1 Basic topology view showing the model

In the figure below, the associations related to recursive aggregation, denoted in red, have been added to the basic topology pattern (note that the Link color scheme has changed from that used in the figure above to the alternative color for Link[2]). Explanation of this aspect of the model is provided in section 4.1 Basic Topology on page 23.

---

[2] There are two pictorial representations of link used in the documentation (one form is highly compact and the other form is expanded to emphasize the similarities between Link and FC). See TR-512.1 for the diagram symbol set.



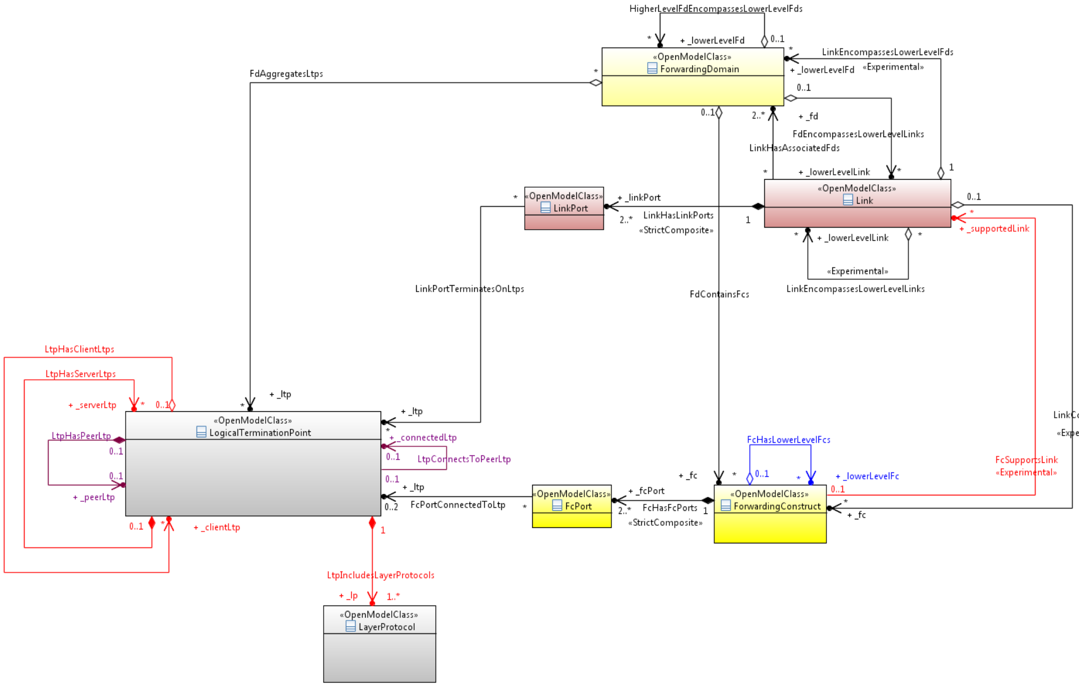
CoreModel diagram: Topology-AggregationSkeleton

Figure 3-2 Topology model highlighting aggregation

In the figure below, several associations have been added to model shown in the figure above. The added associations are related to:

- Layering, denoted in red, have been added to the topology with aggregation.
    - The key forwarding association is the FcSupportsLink.
- Aggregation of FCs, denoted in blue, that reflects the aggregation of FDs/Links
- Peer LTP fixed forwarding, denoted in purple.

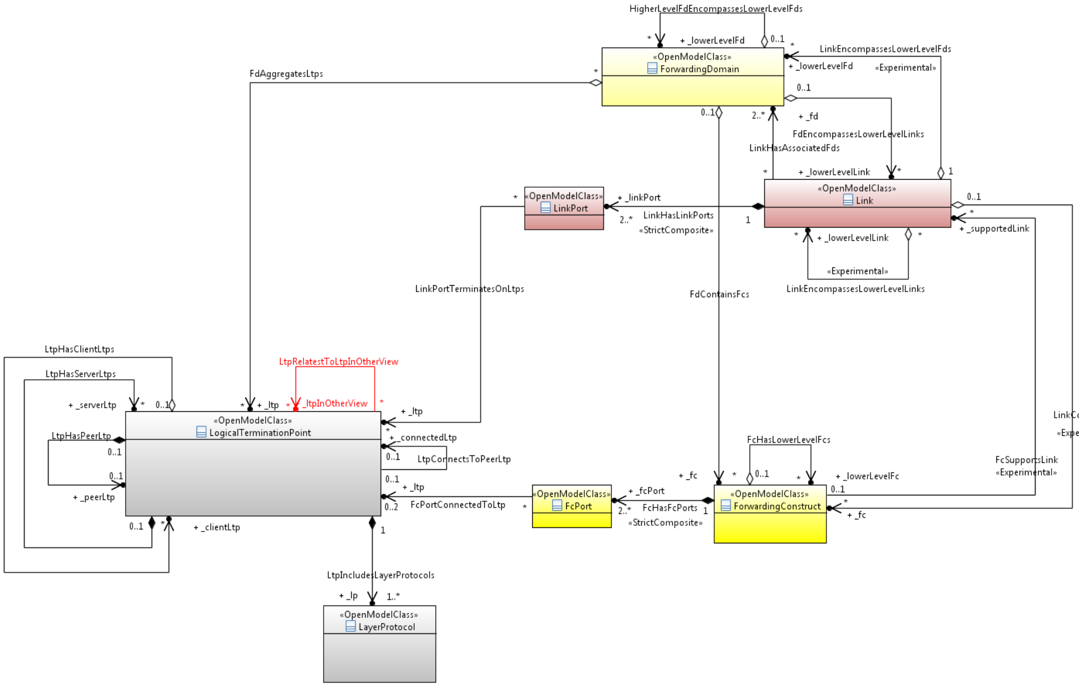
Explanation of this aspect of the model is provided in section 4.2 Topology and views on page 29.


HigherLevelFdEncompassesLowerLevelFds
«OpenModelClass» ForwardingDomain
LinkEncompassesLowerLevelFds «Experimental»
FdAggregatesLtps
FdEncompassesLowerLevelLinks
LinkHasAssociatedFds
«OpenModelClass» LinkPort
«OpenModelClass» Link
LinkHasLinkPorts «StrictComposite»
LinkEncompassesLowerLevelLinks «Experimental»
LinkPortTerminatesOnLtps
FdContainsFcs
LtpHasClientLtps
LtpHasServerLtps
LtpHasPeerLtp
«OpenModelClass» LogicalTerminationPoint
LtpConnectsToPeerLtp
FcPortConnectedToLtp
«OpenModelClass» FcPort
FcHasFcPorts «StrictComposite»
«OpenModelClass» ForwardingConstruct
FcHasLowerLevelFcs
LinkContainsFcs «Experimental»
FcSupportsLink «Experimental»
LtpIncludesLayerProtocols
«OpenModelClass» LayerProtocol
+ _lowerLevelFd
+ _fd
+ _lowerLevelLink
+ _linkPort
+ _supportedLink
+ _serverLtp
+ _peerLtp
+ _clientLtp
+ _ltp
+ _connectedLtp
+ _fcPort
+ _fc
+ _lowerLevelFc
+ _lp

CoreModel diagram: Topology-LayeredSkeleton

Figure 3-3 Topology model highlighting layering

In the figure below an association denoted in red, that supports inter-view navigation, has been added to the model shown in the figure above. Explanation of this aspect of the model is provided in section 4.2 Topology and views on page 29 and subsequet sections.


HigherLevelFdEncompassesLowerLevelFds
«OpenModelClass» ForwardingDomain
LinkEncompassesLowerLevelFds «Experimental»
FdAggregatesLtps
FdEncompassesLowerLevelLinks
LinkHasAssociatedFds
«OpenModelClass» LinkPort
«OpenModelClass» Link
LinkHasLinkPorts «StrictComposite»
LinkEncompassesLowerLevelLinks «Experimental»
LinkPortTerminatesOnLtps
FdContainsFcs
LtpHasClientLtps
LtpHasServerLtps
LtpRelatestToLtpInOtherView
_ltpInOtherView
LtpHasPeerLtp
«OpenModelClass» LogicalTerminationPoint
LtpConnectsToPeerLtp
FcPortConnectedToLtp
«OpenModelClass» FcPort
FcHasFcPorts «StrictComposite»
«OpenModelClass» ForwardingConstruct
FcHasLowerLevelFcs
LinkContainsFcs «Experimental»
FcSupportsLink «Experimental»
LtpIncludesLayerProtocols
«OpenModelClass» LayerProtocol
+ _lowerLevelFd
+ _fd
+ _lowerLevelLink
+ _linkPort
+ _supportedLink
+ _serverLtp
+ _peerLtp
+ _clientLtp
+ _ltp
+ _connectedLtp
+ _fcPort
+ _fc
+ _lowerLevelFc
+ _lp

CoreModel diagram: Topology-InterViewSkeleton

Figure 3-4 Basic topology showing layering

3.2 Topology model detail

The topology aspects of the key classes are covered in the following sub-sections.

Note that the classes show all attributes not just those associated with topology.

3.2.1 ForwardingDomain

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::ForwardingDomain

The ForwardingDomain (FD) class models the topological component that represents the opportunity to enable forwarding (of specific transport characteristic information at one or more protocol layers) between points represented by the LTP in the model. The FD object provides the context for and constrains the formation, adjustment and removal of FCs and hence offers the potential to enable forwarding. The LTPs available are those defined at the boundary of the FD. At a lower level of recursion an FD could represent a fabric (switch matrix) in a Network Element (NE). An NE can encompass more than one switch matrix and hence more than one FD. The FD representing a switch matrix can be further partitioned. The FD corresponds to a subnetwork [ITU-T G.800], FlowDomain [TMF 612] and a MultiLayerSubNetwork (MLSN) [TMF 612]. As in the TMF concept of MLSN and unlike the ITU-T concet of subnetwork model the FD can support more than one layer-protocol.

Inherits properties from:

- GlobalClass
- ForwardingEntity

Table 1: Attributes for ForwardingDomain

Attribute Name	**Lifecycle Stereotype (empty = Mature)**	**Description**
layerProtocolName		One or more protocol layers at which the FD represents the opportunity to enable forwarding between LTP that bound it.
_lowerLevelFd		The FD class supports a recursive aggregation relationship (HigherLevelFdEncompassesLowerLevelFds) such that the internal construction of an FD can be exposed as multiple lower level FDs and associated Links (partitioning). The aggregated FDs and Links form an interconnected topology that provides and describes the capability of the aggregating FD. Note that the model actually represents aggregation of lower level FDs into higher level FDs as views rather than FD partition, and supports multiple views. Aggregation 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).
_fc		An FD aggregares one or more FCs. A aggregated FC connects LTPs that bound the FD.
_ltp		An instance of FD is associated with zero or more LTP objects. The LTPs that bound the FD provide capacity for forwarding.

© 2016 Open Networking Foundation

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_lowerLevelLink		The FD encompasses Links that interconnect lower level FDs and collect links that are wholly within the bounds of the FD. See also _lowerLevelFd.
_fdRuleSet	Experimental	The rules related to an FD.
_layerProtocolParameterSpec		See referenced class

3.2.2 Link

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::Link

The Link class models effective adjacency between two or more ForwardingDomains (FD). 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 ports (LinkPort) 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).

Inherits properties from:

- GlobalClass
- ForwardingEntity

Table 2: Attributes for Link

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
layerProtocolName		The Link can support multiple transport layer protocols via the associated LTP object. 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. A link may support different layer protocols at each Port when it is a transitional link.
linkDirection		The directionality of the Link. Is applicable to simple Links where all LinkPorts are BIDIRECTIONAL (the Link will be BIDIRECTIONAL) or UNIDIRECTIONAL (the Link will be UNIDIRECTIONAL). Is not present in more complex cases.
isProtectionLockOut	Preliminary	The resource is configured to temporarily not be available for use in the protection scheme(s) it is part of. This overrides all other protection control states including forced. If the item is locked out then it cannot be used under any circumstances. Note: Only relevant when part of a protection scheme.
_fd		The Link associates with two or more FDs. This association provides a direct summarization of the association via LinkPort and LTP.
_linkPort		The association of the Link to LTPs is made via LinkPort (essentially the ports of the Link).

© 2016 Open Networking Foundation

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_lowerLevelLink	Experimental	A link may formed from subordinate links (similar FD formations from subordiate FDs). This association is intended to cover concepts such as serial compound links.
_fdRuleSet		The rules related to a Link.
_fc	Experimental	A Link contains one or more FCs. A contained FC connects LTPs that bound the Link. This FC represents the traditional LinkConnection. It is often not supported in implementations as it can be inferred from FCs in the corresponding FDs.
_lowerLevelFd	Experimental	FD(s) that form part of a serial compound link.
_forwardingSpec	Preliminary	See referenced class

3.2.3 LinkPort

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::LinkPort

The association of the Link to LTPs is made via LinkPort. The LinkPort class models the access to the Link function. The traffic forwarding between the associated LinkPorts of the Link depends upon the type of Link. In cases where there is resilience, the LinkPort may convey the resilience role of the access to the Link. The Link can be considered as a component and the LinkPort as a Port on that component

Inherits properties from:

- LocalClass

Table 3: Attributes for LinkPort

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
role		Each LinkPort of the Link has a role (e.g., symmetric, hub, spoke, leaf, root) in the context of the Link with respect to the Link function.
offNetworkAddress	Experimental	A freeform opportunity to express a reference for a Port of the Link that is not visible and hence is outside the scope of the control domain (off-network). This attribute is expected to convey a foreign identifier/name/address or a shared reference for some mid-span point at the boundary between two administrative domains. This is a reference shared between the parties either side of the network boundary. The assumption is that the provider knows the mapping between network port and offNetworkAddress and the client knows the mapping between the client port and the offNetworkAddress and that the offNetworkAddress references some common point or pool of points. It may represent some physical location where the hand-off takes place. This attribute is used when an LTP cannot be referenced. A Link with an Off-network end cannot be encompassed by an FD.
linkPortDirection		The orientation of defined flow at the LinkPort.

© 2016 Open Networking Foundation

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_ltp		The LinkPort may be associated with more than one LTP when the LinkPort is bidirectional and the LTPs are unidirectional. Multiple Ltp - Bidirectional LinkPort to two Uni Ltps Zero Ltp - BreakBeforeMake transition - Planned Ltp not yet in place - Off-network LTP referenced through other mechanism

3.2.4 LogicalTerminationPoint

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::LogicalTerminationPoint

The LogicalTerminationPoint (LTP) class encapsulates the termination and adaptation functions of one or more transport layers represented by instances of LayerProtocol. The encapsulated transport layers have a simple fixed 1:1 client-server relationship defined by association end ordering. The structure of LTP supports all transport protocols including circuit and packet forms.

Inherits properties from:

- GlobalClass

Table 4: Attributes for LogicalTerminationPoint

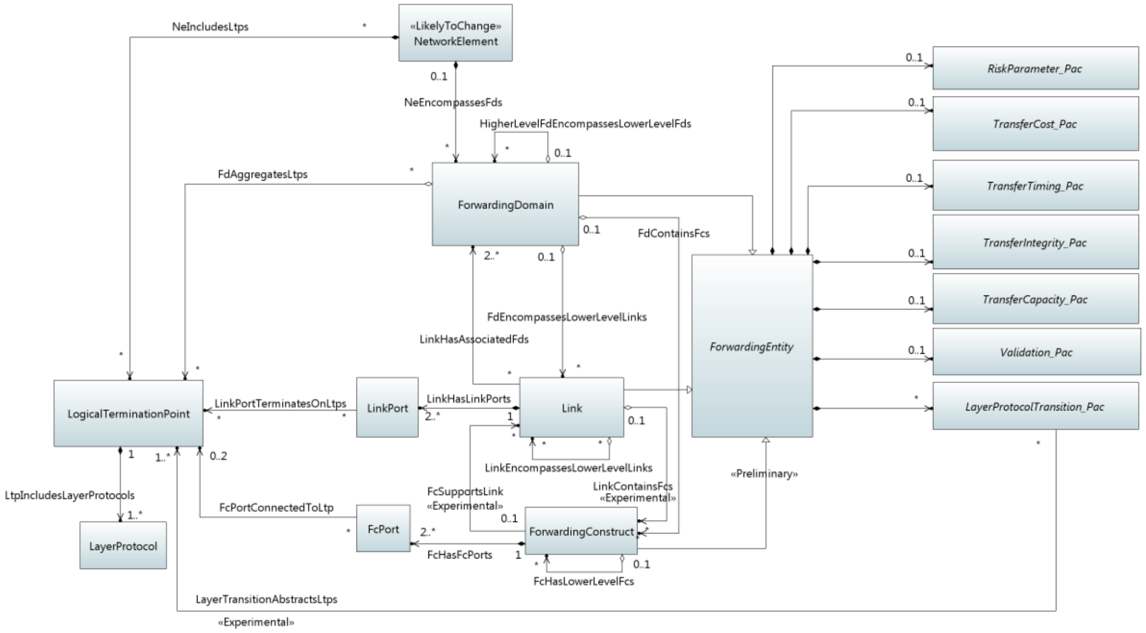
Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
physicalPortReference	Preliminary	One or more text labels for the unmodelled physical port associated with the LTP. In many cases there is no associated physical port.
ltpDirection		The overall directionality of the LTP. - A BIDIRECTIONAL LTP must have at least some LPs that are BIDIRECTIONAL but may also have some SINK and/or SOURCE LPs. - A SINK LTP can only contain SINK LPs - A SOURCE LTP can only contain SOURCE LPs
_serverLtp		References contained LTPs representing servers of this LTP in an inverse multiplexing configuration (e.g. VCAT).
_clientLtp		References contained LTPs representing client traffic of this LTP for normal cases of multiplexing.
_lp		Ordered list of LayerProtocols that this LTP is comprised of where the first entry in the list is the lowest server layer (e.g. physical).
_connectedLtp		Applicable in a simple context where two LTPs are associated via a non-adjustable enabled forwarding. Reduces clutter removing the need for two additional LTPs and an FC with a pair of FcPorts.
_peerLtp		References contained LTPs representing the reversal of orientation of flow where two LTPs are associated via a non-adjustable enabled forwarding and where the referenced LTP is fully dependent on the this LTP.

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_ltpSpec	Experimental	The specification of the LTP defines internal structure of the LTP. The specification allows interpretation of organisation of LPs making up the LTP and also identifies which inter-LTP associations are valid.
_ltpInOtherView	Preliminary	References one or more LTPs in other views that represent this LTP. The referencing LTP is the provider of capability.
_port	Experimental	See referenced class

3.3 Topology model classes, related classes and structures

The classes used to represent topology are summarized in the following figure. The figure highlights that the Link, ForwardingDomain and ForwardingConstruct inherit common topology related properties from ForwardingEntity. The figure also highlights that the ForwardingConstruct inherits from ForwardingEntity[3]. This ForwardingEntity is used as a modeling approach to apply specific packages of attributes to Link, ForwardingDomain and ForwardingConstruct via inheritance.

Some associations not related to the focus of this document are omitted.



CoreModel diagram: Topology-HighLevelOverviewOfStructureAndPacs-LargeText

Figure 3-5 Key classes that form the network topology

---

[3] ForwardingEntity used to be called TopologicalEntity in previous releases. The name change reflects the broader application.

The figure above focuses on interrelationships and shows that:

- An FD may be a subordinate part of a NetworkElement, may coincide with a NetworkElement or may be larger than, and independent of, any NetworkElement (See for example FDs A.1 and A.3 in Figure 4-2 ForwardingDomain recursion with link and NetworkElement on page 26).
- An FD may encompass lower level FDs. This may be such that:
    - An FD directly contained in a NetworkElement is divided into smaller parts
    - An FD not encompassed by a NetworkElement is divided into smaller parts some of which may be encompassed by NetworkElements (see Figure 4-2 ForwardingDomain recursion with link and NetworkElement on page 26)
    - The FD represents the whole network

  Note that an FD at the lowest level of abstraction (a fabric or some piece of 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[4]
        - Note that it is usual for a Link to interconnect 2 FDs but there are cases where many FDs may be interconnected by a Link.

- A Link has LinkPorts that represent the accesses to the Link itself
    - LinkPorts are especially relevant for multi-ended asymmetric Link
- An FD aggregates LogicalTerminationPoints (LTPs) that bound it. An LTP represent a stack of layer-protocol terminations, where the details of each is held in the LayerProtocol (LP). An LTP may be:
    - Part of a NetworkElement
    - Conceptually independent from any NetworkElement[5]
- A Link terminates on LTPs via its contained LinkPorts.

---

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

[5] The assumption is that the LTP can be floating (representing a pool) in the context of a network as a whole (represented by an FD). The LTP has a UUID to allow it to be identified. Under these circumstances it does not need to be contained in anything (although it is pooled by the FD representing the network). It clearly does need to be accessible via a controller.

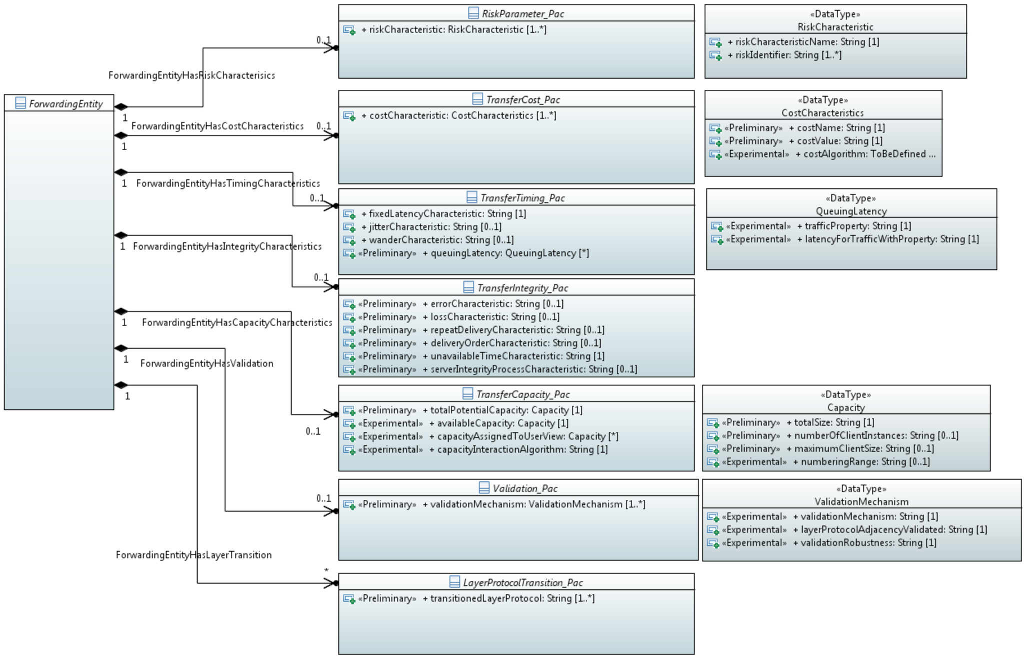
The figure also highlights the relationships between FC and Link:

- A Link can be (known to be) supported by an FC in a server layer
  - "All" Links are supported by Forwarding in a server layer. A link is an abstraction of underlying forwarding
  - The Link topology is essentially an abstraction of the layout of supporting FCs
- A Link can give rise to FCs that represent the forwarding in the Link in the layer of the link
  - Note that the term "topology" is used predominantly in the context of layout of available capacity. Although the FC layout, usage of capacity, could also considered as a Topology, this is not a usual usage of the term

3.4 Topological properties of the ForwardingEntity

The ForwardingEntity brings attributes related to transfer characteristics and other forwarding considerations, these attributes are elaborated below. For further details of types etc refer to the TR-512.8 ONF Core IM - Data Dictionary.

The figure below shows the _Pacs in more detail.



ForwardingEntity
RiskParameter_Pac
+ riskCharacteristic: RiskCharacteristic [1..*]
«DataType» RiskCharacteristic
+ riskCharacteristicName: String [1]
+ riskIdentifier: String [1..*]
ForwardingEntityHasRiskCharacterisics
1
0..1
ForwardingEntityHasCostCharacteristics
1
0..1
TransferCost_Pac
+ costCharacteristic: CostCharacteristics [1..*]
«DataType» CostCharacteristics
«Preliminary» + costName: String [1]
«Preliminary» + costValue: String [1]
«Experimental» + costAlgorithm: ToBeDefined ...
ForwardingEntityHasTimingCharacteristics
1
0..1
TransferTiming_Pac
+ fixedLatencyCharacteristic: String [1]
+ jitterCharacteristic: String [0..1]
+ wanderCharacteristic: String [0..1]
«Preliminary» + queuingLatency: QueuingLatency [*]
«DataType» QueuingLatency
«Experimental» + trafficProperty: String [1]
«Experimental» + latencyForTrafficWithProperty: String [1]
ForwardingEntityHasIntegrityCharacteristics
1
0..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]
ForwardingEntityHasCapacityCharacteristics
1
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]
ForwardingEntityHasValidation
1
0..1
Validation_Pac
«Preliminary» + validationMechanism: ValidationMechanism [1..*]
«DataType» ValidationMechanism
«Experimental» + validationMechanism: String [1]
«Experimental» + layerProtocolAdjacencyValidated: String [1]
«Experimental» + validationRobustness: String [1]
ForwardingEntityHasLayerTransition
1
*
LayerProtocolTransition_Pac
«Preliminary» + transitionedLayerProtocol: String [1..*]

CoreModel diagram: Topology-Detailed_PacProperties

Figure 3-6 Topology _Pac detail

© 2016 Open Networking Foundation

As shown in the figure above, an abstract class "ForwardingEntity" has been defined to collect forwarding/topology-related properties (characteristics, etc.) that are common for FC, FD and Link. The FC, FD and Link can acquire contents from the conditional packages (_Pacs). The conditional packages provide all key forwarding properties of a topology.

Note that a number of areas are still under development (highlighted using «Experimental» or «Preliminary»).

3.4.1 ForwardingEntity

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::Topology::ForwardingEntity

A ForwardingEntity 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 ForwardingEntity. The ForwardingEntity 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.

This class is abstract.

Table 5: Attributes for ForwardingEntity

Attribute Name	**Lifecycle Stereotype (empty = Mature)**	**Description**
_riskParameter_Pac		See referenced class
_transferCost_Pac		See referenced class
_transferTiming_Pac		See referenced class
_transferCapacity_Pac		See referenced class
_transferIntegrity_Pac		See referenced class
_validation_Pac		See referenced class
_layerTransition_Pac		See referenced class

3.4.2 LayerProtocolTransition_Pac

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::Topology::LayerProtocolTransition_Pac

The transition characteristics are 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 needs to be accounted for (this is derived from the LTP spec details). This Pac provides the relevant abstractions of the LTPs and

© 2016 Open Networking Foundation

provides the necessary association to the LTPs involved. Links that include details in this Pac are often referred to as Transitional Links.

This class is abstract.

Table 6: Attributes for LayerProtocolTransition_Pac

Attribute Name	**Lifecycle Stereotype (empty = Mature)**	**Description**
transitionedLayerProtocol	Preliminary	Provides the ordered structure of layer protocol transitions encapsulated in the ForwardingEntity. The ordering relates to the LinkPort role.
_ltp	Experimental	Lists the LTPs that define the layer protocol transition of the transitional link.

3.4.3 RiskParameter_Pac

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::Topology::RiskParameter_Pac

The risk characteristics of a ForwardingEntity 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 ForwardingEntity 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 ForwardingEntity that uses any part of that larger physical/geographical unit will suffer impact and hence each ForwardingEntity shares risk. The identifier of each physical/geographical unit that is involved in the realization of each segment of a Forwarding entity can be listed in the RiskParameter_Pac of that ForwardingEntity. A segment has one or more risk characteristic. Shared risk between two TopologicalEntities compromises the integrity of any solution that use one of those ForwardingEntity 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.

This class is abstract.

Table 7: Attributes for RiskParameter_Pac

Attribute Name	**Lifecycle Stereotype (empty = Mature)**	**Description**
riskCharacteristic		A list of risk characteristics for consideration in an analysis of shared risk. Each element of the list represents a specific risk consideration.

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description

3.4.4 TransferCapacity_Pac

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::Topology::TransferCapacity_Pac

The ForwardingEntity derives capacity from the underlying realization. A ForwardingEntity 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 ForwardingEntity may be directly used in the view or may be assigned to another view for use. The clients supported by a multi-layer ForwardingEntity may interact such that the resources used by one client may impact those available to another. This is derived from the LTP spec details. Represents the capacity available to user (client) along with client interaction and usage. A ForwardingEntity may reflect one or more client protocols and one or more members for each profile.

This class is abstract.

Table 8: Attributes for TransferCapacity_Pac

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
totalPotentialCapacity	Preliminary	An optimistic view of the capacity of the ForwardingEntity assuming that any shared capacity is available to be taken.
availableCapacity	Experimental	Capacity available to be assigned.
capacityAssignedToUserView	Experimental	Capacity already assigned.
capacityInteractionAlgorithm	Experimental	A reference to an algorithm that describes how various chunks of allocated capacity interact (e.g. when shared).

3.4.5 TransferCost_Pac

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::Topology::TransferCost_Pac

The cost characteristics of a ForwardingEntity not necessarily correlated to the cost of the underlying physical realization. They may be quite specific to the individual ForwardingEntity (e.g. opportunity cost) and relates to layer capacity There may be many perspectives from which cost may be considered for a particular ForwardingEntity and hence many specific costs and potentially cost algorithms. Using an entity will incur a cost.

This class is abstract.

Table 9: Attributes for TransferCost_Pac

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
costCharacteristic		The list of costs where each cost relates to some aspect of the ForwardingEntity.

3.4.6 TransferIntegrity_Pac

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::Topology::TransferIntegrity_Pac

Transfer integrity characteristic covers expected/specified/acceptable characteristic of degradation of the transferred signal. It includes all aspects of possible degradation of signal content as well as any damage of any form to the total ForwardingEntity and to the carried signals. Note that the statement is of total impact to the ForwardingEntity so any partial usage of the ForwardingEntity (e.g. a signal that does not use full capacity) will only suffer its portion of the impact.

This class is abstract.

Table 10: Attributes for TransferIntegrity_Pac

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
errorCharacteristic	Preliminary	Describes the degree to which the signal propagated can be errored. Applies to TDM systems as the errored signal will be propagated and not packet as errored packets will be discarded.
lossCharacteristic	Preliminary	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	Preliminary	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	Preliminary	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	Preliminary	Describes the duration for which there may be no valid signal propagated.
serverIntegrityProcessCharacteristic	Preliminary	Describes the effect of any server integrity enhancement process on the characteristics of the ForwardingEntity.

3.4.7 TransferTiming_Pac

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::Topology::TransferTiming_Pac

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

This class is abstract.

Table 11: Attributes for TransferTiming_Pac

Attribute Name	**Lifecycle Stereotype (empty = Mature)**	**Description**
fixedLatencyCharacteristic		A ForwardingEntity 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 ForwardingEntity.
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).
wanderCharacteristic		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).
queuingLatency	Preliminary	The effect on the latency of a queuing process. This only has significant effect for packet based systems and has a complex characteristic.

3.4.8 Validation_Pac

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::Topology::Validation_Pac

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

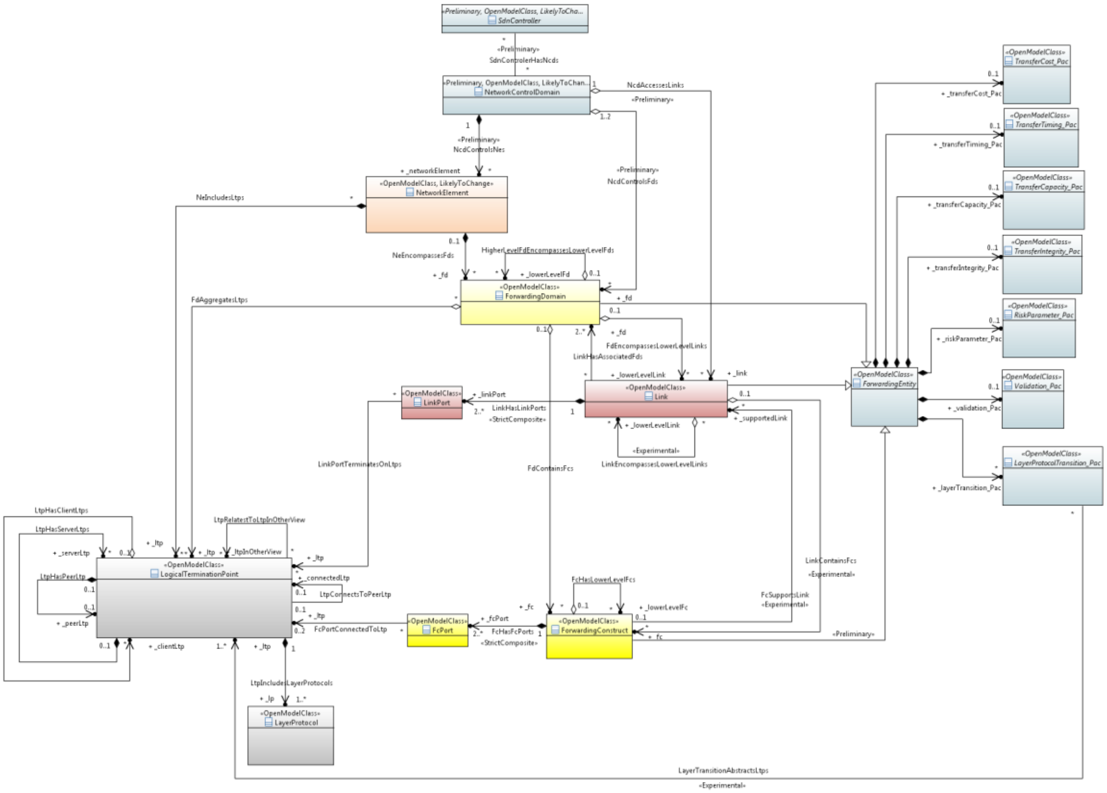
This class is abstract.

Table 12: Attributes for Validation_Pac

Attribute Name	**Lifecycle Stereotype (empty = Mature)**	**Description**
validationMechanism	Preliminary	Provides details of the specific validation mechanism(s) used to confirm the presence of an intended ForwardingEntity.

3.5 Model showing topology, forwarding and termination

The figure below shows the topology model in the context of the Core Network Model.



CoreModel diagram: Topology-FullSkeleton

Figure 3-7 Topology, Forwarding and Termination

4 Explanatory figures

This section provides figures that illustrate the application of the model to various network scenarios. The section builds up from simple topologies to complex multi-layer schemes observed from different viewpoints.

For an explanation of the symbol set being used in the figures see section 1.3 Conventions on page 6.

4.1 Basic Topology

In this section a basic stylized network example is used to illustrate some of the associations in the Topology model fragment. 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.

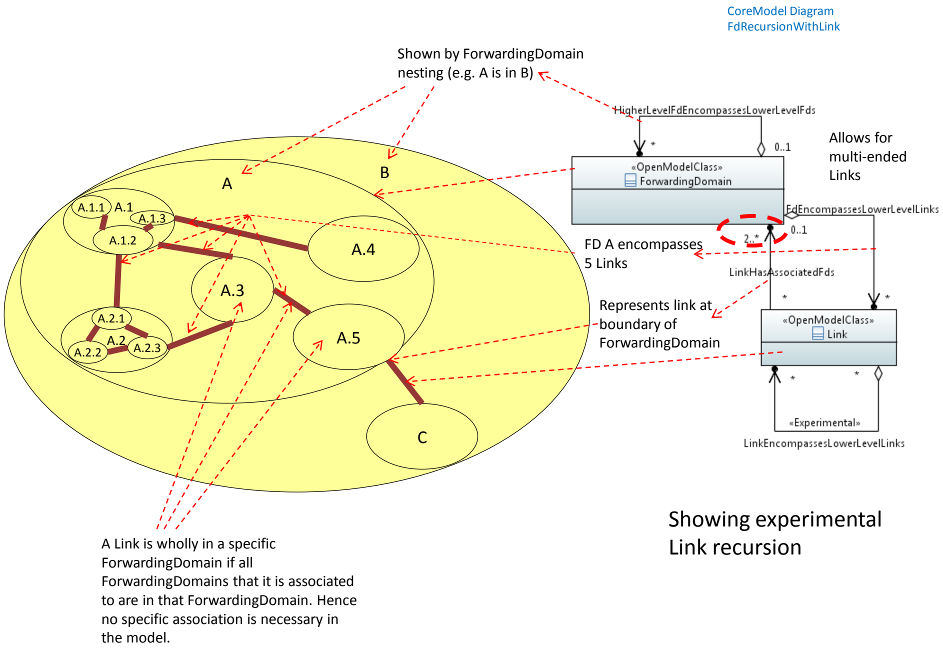

CoreModel Diagram
FdRecursionWithLink
Shown by ForwardingDomain nesting (e.g. A is in B)
HigherLevelFdEncompassesLowerLevelFds
*
0..1
Allows for multi-ended Links
«OpenModelClass»
ForwardingDomain
FdEncompassesLowerLevelLinks
2..*
0..1
FD A encompasses 5 Links
LinkHasAssociatedFds
*
*
Represents link at boundary of ForwardingDomain
«OpenModelClass»
Link
*
*
«Experimental»
LinkEncompassesLowerLevelLinks
A
B
A.1.1
A.1
A.1.3
A.1.2
A.4
A.3
A.2.1
A.2
A.2.2
A.2.3
A.5
C
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.
Showing experimental Link recursion

Figure 4-1 ForwardingDomain recursion with Link[6]

The figure above 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 layer-protocol although an FD can support a list of layer-protocols.

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 in the figure above, 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

---

[6] The numbering on the figure implies strict and fixed hierarchy. It should be noted that the association is aggregation and hence the hierarchy can change and an FD may move from being encompassed by one FD to being encompassed by another. Consider the numbering as simply a view of the current structure.

- A.2 encompasses three FDs: A.2.1, A.2.2 and A.2.3

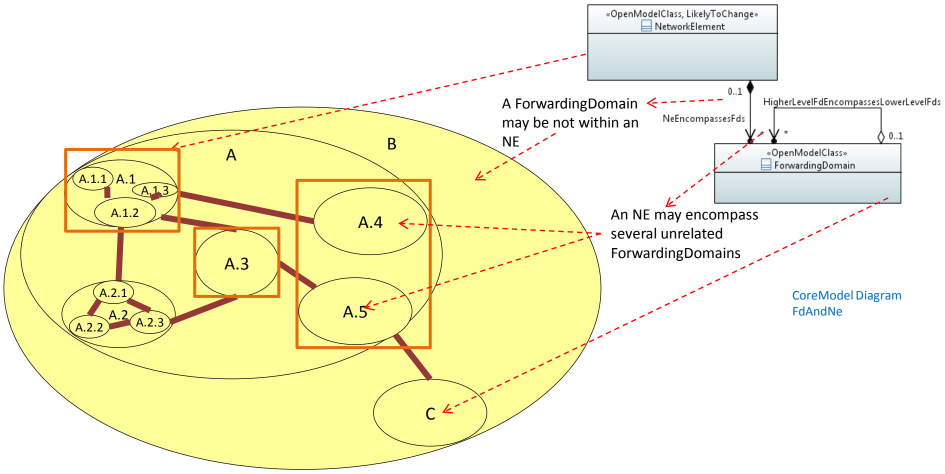
When one FD is removed, the "HigherLevelFdEncompassesLowerLevelFds" relationships are modified. For example, if FD A.1 in Figure 4-2 is removed, the instances of the "HigherLevelFdEncompassesLowerLevelFds" relationships will be modified as follows:

- 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[7]
- 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 Figure 4-3 ForwardingDomain, Link and LTP associations on page 27). 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.[8]

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


«OpenModelClass, LikelyToChange»
NetworkElement
0..1
NeEncompassesFds
HigherLevelFdEncompassesLowerLevelFds
*
*
0..1
«OpenModelClass»
ForwardingDomain
A ForwardingDomain may be not within an NE
An NE may encompass several unrelated ForwardingDomains
CoreModel Diagram FdAndNe
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

---

[7] Clearly the FD naming in the figure is for ease of reading the diagram and does not represent hierarchy.

[8] Work supporting this was liaised from TM Forum.

Figure 4-2 ForwardingDomain recursion with link and NetworkElement

The figure 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.

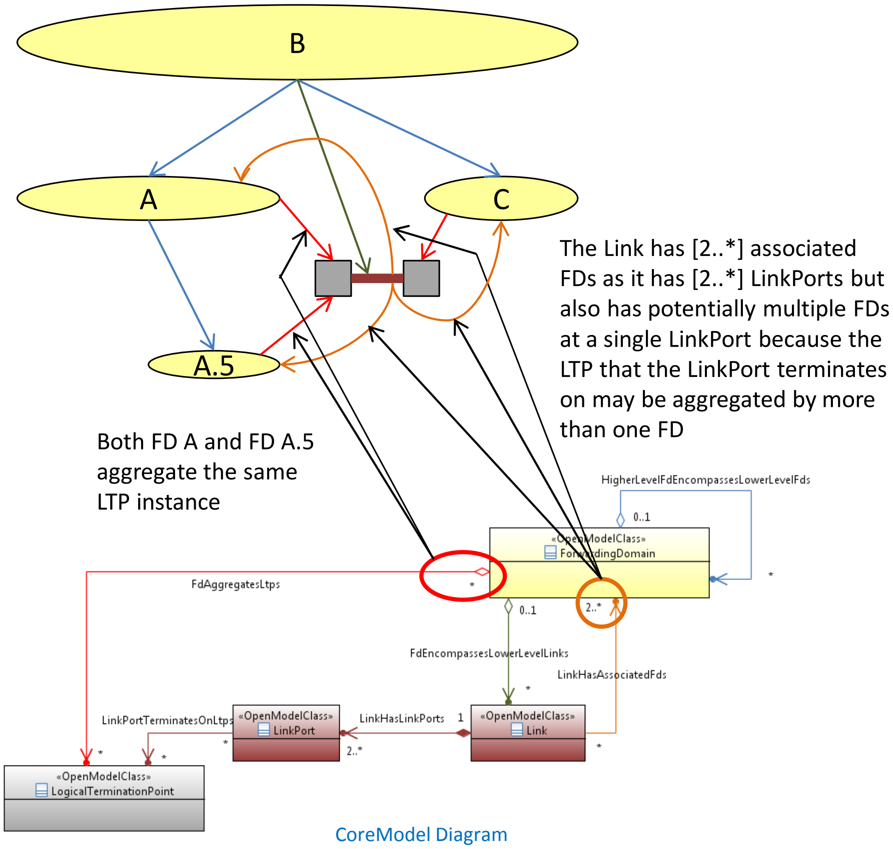
The figure emphasizes that at and below one particular level of abstraction of ForwardingDomain, the ForwardingDomains are all bounded by a specific NetworkElement (brown square). This is represented in the UML fragment by the composition association (black diamond) that explains that there is a lifecycle 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 a NetworkElement (as explained in the UML fragment by the 0..1 composition), and that a ForwardingDomain may have a smaller scope than the whole NetworkElement (even when considering only a single layer-protocol 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.[9]

The following figure zooms in on a fragment of the network used in previous figures. The figure shows detail of the LinkPort and LTP (intentionally omitted from the earlier figures). The key points are highlighted in the figure.

---

[9] 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.



The Link has [2..*] associated FDs as it has [2..*] LinkPorts but also has potentially multiple FDs at a single LinkPort because the LTP that the LinkPort terminates on may be aggregated by more than one FD

Both FD A and FD A.5 aggregate the same LTP instance

CoreModel Diagram
LinkFdFragmentInLtpContext

Figure 4-3 ForwardingDomain, Link and LTP associations

An alternative way of depicting the topology of the example is shown in the next figure. The link shown in the figure above is shown twice in the next figure as highlighted in the figure.

© 2016 Open Networking Foundation

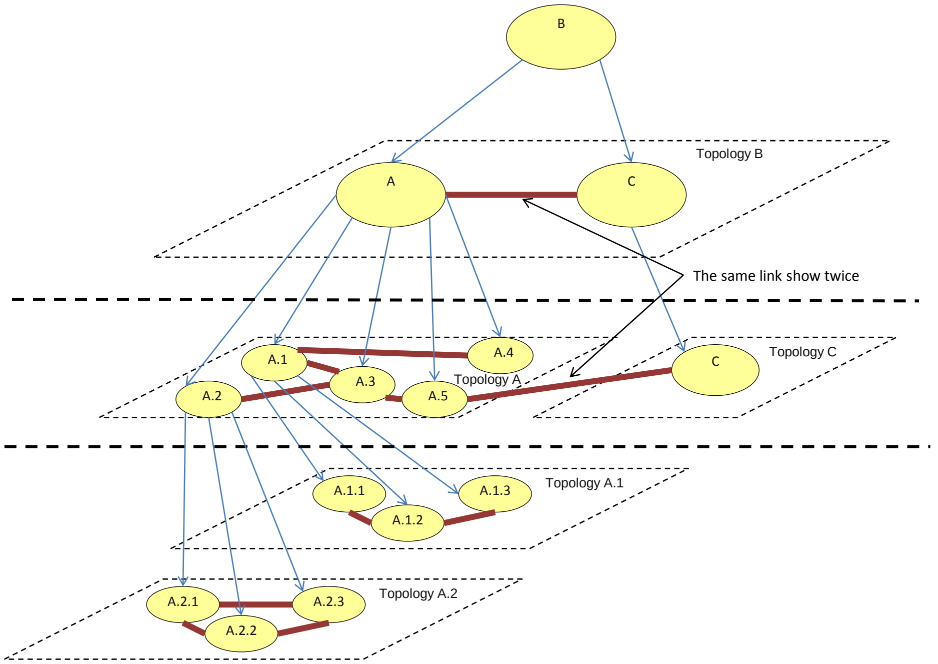
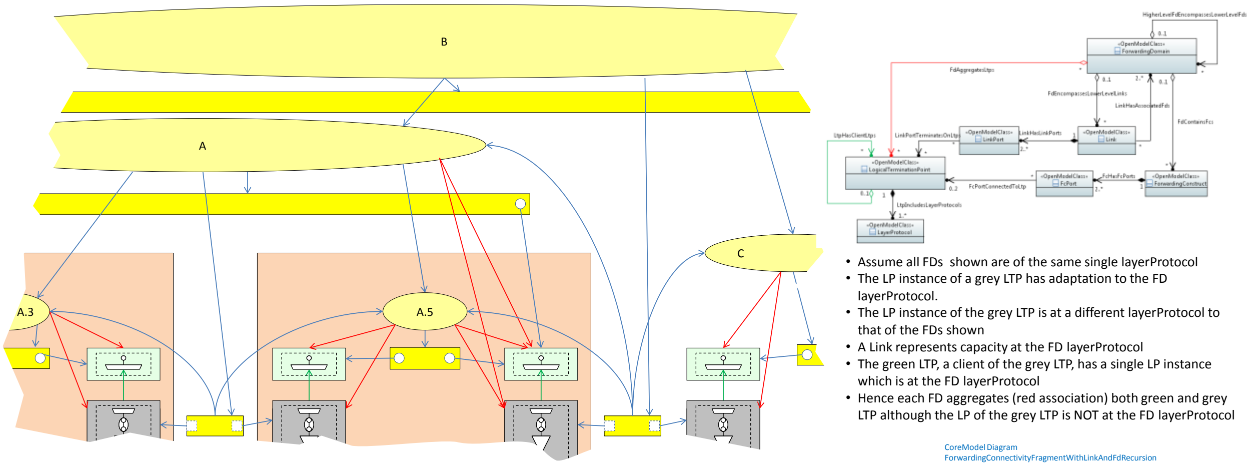


Figure 4-4 FDs and Topology

The following figure considers the topology further in combination with the FCs. The figure shows LTP aggregated in two (or more) FDs and highlights that the FD may be of different layer-protocols than the LTP. The LTP will be aggregated by an FD if a LayerProtocol of the LTP:

- Is for the same layer-protocol of the FD (and the LTP is on the FD boundary)
- Adapts to the layer-protocol of the FD (and the LTP is on the FD boundary)

Note that the figure shows an NE context to assist in understanding that the principle also applies in a fully "virtualized" case (simply remove the NE boxes).

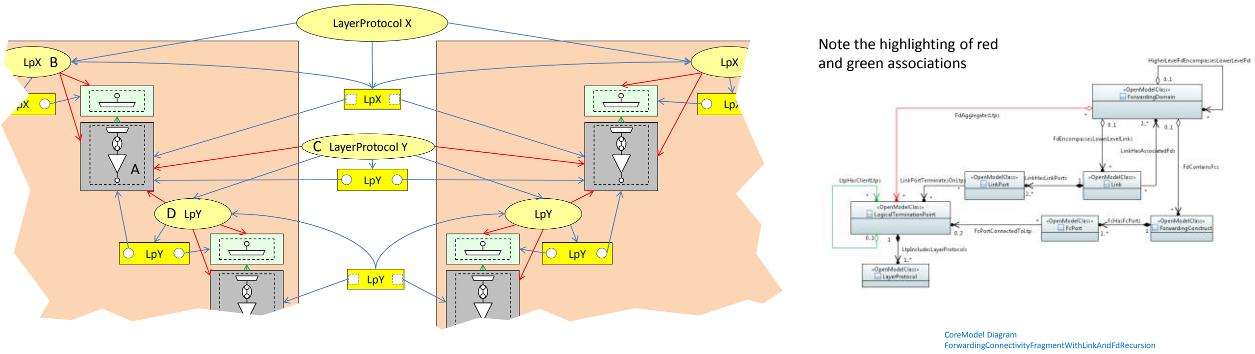


- Assume all FDs shown are of the same single layerProtocol
- The LP instance of a grey LTP has adaptation to the FD layerProtocol.
- The LP instance of the grey LTP is at a different layerProtocol to that of the FDs shown
- A Link represents capacity at the FD layerProtocol
- The green LTP, a client of the grey LTP, has a single LP instance which is at the FD layerProtocol
- Hence each FD aggregates (red association) both green and grey LTP although the LP of the grey LTP is NOT at the FD layerProtocol

CoreModel Diagram
ForwardingConnectivityFragmentWithLinkAndFdRecursion

Figure 4-5 LTPs Encompassed by FDs (at one layer-protocol)

Clearly an LTP with multiple LPs may be aggregated by FDs at multiple layer-portocols. An LTP may be aggregated by FDs of different layer-protocols even where the LTP only has one LP. The figure below shows a case where there is a floating LTP, A, containing a single LP where the LTP is aggregated in FD B at layer-protocol X and FD C and FD D at layer-protocol Y.



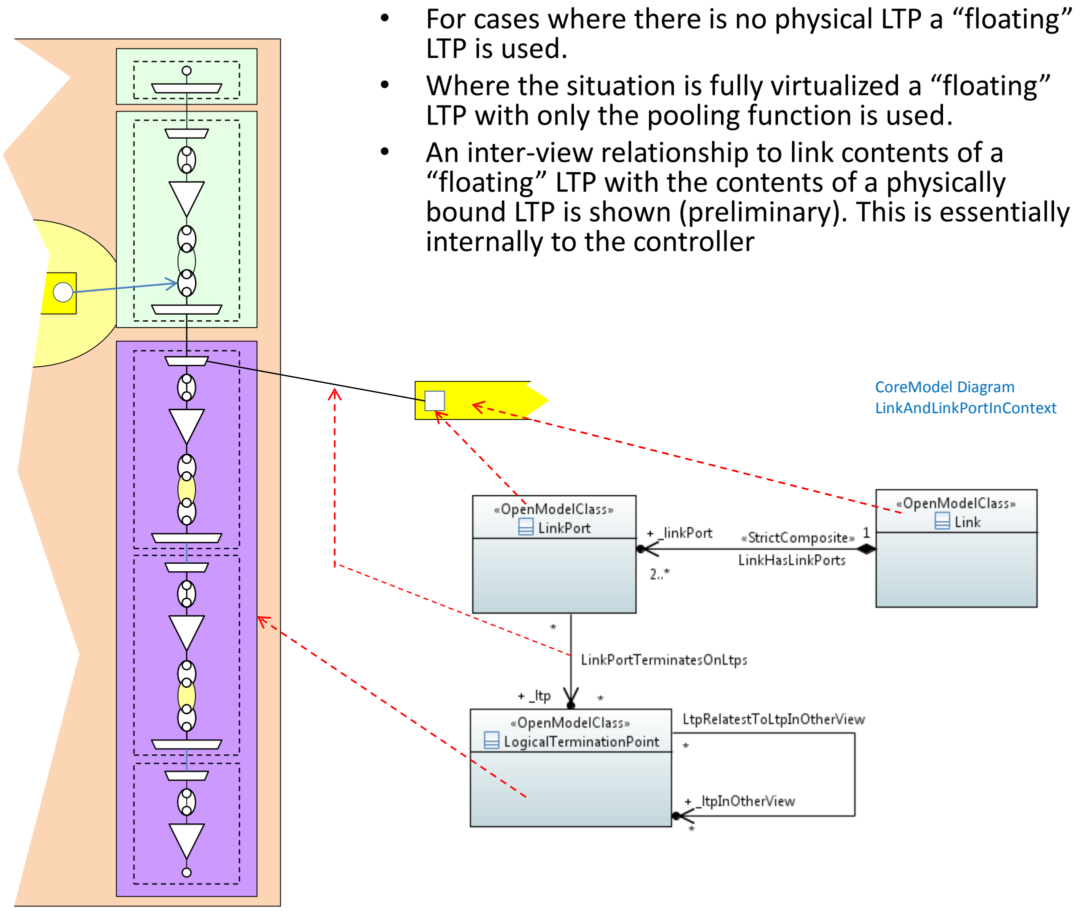
Note the highlighting of red and green associations

CoreModel Diagram
ForwardingConnectivityFragmentWithLinkAndFdRecursion

Figure 4-6 LTPs Encompassed by FDs (at several layer-protocols)

4.2 Topology and views

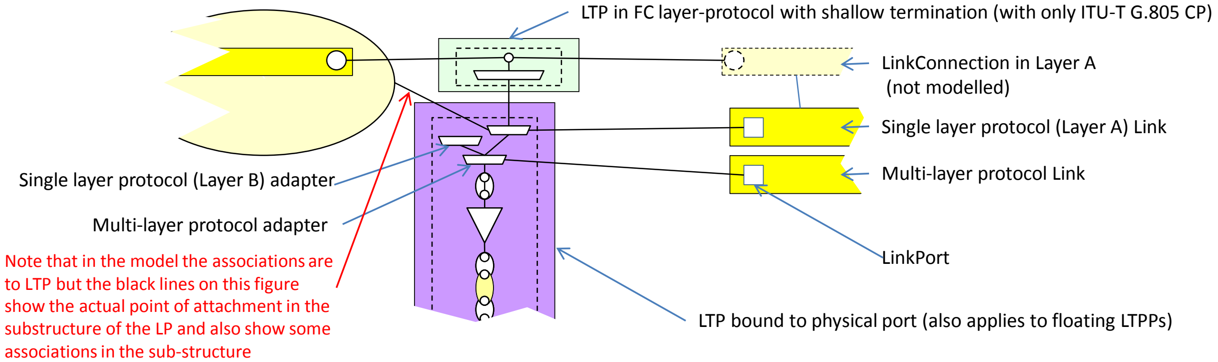
© 2016 Open Networking Foundation



- 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

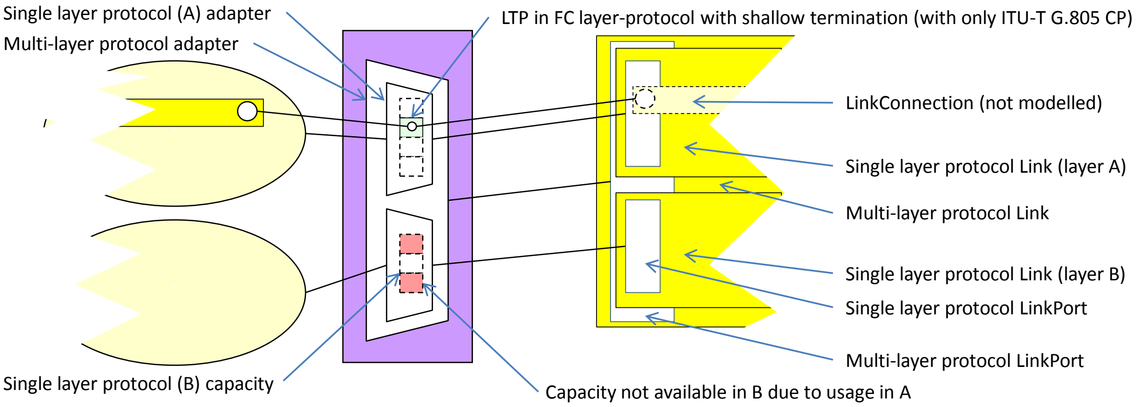
Figure 4-7 LTP "pooling" client LTPs

Figure 6-16 above shows how the Link terminates on the LTP via the LinkPort.


LTP in FC layer-protocol 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
LinkPort
LTP bound to physical port (also applies to floating LTPPs)

Note that in the model the associations are to LTP but the black lines on this figure show the actual point of attachment in the substructure of the LP and also show some associations in the sub-structure

Showing layering in elevation view (above)


Single layer protocol (A) adapter
Multi-layer protocol adapter
LTP in FC layer-protocol 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 LinkPort
Multi-layer protocol LinkPort
Single layer protocol (B) capacity
Capacity not available in B due to usage in A

Showing multiple channels in plan view (above)

Figure 4-8 Views of Link, LinkPort and LTP showing LTP pooling

The LTP may have the capability[10] 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.

---

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

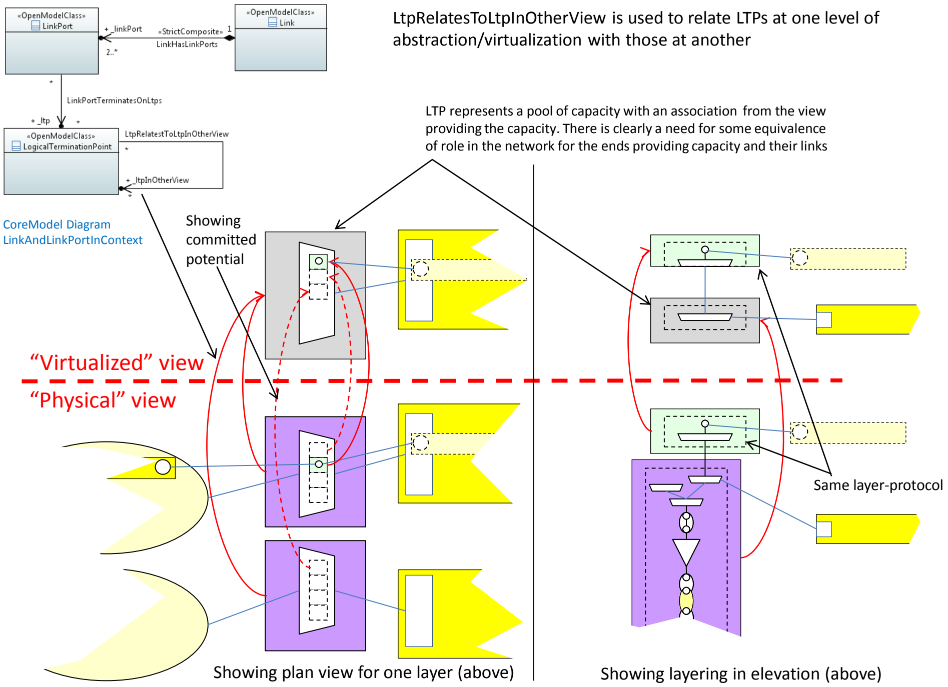

«OpenModelClass»
LinkPort
+ _linkPort
«StrictComposite»
1
LinkHasLinkPorts
2..*
«OpenModelClass»
Link
*
LinkPortTerminatesOnLtps
+ _ltp
*
«OpenModelClass»
LogicalTerminationPoint
LtpRelatestToLtpInOtherView
*
+ _ltpInOtherView
*
CoreModel Diagram
LinkAndLinkPortInContext
LtpRelatesToLtpInOtherView is used to relate LTPs at one level of abstraction/virtualization with those at another
LTP represents a pool of capacity with an association from the view providing the capacity. There is clearly a need for some equivalence of role in the network for the ends providing capacity and their links
Showing committed potential
"Virtualized" view
"Physical" view
Same layer-protocol
Showing plan view for one layer (above)
Showing layering in elevation (above)

Figure 4-9 Views of "virtualization" [11] of LTPs with server side LTP representing a pool

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.

It is important to emphasize that the Virtual/Physical split is a gross simplification. In reality a server provides an abstracted/virtualized view of an underlying system to its client where that underlying system is provided by further servers hence "Physical" view obscures this complexity (but is sufficient for this description).

- Both views are virtualized where the lower view is "providing" to the upper view
- Using the term "physical" at this point is tolerable as it enables easier case oriented interpretation of the figures and concepts.
- Something like "provider's resource context" and "provider's client view context" may be better terms in the long run

The figures below provide a view of sequence of realization of a virtualized view.

The first figure provides a starting position. The figure depicts a virtualized view and a completely disassociated physical view. At this point although the network does exist it has no capacity allocated to the client or used in any way.

---

[11] The terms "physical", "virtualization" and "virtualized" are used loosely here. The "physical" aspects are shown in the context of LTPs bound to physical but in general this is really the "provider view" and the "virtualized" aspects are really "provider's client view context" (which is essentially what the provider exposes to the client".

However, the client has been offered some pre-planned resources and has chosen usage of some resources. These resources are clearly not operable. This is analogous to pre-provisioning an equipment slot in an NE.

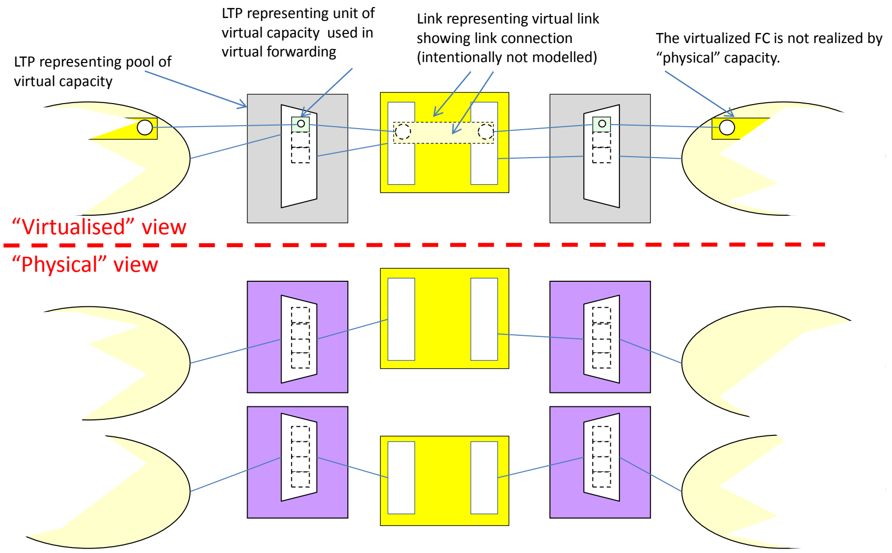

LTP representing pool of virtual capacity
LTP representing unit of virtual capacity used in virtual forwarding
Link representing virtual link showing link connection (intentionally not modelled)
The virtualized FC is not realized by “physical” capacity.
“Virtualised” view
“Physical” view

Figure 4-10 Starting condition

The operator then chooses to allocate capacity is allocated by the provider from the "Physical" view to the "Virtualized" view. Note that the association "LtpRelatesToLtpInOtherView" is from the "Physical" view to the "Virtualized" view and is from both levels of LTP (LayerProtocol Client and LayerProtocol Server). This orientation emphasizes that real resources are provided and that the actual client will not see anything other than the virtualized view. Clearly in some places in an actual solution realization both directions of association may be beneficial.

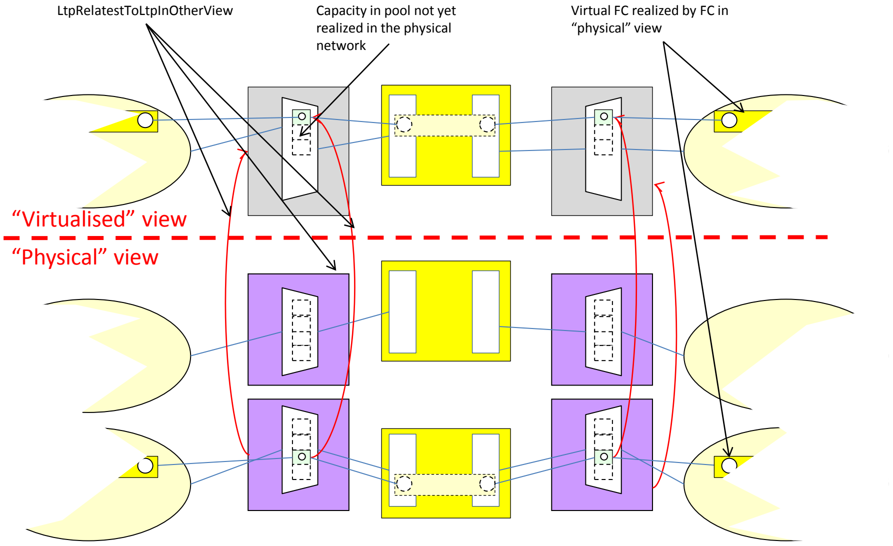

LtpRelatestToLtpInOtherView
Capacity in pool not yet realized in the physical network
Virtual FC realized by FC in “physical” view
“Virtualised” view
“Physical” view

Figure 4-11 Resource allocation

The figure illustrates that there is no necessary ordering/numbering consistency between the "Physical" view and the "Virtualized" view.

At some future point the provider my decide to reallocate resources in the network such that the "Virtualized" view LTP is now supported by a different "Physical" view LTP (as shown in the figure below). Clearly there are various sequencing considerations to minimize impact. The essential thing to note is that the naming/addressing in the "Virtualized" view is unchanged through the process.

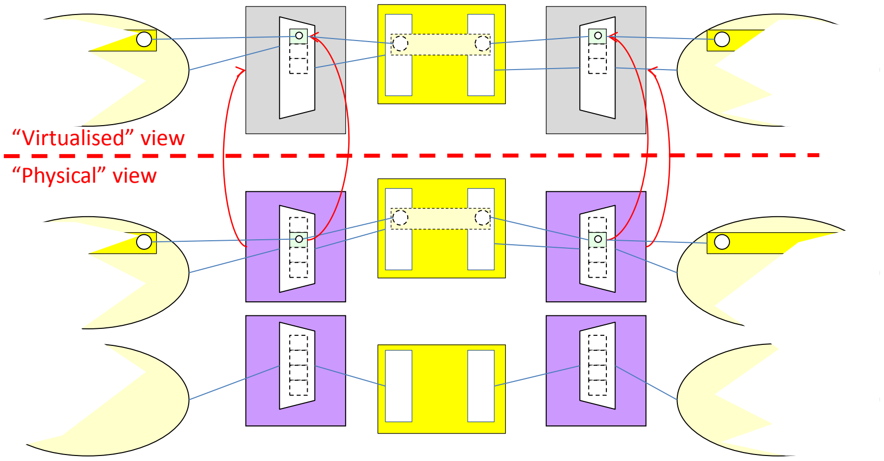


Figure 4-12 Move of allocation with no change to "Virtualized" view

After some further time the operator may choose to add capacity to the "Virtualized" view as illustrated in the next figure.

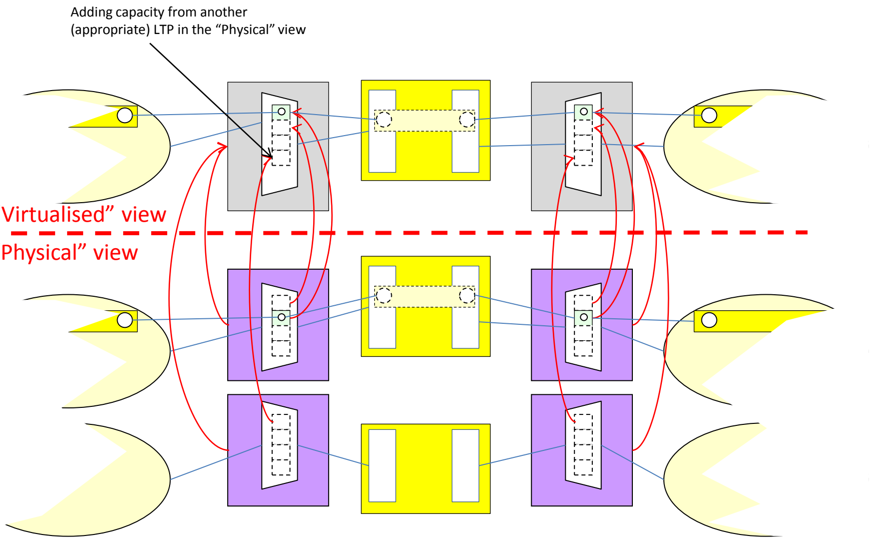

Adding capacity from another (appropriate) LTP in the "Physical" view
"Virtualised" view
"Physical" view

Figure 4-13 Capacity from server LayerProtocol Server LTPs

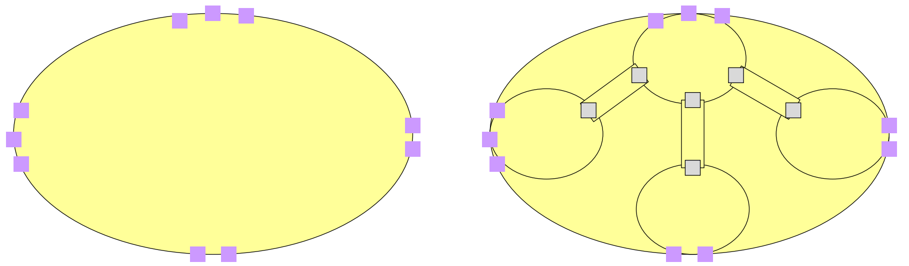
In this case there are two "Physical" view LayerProtocol Server LTPs associated with a single "Virtualized" view Pool LTP. Note that the Server LTP plays a Pool role.

The above illustration sequence leads to the following observations:

- There is no fixed association between the resources represented in the "Virtualized" view and the resources represented in the "Physical" view.
  - The identifiers in the two spaces must be different. This will be discussed in a following section.
- The "LtpRelatesToLtpInOtherView" association can provide all necessary view interrelationships

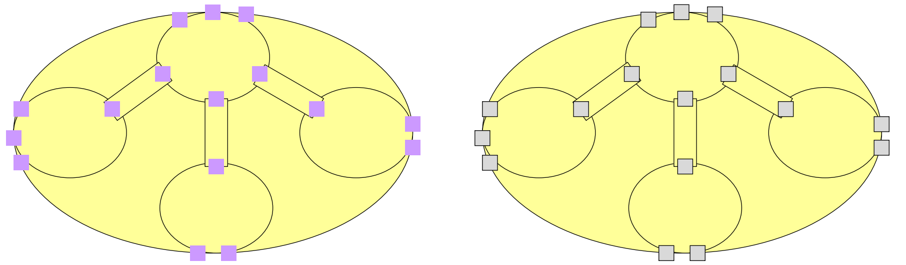
4.3 View boundaries and intermediates

In the previous section the "Virtualized" view had no physical ports. However, clearly a client to a network may need to connect at a physical port. The following figure shows several network cases as simple sketches where the outer ellipse boundary represents the actual commercial network boundary. A normal interworking case the operator exposes nothing of the interior of the network so the network is opaque and only the physical edge detail is provided (as show in the upper left diagram in the figure below). In some cases the operator may choose to expose apparent interior structure to perhaps explain capacity limitations. The network is essentially semi-transparent. It is possible that the network edge is essentially in the cloud so that even the interconnects are virtualized. A fully virtualized case where there is some exposure of internal constraints is shown in the lower right diagram in the figure below.



Opaque network
Exposing only physical edge

Semi-Transparent Network
Exposing physical edge and virtual interior topology



Transparent Network
Exposing physical interior topology

Semi-Transparent Virtual network
Exposing virtual edges and virtual interior topology

🟪 LTP bound to physical port
⬜ LTP without direct physical port (Floating/virtual/pool)
⬭ ForwardingDomain
▭ Link

Figure 4-14 Various view boundaries

4.4 More on views and names/identifiers

Each view may have its own name spaces and/or identifier spaces. An entity, regardless of which view it is in, will expose the appropriate name and identifiers using the attributes highlighted in TR-512.3 ONF Core IM Foundation. An entity may have several names and several identifires. An entity may be referred to using an address (a sequence of names/identifiers) where the names/identifiers have a local scope smaller than the context in which the entity needs to be uniquely determined.

In the following figure a number of views are exposed where each has its own namespace and where the LTPs relate via the "LtpRelatesToLtpInOtherView" association as discussed in the earlier section. There could be more or less views in the recursion and the discussion here is not on the absolute number of levels but instead on how they relate and on how the things in the views are referenced.

The most abstracted view (Abstract Intent[12]) shows the FC bounded represents a "Service"[13] or Call [ITU-T G.8081][14]. The FC is, as usual by LTPs, at the actual physical edge of the

---

[12] The term “Intent” is being used loosely here

[13] The usage of the term “Service” is intentionally vague here.

administration of the network. There is a two level hierarchy of LTPs shown where the lower (grey) represents the pool of physical network access ports and the upper LTP represents the per-"Service"/Call forwarding termination[15]. The layering of the upper LTP is that of the "Service"/Call.

These LTPs have abstract references. A common acronym for references at this level of abstraction is TRI. The TRI will carry a reference that is known by both either side of the administrative demarcation.

Depending upon the approach to the TRI generation, the TRI may be structured with a number of fields as an address or may be a single opaque field. Depending upon the quality of the TRI scheme, the TRI could be considered as either a name or an identifier (or address of names or identifiers). Regardless, the name "TRI" would be conveyed in the valueName field of the NameAndValue type (used for the appropriate localId or for the appropriate name).

At the next level of abstraction shown (Detailed Intent) the FC represents a "Service" decomposition or a Connection etc. The same approach is used for the SNP reference relevant at the next level of abstraction. The layering here is more precise, representing the effect of the network as viewed through the physical port. In this particular case, each LTP bounding the call is realized by a pair of LTPs in the connection[16].

In the final two levels of abstraction ("Realization" and "Physical Network") the FCs and LTPs take their more familiar roles.

---

[14] The choice of term depends upon the terminology context and the usages are not always directly analogous in detail (but at this level of description are sufficient).

[15] The layering of the lower LTPs depends upon the variety of accesses and will not be discussed further here.

[16] This level may be decomposed into “connection” segments and there may be many recursions of abstraction decomposition.

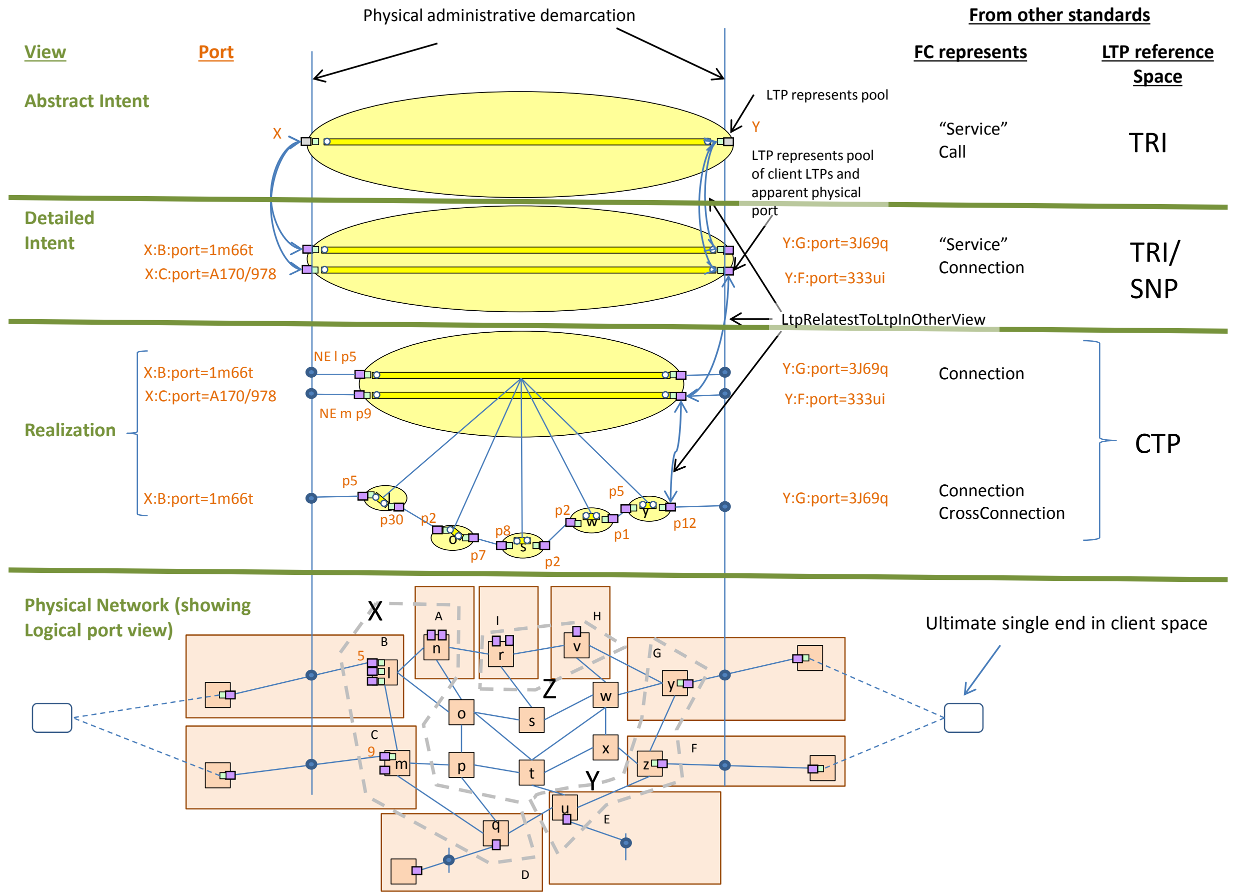


Figure 4-15 Various interrelated network views in a multi-party context

A final consideration at the edge of the network is the layering perceived by the client in a case where there is a device at the edge of the network that is not operating at the layer of the service. The figure below shows such a case. The key observation is that the layering of ports deep in the network is projected through the ports at the edge to form a hybrid apparent layering structure that is then exposed to the client. The exposure is exactly what would be seen if the client were to "look into" the edge port.

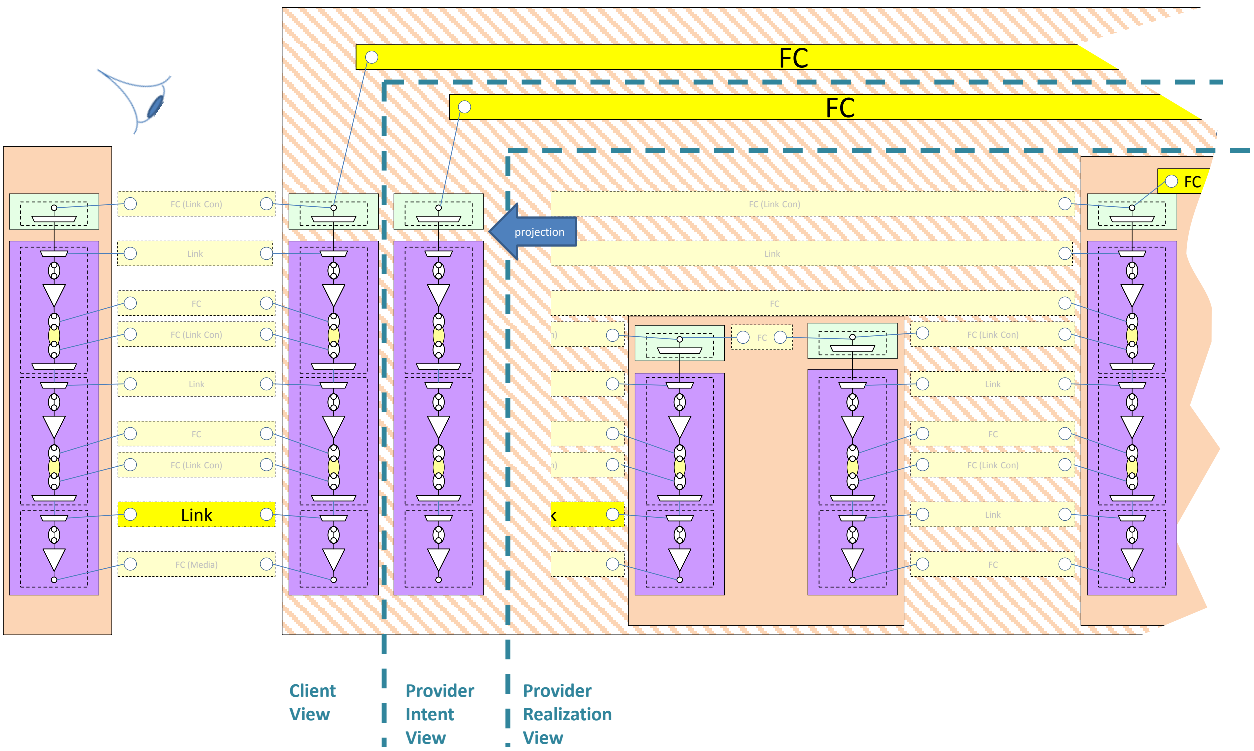


Figure 4-16 Complex network edge

4.5 Off-network reference and the clients view

The following figure shows the positioning of a link with an LinkPort that will use the "offNetworkAddress" attribute rather than a fully resolved LTP. Each blue dot in the figure represents an off-network address.

Unlike the case of the Client in the previous section, the Provider does not need to have any knowledge of the client port, the client does not need to present any view of their network to the Provider. The provider could create a dummy LTP to represent the client port or could simply end the Link with an off-network reference (offNetworkAddress)[17] in the LinkPort.

---

[17] Note that the attribute in the model is «Experimental»

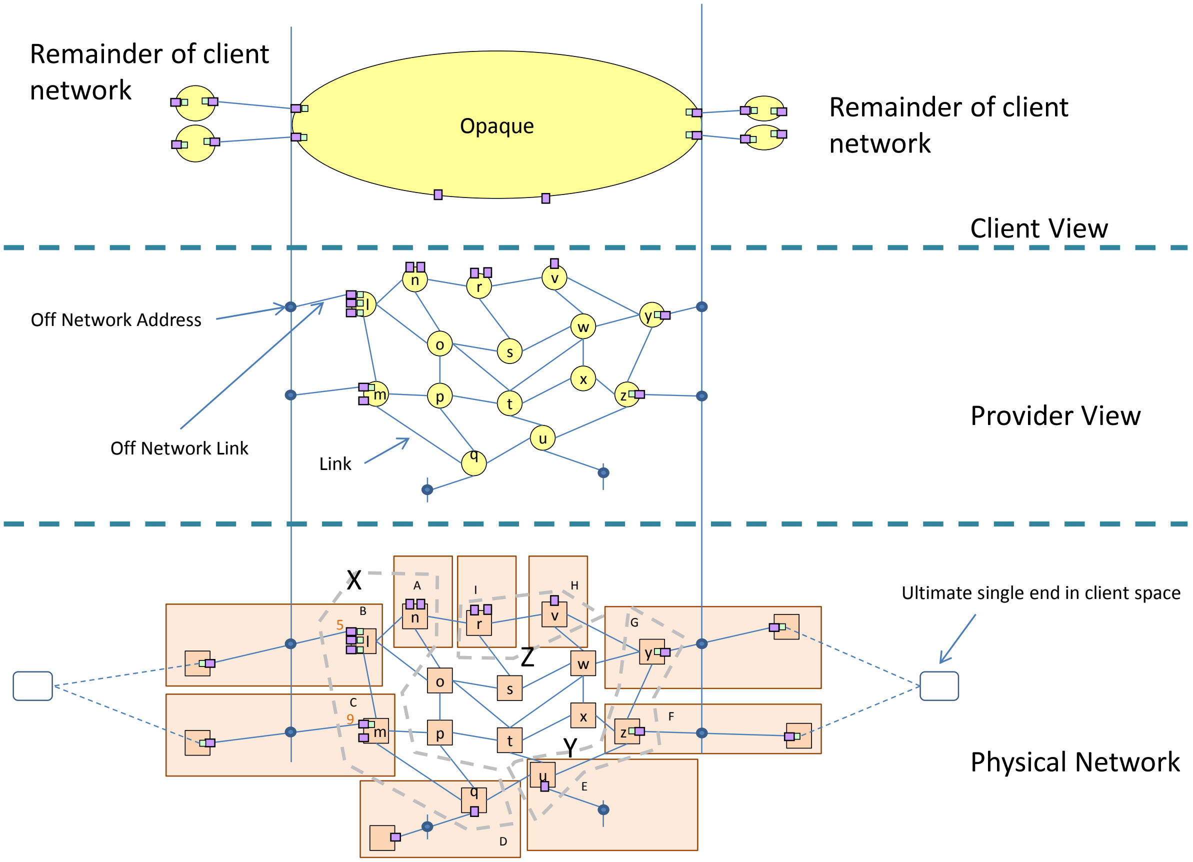
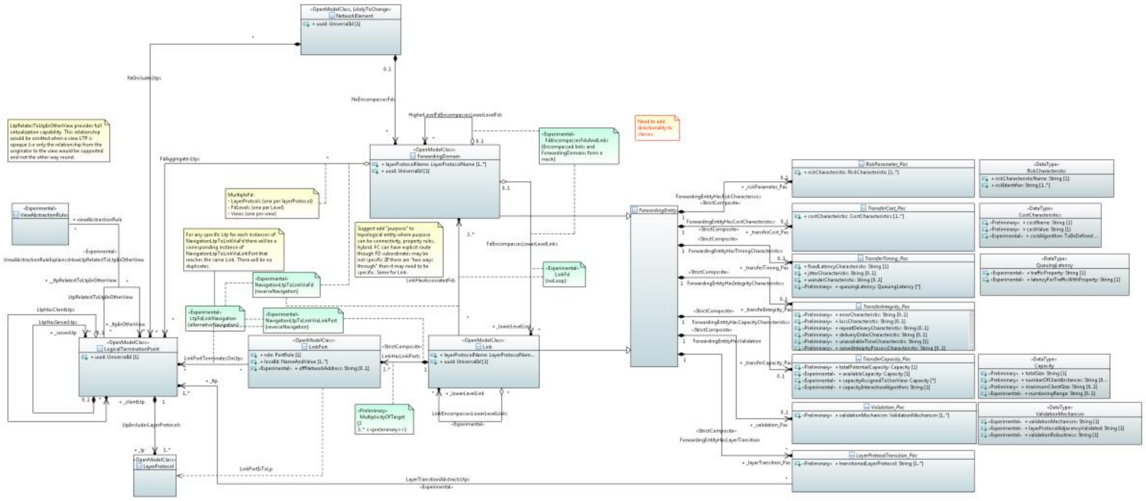


Figure 4-17 Complex network edge

4.6 Detailed properties of Topology


«OpenModelClass, LikelyToChange» NetworkElement
+ uuid: UniversalId [1]
NeIncludesLtps
NeEncompassesFds
HigherLevelFdEncompassesLowerLevelFds
«OpenModelClass» ForwardingDomain
+ layerProtocolName: LayerProtocolName [1..*]
+ uuid: UniversalId [1]
FdAggregatesLtps
«Experimental» FdEncompassesFdsAndLinks
Need to add directionality to classes
«Experimental» ViewAbstractionRules
+ viewAbstractionRule
ViewAbstractionRuleExplainsHowLtpRelatesToLtpInOtherView
LtpRelatesToLtpInOtherView
LtpHasClientLtps
LtpHasServerLtps
«OpenModelClass» LogicalTerminationPoint
+ uuid: UniversalId [1]
LinkPortTerminatesOnLtps
«OpenModelClass» LinkPort
+ role: PortRole [1]
+ localId: NameAndValue [1..*]
«Experimental» + offNetworkAddress: String [0..1]
LinkHasLinkPorts
«OpenModelClass» Link
+ layerProtocolName: LayerProtocolName...
+ uuid: UniversalId [1]
LinkHasAssociatedFds
FdEncompassesLowerLevelLinks
LinkEncompassesLowerLevelLinks
LtpIncludesLayerProtocols
«OpenModelClass» LayerProtocol
LinkPortIsToLp
LayerTransitionAbstractsLtps
ForwardingEntity
RiskParameter_Pac
TransferCost_Pac
TransferTiming_Pac
TransferIntegrity_Pac
TransferCapacity_Pac
Validation_Pac
LayerProtocolTransition_Pac
«DataType» RiskCharacteristic
«DataType» CostCharacteristic
«DataType» QueuingLatency
«DataType» Capacity
«DataType» ValidationMechanism

CoreModel diagram: Topology-DetailWithRules

© 2016 Open Networking Foundation

Figure 4-18 Topology details with rules

The figure above shows finalized, preliminary and experimental extensions of the Topology model.

5 Work in progress

Development of Rules for propagation of topological parameters to clients (e.g. cost from Link to FC riding over it and from FC to its client links).

End of document

© 2016 Open Networking Foundation