Core Information Model (CoreModel)

TR-512.4

Topology

Version 1.3.1
January 2018

ONF Document Type: Technical Recommendation
ONF Document Name: Core Information Model version 1.3.1

Disclaimer

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

Important note

This Technical Recommendations has been approved by the Project TST, but has not been approved by the ONF board. This Technical Recommendation is an update to a previously released TR specification, but it has been approved under the ONF publishing guidelines for 'Informational' publications that allow Project technical steering teams (TSTs) to authorize publication of Informational documents. The designation of '-info' at the end of the document ID also reflects that the project team (not the ONF board) approved this TR.

Table of Contents

Disclaimer **2**

Important note **2**

Document History **6**

1 Introduction **7**

1.1 References 7
1.2 Definitions 7
1.3 Conventions 7
1.4 Viewing UML diagrams 7
1.5 Understanding the figures 7

2 Introduction to the Topology Model **7**

3 Topology model **8**

3.1 Topology model overview 8
3.2 Topology model detail 12
3.2.1 Link 12
3.2.2 LinkPort 13
3.2.3 ForwardingDomain 14
3.2.4 FdPort 16
3.2.5 LogicalTerminationPoint 17
3.3 Topology model classes, related classes and structures 18
3.4 Topological properties of the ForwardingEntity 20
3.4.1 ForwardingEntity 20
3.4.2 LayerProtocolTransition_Pac 21
3.4.3 RiskParameter_Pac 22
3.4.4 TransferCapacity_Pac 23
3.4.5 TransferCost_Pac 23
3.4.6 TransferIntegrity_Pac 24
3.4.7 TransferTiming_Pac 25
3.4.8 Validation_Pac 25
3.5 Model showing topology, forwarding and termination 26
3.6 Further defining the Link 26

4 Explanatory figures **28**

4.1 Basic Topology 28
4.2 Topology in a Control Context 30
4.3 Topology and views 35
4.4 View boundaries and intermediates 41
4.5 The FdPort 42
4.6 More on views and names/identifiers – The FC representing a Call 42

4.6.1 Call, Service, the Resource-Service Continuum and the Capability Continuum 45
4.7 Off-network reference and the clients view 46
4.8 Serial-Compound Links 47
4.9 Transitional Links 52
4.10 Multi-Port Link 55
4.11 State Dependency 57
4.12 Inverse Multiplexing 58

5 Work in progress (see also TR-512.FE) 59
5.1 Detailed properties of Topology 59
5.2 Cost algorithms 60
5.3 FC/Link Convergence 60
5.4 NearEnd/FarEnd, Input/output and ingress/egress 60
5.5 Complex Transitional Links 60
5.6 Non-orthogonal FDs 62

List of Figures

Figure 3-1 Basic topology view showing the model 9
Figure 3-2 Topology model highlighting aggregation 10
Figure 3-3 Topology model highlighting layering 11
Figure 3-4 Basic topology inter-view navigation 12
Figure 3-5 Topology _Pac detail 20
Figure 3-6 Topology, Forwarding and Termination 26
Figure 4-1 ForwardingDomain recursion with Link 29
Figure 4-2 Topology in a Control Context 31
Figure 4-3 ForwardingDomain link recursion showing device (Constraint Domain) and control 32
Figure 4-4 ForwardingDomain, Link and LTP associations 33
Figure 4-5 FDs and Topology 34
Figure 4-6 LTPs Encompassed by FDs (at one layer-protocol) 35
Figure 4-7 LTPs Encompassed by FDs (at several layer-protocols) 35
Figure 4-8 LTP "pooling" client LTPs 36
Figure 4-9 Views of Link, LinkPort and LTP showing LTP pooling 37
Figure 4-10 Views of "virtualization" of LTPs with server side LTP representing a pool 38
Figure 4-11 Starting condition 39
Figure 4-12 Resource allocation 40
Figure 4-13 Move of allocation with no change to "Virtualized" view 40
Figure 4-14 Capacity from server LayerProtocol Server LTPs 41

Figure 4-15 Various view boundaries 42
Figure 4-16 Various interrelated network views in a multi-party context 43
Figure 4-17 Complex network edge 45
Figure 4-18 Complex network edge 47
Figure 4-19 Opaque network between two access points 48
Figure 4-20 Simple adjacency between two access points viewed as a Link 48
Figure 4-21 Network between two accesses in the same layer protocol as the access 48
Figure 4-22 The realization of an apparent Link 49
Figure 4-23 The Link resulting from the underlying configuration 50
Figure 4-24 Serial compound Link showing model 50
Figure 4-25 Network demarcation at some point along a cable 50
Figure 4-26 Network demarcation showing network detail 51
Figure 4-27 The realizing of an apparent Link in a mid-cable demarked network 51
Figure 4-28 Signaling is necessary to provide dynamic operation of the apparent Link 51
Figure 4-29 Network exposing configurable FD to client and providing self-service control 52
Figure 4-30 Sketch of two network layers 53
Figure 4-31 Forming the Transition Links 54
Figure 4-32 Path computation finds available routes through the topology 54
Figure 4-33 Path computation finds available routes through the topology 55
Figure 4-34 Two FCs are created, one in the ODU2 layer and one in Ethernet 55
Figure 4-35 Multi-ported Link supporting an FC 56
Figure 4-36 Multi-ported Link supporting supported by a complex server configuration 57
Figure 4-37 Lightweight sketch of a multi-layered network 57
Figure 4-38 Representing Inverse Multiplexing 59
Figure 5-1 Topology details with rules 60
Figure 5-2 Two forms of complex Transitional Link 61
Figure 5-3 A particularly complex multi-layer multi-port Transitional Link 61
Figure 5-4 Off-network Transitional Link 62

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]
1.3	September 2017	Version 1.3 [Published via wiki only]
1.3.1	January 2018	Addition of text related to approval status.

1 Introduction

This document is an addendum to the TR-512 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.

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

- 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. Topology capability and other specification considerations are covered in TR-512.7. Topology should also be considered in the context of the Processing Construct and Constraint Domain described in TR-512.11.

A data dictionary that sets out the details of all classes, data types and attributes is also provided (TR-512.DD).

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 12. The figure below shows the pictorial representation of these key classes (see 1.5 for reference to diagram key).

[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. These associations are NOT invariant and hence can change through the life of the entity. Under many circumstances these associations need not be populated (e.g. when the rate of change of the association is excessive or derivation of the supporting physical components is complex)

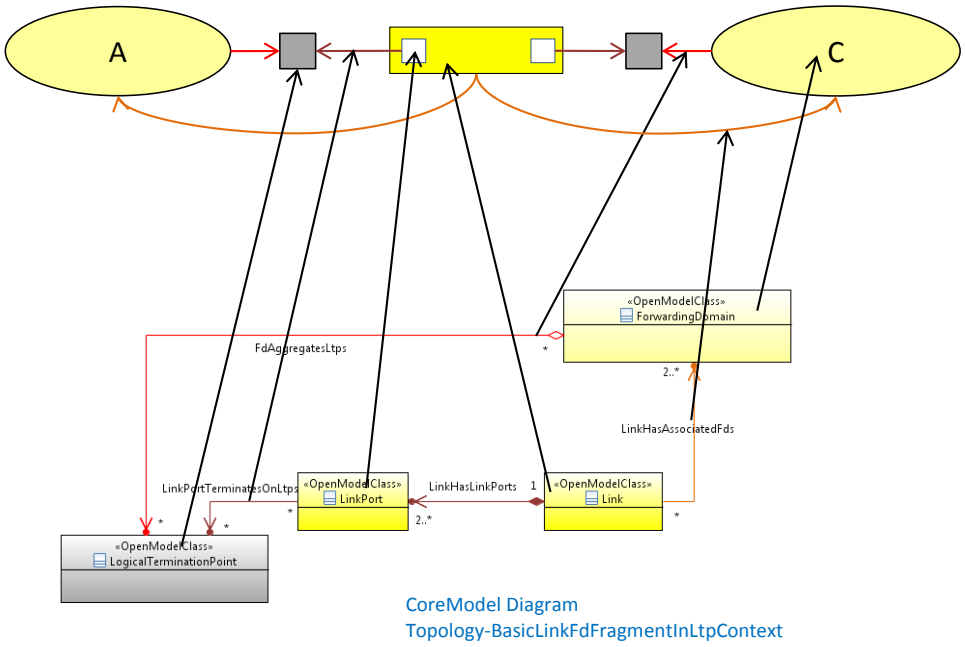
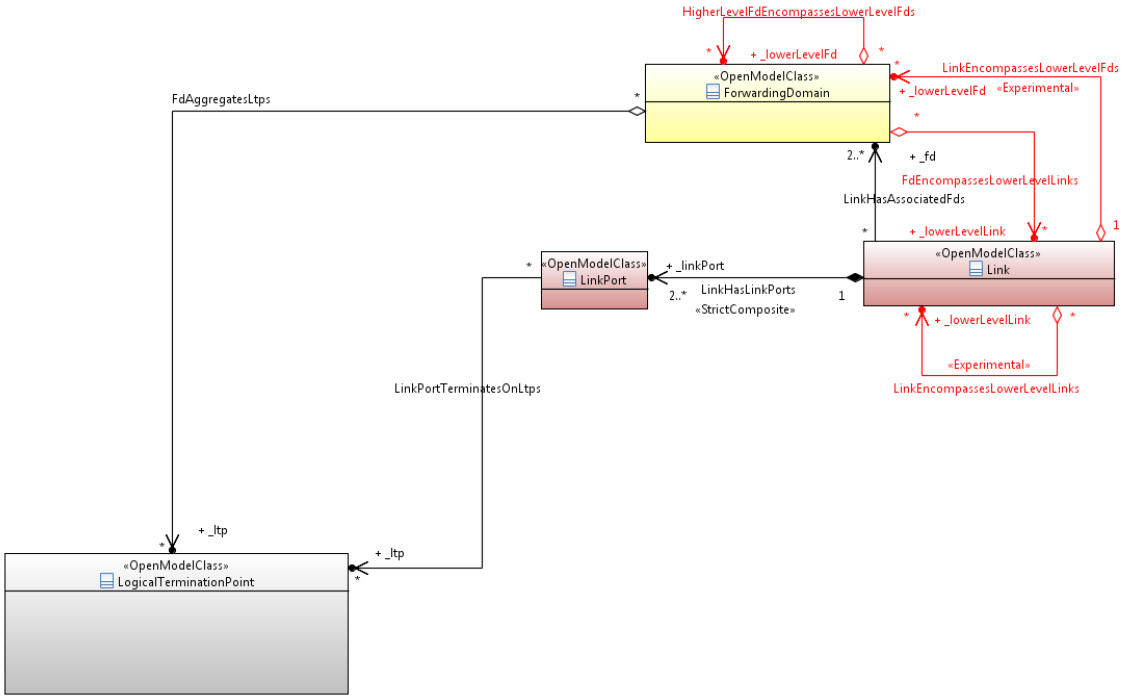


Figure 3-1 Basic topology view showing the model

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 28.

[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.



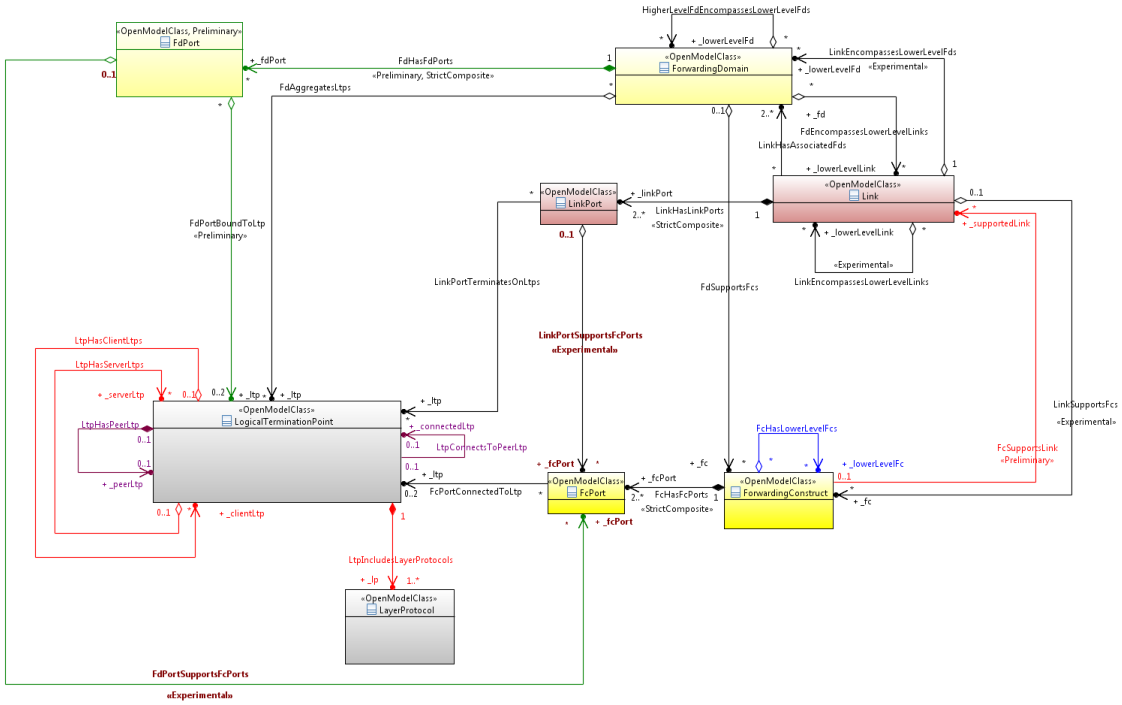
CoreModel diagram: Topology-AggregationSkeleton

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.
- The FdPort, which provides support for asymmetric FDs, denoted in green
- Inter-port associations, showing topology port supporting FcPort, highlighted with maroon bold text

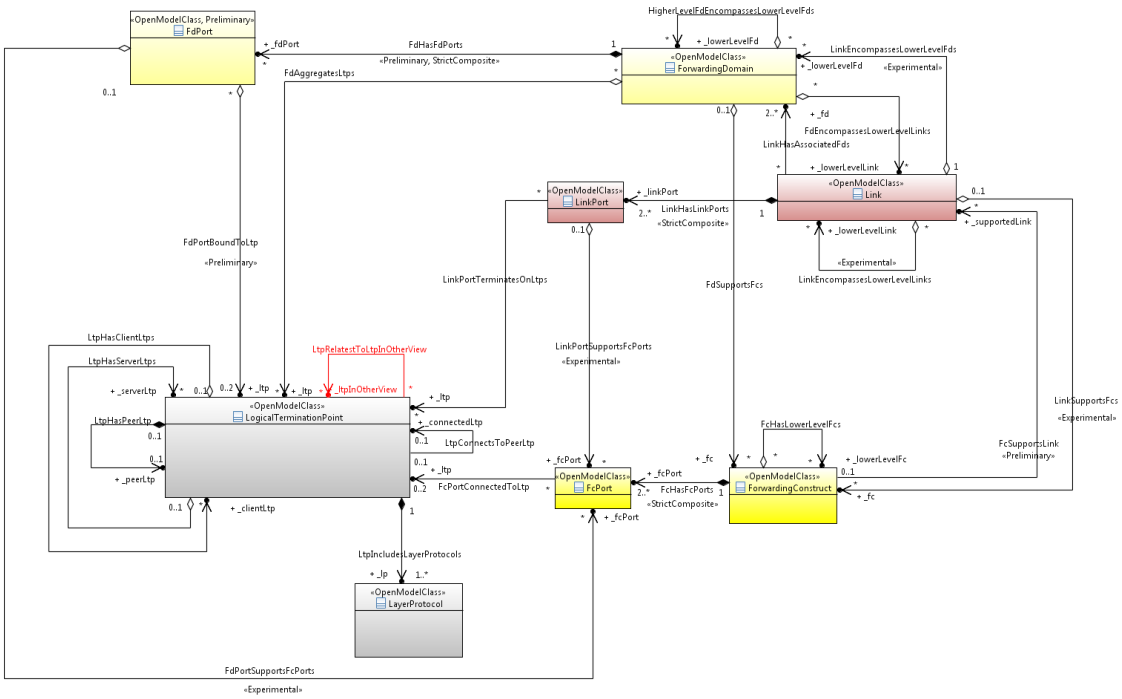
Explanation of this aspect of the model is provided in section 4.3 Topology and views on page 35.



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 4.3 Topology and views on page 35 and subsequent sections.



CoreModel diagram: Topology-InterViewSkeleton

Figure 3-4 Basic topology inter-view navigation

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 Link

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

The Link class models effective adjacency between two or more ForwardingDomains (FD). For digital layer networks, 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:

- ForwardingEntity
- GlobalClass

Table 1: Attributes for Link

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description

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).
_lowerLevelLink	Experimental	A Link may be formed from subordinate links (similar FD formations from subordinate 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 link connection. 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.
_linkSpec	Experimental	See referenced class
_linkSpecReference:ClassRef	SpecReference Experimental	See referenced class

3.2.2 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 2: 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 capability.
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 the defined flow at the LinkPort.
_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-directional LTPs Zero LTP - BreakBeforeMake transition - Planned LTP not yet in place - Off-network LTP referenced through other mechanism.
_fcPort	Experimental	Where a Link supports FCs each LinkPort of that Link supports the corresponding FcPorts.

3.2.3 ForwardingDomain

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

The ForwardingDomain (FD) class models the topological component that represents a forwarding capability that provides the opportunity to enable forwarding (of specific transport

characteristic information at one or more protocol layers) between points.
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 FCs may be formed between LTPs at the boundary of the FD or between physical ports at the boundary of the FD (for media layers).
A number of FDs (related by Links) may be grouped and abstracted to form an FD where that FD represents the effect of the underlying FDs but where the detailed structure is not apparent. This grouping and abstraction is potentially recursive.
An FD represents an abstraction of some combination of software behavior, electronic behavior and physical structure that provides a forwarding capability.
At a lower level of recursion an FD could represent a forwarding capability within a device.
A device may encompass two or more disjoint forwarding capabilities and may support more than one layer protocol, hence more than one FD.
A routing fabric may be logically partitioned to represent connectivity constraints, hence the FD representing the routing fabric may be partitioned into a number of FDs representing the constraints.
The FD represents a subnetwork [ITU-T G.800], FlowDomain [TMF 612] and a MultiLayerSubNetwork (MLSN) [TMF 612].
As in the TMF concept of MLSN the FD can support more than one layer-protocol.
Note that the ITU-T G.800 subnetwork is a single layer entity.

Inherits properties from:

- ForwardingEntity
- GlobalClass

Table 3: 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 an 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 aggregates one or more FCs. An 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. For asymmetric FDs, the association to the LTP is via the FdPort.

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
_fdSpec	Experimental	See referenced class
_fdPort	Preliminary	The association of the FD to LTPs is either made directly for symmetric FDs or via FdPort for asymmetric FDs.
_pc	Experimental	An FD constrains one or more PCs. A constrained PC connects LTPs on the boundary of the FD

3.2.4 FdPort

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::FdPort

The association of the FD to LTPs may be direct for symmetric FDs and may be via FdPort for asymmetric FDs.
The FdPort class models the role of the access to the FD function.
The capability to set up FCs between the associated FdPorts of the FD depends upon the type of FD. It is asymmetry in this capability that brings the need for FdPort.
The FD can be considered as a component and the FdPort as a Port on that component.

Inherits properties from:

- LocalClass

This class is Preliminary.

Table 4: Attributes for FdPort

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
role		Each FdPort of the FD has a role (e.g., symmetric, hub, spoke, leaf, root) in the context of the FD with respect to the FD capability.
fdPortDirection		The orientation of the defined flow at the FdPort.
_ltp		An instance of FD is associated with zero or more LTP objects. The LTPs that bound the FD provide capacity for forwarding. For asymmetric FDs, the association to the LTP is via the FdPort.

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_fcPort	Experimental	Where an FD is asymmetric and hence has FdPorts and where that FD and supports FCs, appropriate FdPorts of that FD support the corresponding FcPorts.
_pin	Experimental	For media, a pin on the boundary of the FD.
_fdPort	Experimental	An FdPort may have a direct association to another FdPort where there is a transition from one domain to another but where there has been no termination.

3.2.5 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 5: Attributes for LogicalTerminationPoint

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
physicalPortReference	Preliminary	One or more text labels for the unmodeled 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).

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_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 this LTP.
_ltpInOtherView	Preliminary	References one or more LTPs in other views that represent this LTP. The referencing LTP is the provider of capability.
_accessPort	Experimental	Provides a reference to the place where the signal is accessed. It may represent a physical place (some part of one or more connectors) or a virtual equivalent where there is no further protocol layering (visible).
_transferCapacity_Pac	Experimental	The LTP has as an inherent capacity derived from underlying capability. The capacity of a particular LTP may be dependent upon other uses of resource in the device and hence it may vary over time. The capacity of a Link is dependent upon the capacity of the LTPs at its ends. An LTP 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.
_ltpSpecReference:ClassRef	SpecReference Experimental	Provides a reference to a specification which is in the form of a class definition. An instance of LTP will reference a class (by a universally unique id) that provides definition that extends the LTP including attributes and structure that are present in the LTP instance but that are not defined in the native LTP class.
_fdRuleGroup	Experimental	An LTP can reference FD rules that the FD that aggregates it also references so that the rules can then apply to the LTP.
_embeddedClock		See referenced class

3.3 Topology model classes, related classes and structures

The classes used to represent topology are summarized in the figures in section 3.1[3]. In addition figures in TR-512.8 and TR-512.11 highlight the generalized relationship between the device[4] and the topology model classes.

Considering the topology model in the context of a device and of a network:

[3] Some associations not related to the focus of this document are omitted.
[4] A device is essentially a small network. The network element concept used in previous releases to represent a device has been found to be inadequate and a more general model has been developed. The device model is described in TR-512.8 and TR-512.11.

- An FD may be a subordinate part of a device, may coincide with a device boundary or may be larger than, and independent of, any device (See for example FDs A.1 and A.3 in Figure 4-3 ForwardingDomain link recursion on page 32).
- An FD may encompass lower level FDs. This may be such that:
    - An FD directly contained in a device is divided into smaller parts
    - An FD not encompassed by a device is divided into smaller parts some of which may be encompassed by a device (see Figure 4-3 ForwardingDomain link recursion on page 32)
    - 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 Links that go off-network and hence outside the view are beyond the bounds of the uppermost FD and are not encompassed by any FD. All other Links are always encompassed by one or more FDs which may be the FD representing the whole network.
- 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[5]
        - 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-ported asymmetric Link
- An FD aggregates LogicalTerminationPoints (LTPs) that bound it. An LTP represents a stack of layer-protocol terminations, where the details of each layer-protocol are held in the LayerProtocol (LP). An LTP may be:
    - Part of a device
    - Conceptually independent from any device[6]
- A Link terminates on LTPs via its contained LinkPorts.

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

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

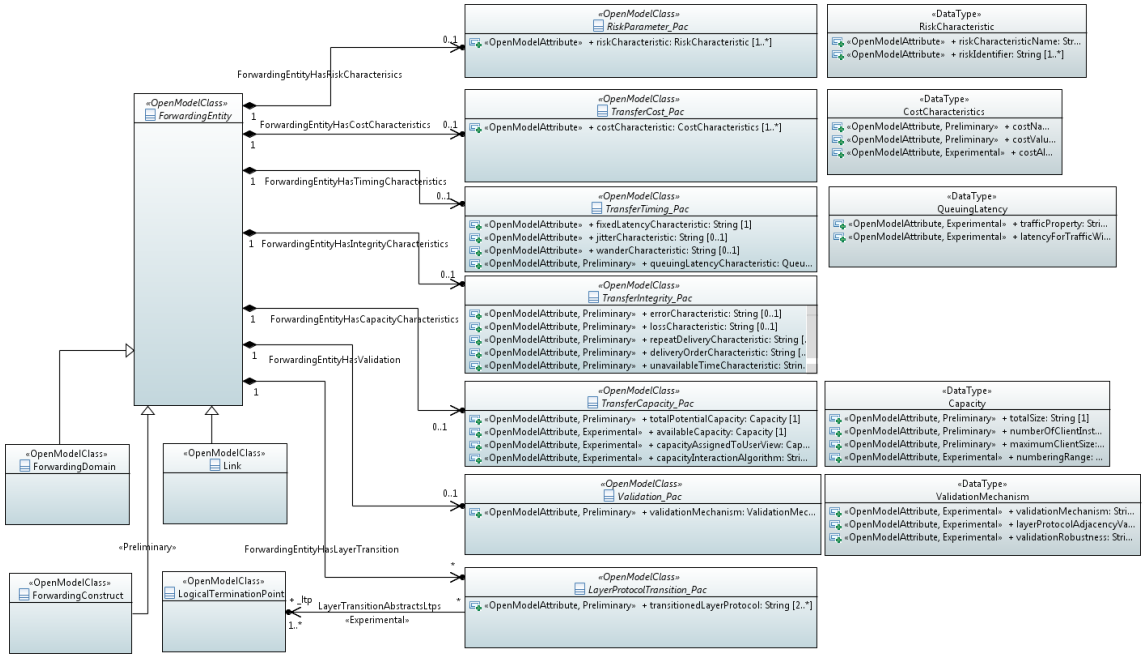
[6] The assumption is that the LTP can be floating (representing a pool) in the context of a network as a whole (represented by an FD). The LTP has a UUID to allow it to be identified. Under these circumstances it does not need to be contained in anything (although it is pooled by the FD representing the network). It clearly does need to be accessible via a controller (see TR-512.8).

- 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 in _Pacs[7]of attributes. These attributes are elaborated in the figure below. For further details of types etc. refer to TR-512.DD.

.



CoreModel diagram: Topology-Detailed_PacProperties

Figure 3-5 Topology _Pac detail

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

[7] Note that the _Pac mechanism will be replaced with a decoration approach using the specification model as described in TR-512.7.

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 6: 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 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 7: Attributes for LayerProtocolTransition_Pac

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
transitionedLayerProtocol	Preliminary	Provides the ordered structure of layer protocol transitions encapsulated in the ForwardingEntity. The list starts with the client side as the first entry and includes all layer-protocol names (hence the smallest number is 2 as otherwise the Link is not transitional). The ordering relates also to the LinkPort role (which emphasizes the orientation). Where the transitional link is multi-ported and layer asymmetric the list includes the superset of layer-protocol names. Transitional links can only be applied where the transition for each port is such that all transitions between any ports are subsequences of the list. The specific subsequence is determined by the LayerProtocols of the LTP associated with the LinkPort and the role of the LinkPort.
_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 ForwardingEntities 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 ForwardingEntity can be listed in the RiskParameter_Pac of that ForwardingEntity.
A segment has one or more risk characteristic.
Shared risk between two ForwardingEntities compromises the integrity of any solution that use one of those ForwardingEntity as a backup for the other.
Where two ForwardingEntities have a common risk characteristic they have an elevated probability of failing simultaneously compared to two ForwardingEntities that do not share risk characteristics.

This class is abstract.

Table 8: 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.

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 9: 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 10: 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 11: 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.

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
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 12: 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).
queuingLatencyCharacteristic	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.

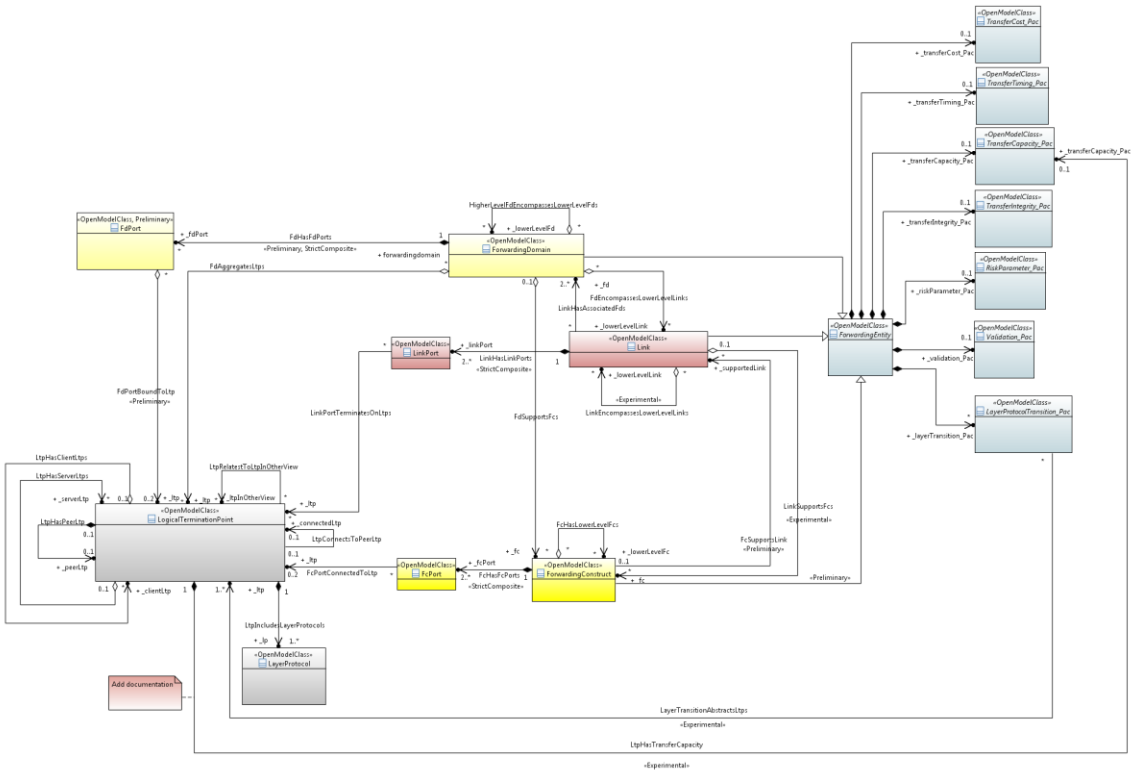
This class is abstract.

Table 13: 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 3-6 Topology, Forwarding and Termination

3.6 Further defining the Link

It is important to clarify a number of definitions and to derive further definition:

- Link (from [ITU-T G.800]):
  - A "topological component" which describes a fixed relationship between a "subnetwork" or "access group" and another "subnetwork" or "access group".
- "Normal" Link (for the purpose of this document)

    - A Link that is supported by a single server layer trail and the client label is the same at both ends of the link.
- [ITU-T G.800] also describes four other ways that server layer may support a link (rephrased slightly here):
    - Transitional link: A transitional link is a type of link that consists of the LTP at the edge of one subnetwork and a corresponding LTP at the edge of another subnetwork that operates on different instances of CI or whose CI is the same, but with different layer information.
        - Transitional Links are dealt with in section 4.9
    - Links supported by multiple server layer trails:
        - Multiple parallel links between the same subnetworks can be bundled together into a single composite link. Each component of the composite link is independent in the sense that each component link is supported by a separate server layer trail. The composite link conveys communication information using different server layer trails, thus the sequence of symbols crossing this link may not be preserved.
        - Multiple server layer trails can be combined using the inverse multiplexing technique described in [ITU-T G.805]. This creates a new composite rate trail with a capacity that is the sum of the capacity of the component trails.
            - Inverse multiplexing is covered in section 4.12
        - A link can also be constructed by a concatenation of component links and configured subnetwork connections. This is called a serial compound link.
            - Serial Compound Links are covered in section 4.8

A Link is a representation of potential/capability to enable forwarding between two or more ForwardingDomains. In most cases a Link is bidirectional between two ForwardingDomains (it can clearly be used unidirectionally). A Link represents potential capacity between the ForwardingDomains that bound it. A Link represents an adjacency.

A link capability is usually realized by a server layer protocol that ensures transfer transparently between its client's ForwardingDomains. A Link is a highly restricted capability that maintains all properties of the client traffic. When realized by a server layer, a Link is a reflection of:

- A ForwardingConstruct in the server layer
- A parallel set of ForwardingConstructs in the server layer

Note: As the Link is a reflection of server layer forwarding, it could be represented by a ForwardingConstruct where that ForwardingConstruct represents the full available capacity that can then be divided into capacity segments that are again represented by ForwardingConstructs

A Link can be considered to represent one or more of the following:

- The need for adjacency
- The offer of adjacency capability
- The expectation of adjacency
- The intent to provide adjacency
- A realized adjacency

- Actual current adjacency

Enabled forwarding between the ForwardingDomains is represented by a ForwardingConstruct within a Link where the ForwardingConstruct represents the transparent transfer of a flow of traffic. The Link may be channelized and provide an identifier for that traffic that is the same at all ends of the Link

- The link capacity could be enabled on a per channel basis
- When realized by a server layer in simple cases all capacity is immediately enabled

There may be a demand for adjacency, represented by a Link that is not realized by any server layer forwarding. The Link may offer channelization and expose disabled ForwardingConstructs.

A server layer may offer mappings to different client layers. This may require configuration of the adapter between the client and the server. This configuration may be achieved automatically at ForwardingConstruct creation.

Note: whilst contention for resources between client layers is likely it is not explored further here.

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 7.

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 figure below focus on the ForwardingDomain class and the recursive aggregation relationship.

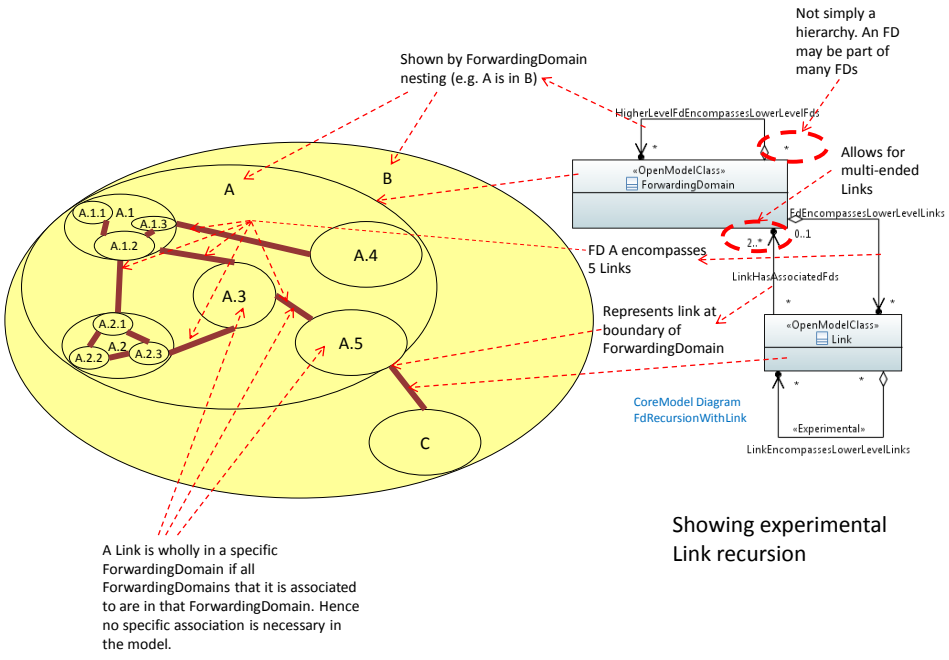


Figure 4-1 ForwardingDomain recursion with Link[8]

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. Also, we will assume symmetric FDs, so FdPorts are not needed.

The pictorial form shows a number of instances of FD interconnected by Links and shows nesting of FDs. The recursive aggregation (represented by an open diamond) association HigherLevelFdEncompassesLowerLevelFds 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

[8] 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.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[9]
- 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.

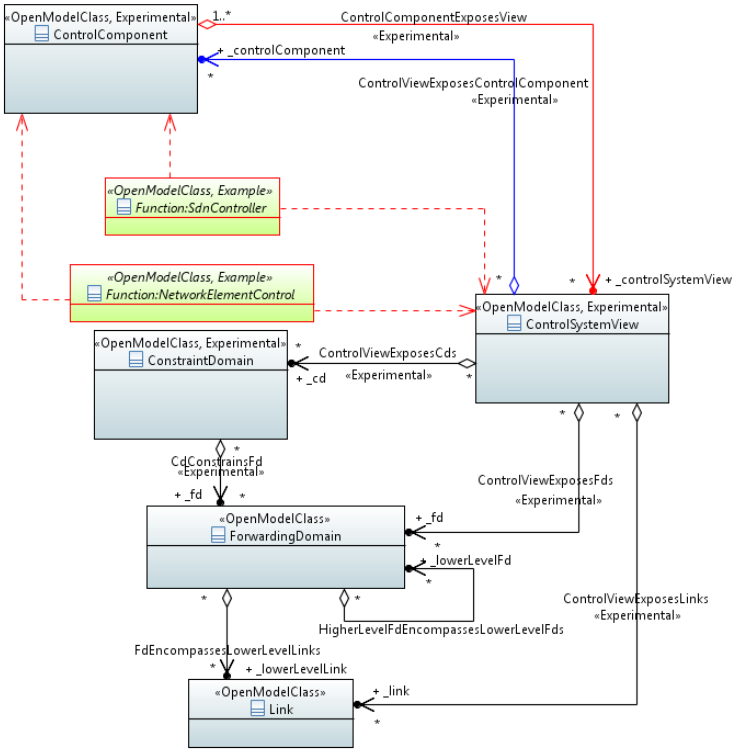
The association between Link and FD allows a Link to be terminated on two or more FDs (see Figure 4-4 ForwardingDomain, Link and LTP associations on page 33). Through this the model supports point to point Links as well as cases where the server ForwardingConstruct is multi-point terminated giving rise to a multi-ported Link. Multi-ported links occur in PON and Layer 2 MAC in MAC.[10]

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

4.2 Topology in a Control Context

The model fragment below shows the Control model (that has replaced the model of NetworkElement and SdnController – see TR-512.8).

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



CoreModel diagram: Topology-FdAndControl

Figure 4-2 Topology in a Control Context

The figure below shows a simplified view of the model and a pictorial

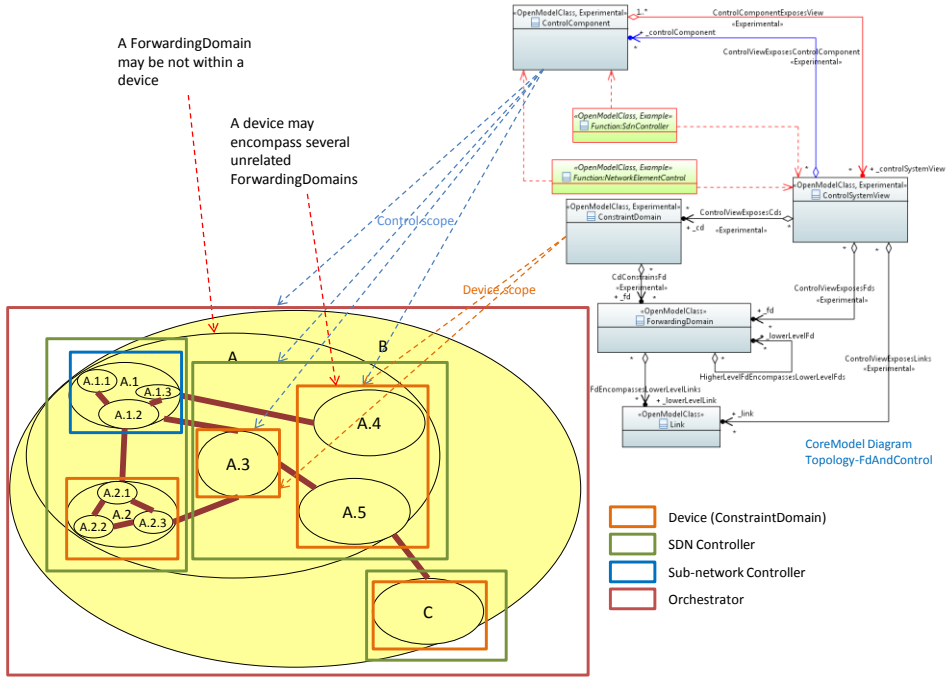


Figure 4-3 ForwardingDomain link recursion showing device (Constraint Domain) and control

The figure above the pictorial form shows an overlay of the new representation of the device and the SDN controller on the ForwardingDomains and a corresponding fragment of UML showing relevant classes (and examples of usage).

The figure emphasizes that at and below one particular level of abstraction of ForwardingDomain (the device boundary), the ForwardingDomains are all bounded by a specific ConstraintDomain (brown square). This is represented in the UML fragment by the aggregation association (diamond). There is potentially a lifecycle dependency in that the ForwardingDomain at this level can only be PLANNED if the device is not present or there is no traceability to physical resources.

The figure also shows that a ForwardingDomain need not be bounded by any control entity (as explained in the UML fragment by the * on the aggregation association), and that a ForwardingDomain may have a smaller scope than the whole device (even when considering only a single layer-protocol as noted earlier). In one case depicted the two ForwardingDomains, A.4 and A.5, in the scope of the device level Control Component are completely independent. In the other cases depicted, the subordinate ForwardingDomains (A.2.2.1- A.2.2.3) are themselves joined by Links emphasizing that the device 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 device does not mean that all ForwardingDomains at that level are also bounded by devices.[11]

[11] It should be noted that a device/ControlComponent is never within the bounds of an FD. The device is associated with levels in the FD hierarchy.

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. The LTP represents the pool of capacity to support clients. This pooling capability is a view of the potential of the LTP for adaptation from the server layer to the layer of concern. The Link capacity is defined by the intersection of adaptation of the LTPs at either end.

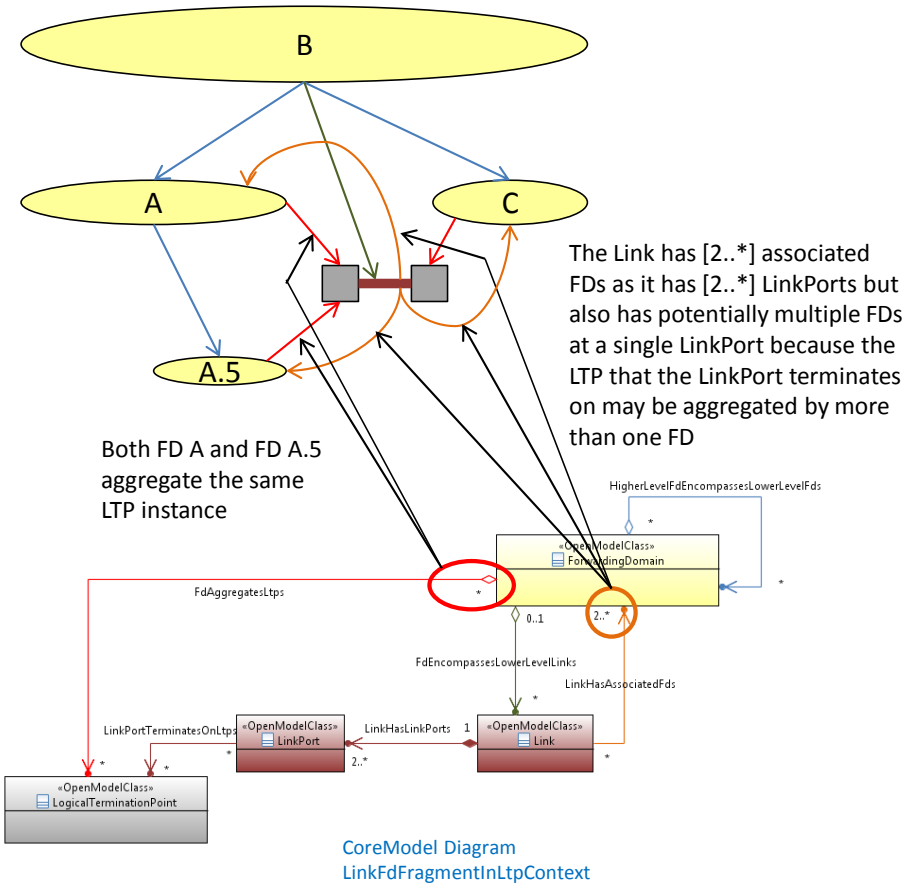


Figure 4-4 ForwardingDomain, Link and LTP associations

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.

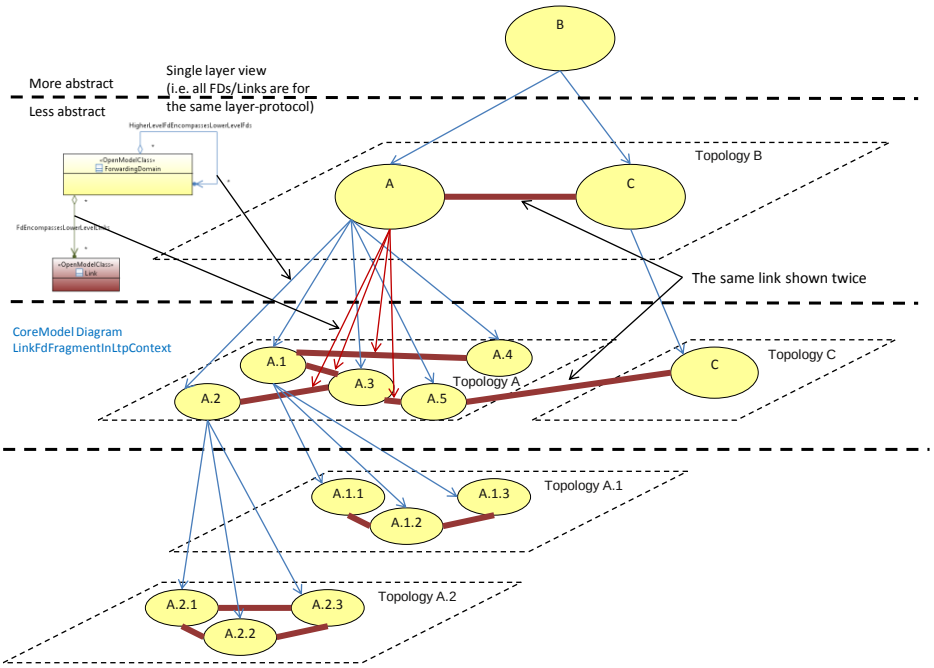


Figure 4-5 FDs and Topology

The following figure considers the topology further in combination with the FCs. The figure shows LTP on the boundary of two (or more) FDs. The LTP may be considered as on the boundary of FD if a LayerProtocol of the LTP:

- Is for the layer-protocol of the FD (and the LTP is accessible to the FD)
- Adapts to the layer-protocol of the FD (and the LTP is accessible to the FD)

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

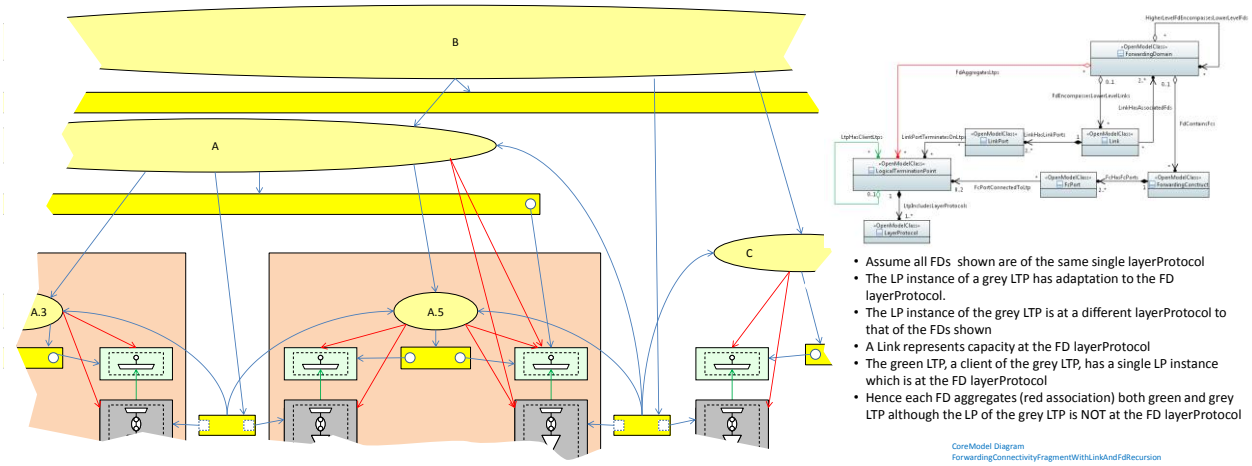


Figure 4-6 LTPs Encompassed by FDs (at one layer-protocol)

Clearly an LTP with multiple LPs may be on the boundary of FDs at multiple layer-protocols. An LTP may be on the boundary of several 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. The LTP is on the boundary of FD C and FD D at layer-protocol Y and on the boundary of FD B at layer-protocol X (as it adapts to layer-protocol X).

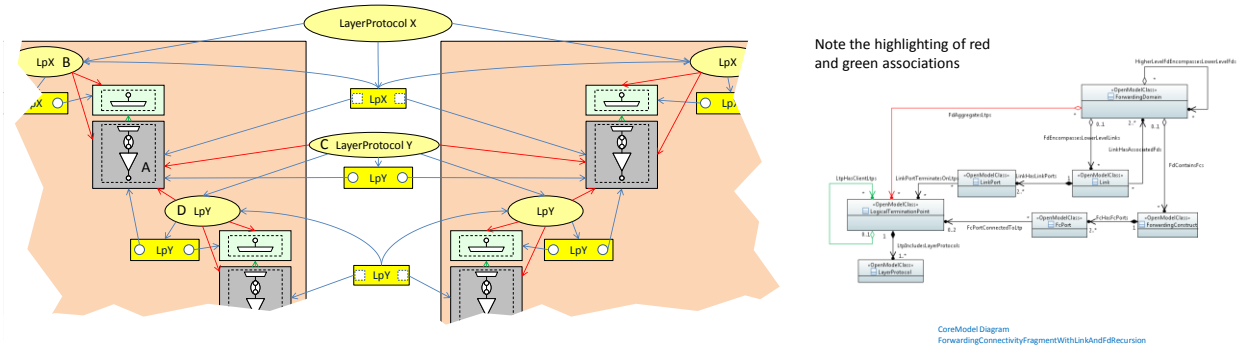


Figure 4-7 LTPs Encompassed by FDs (at several layer-protocols)

4.3 Topology and views

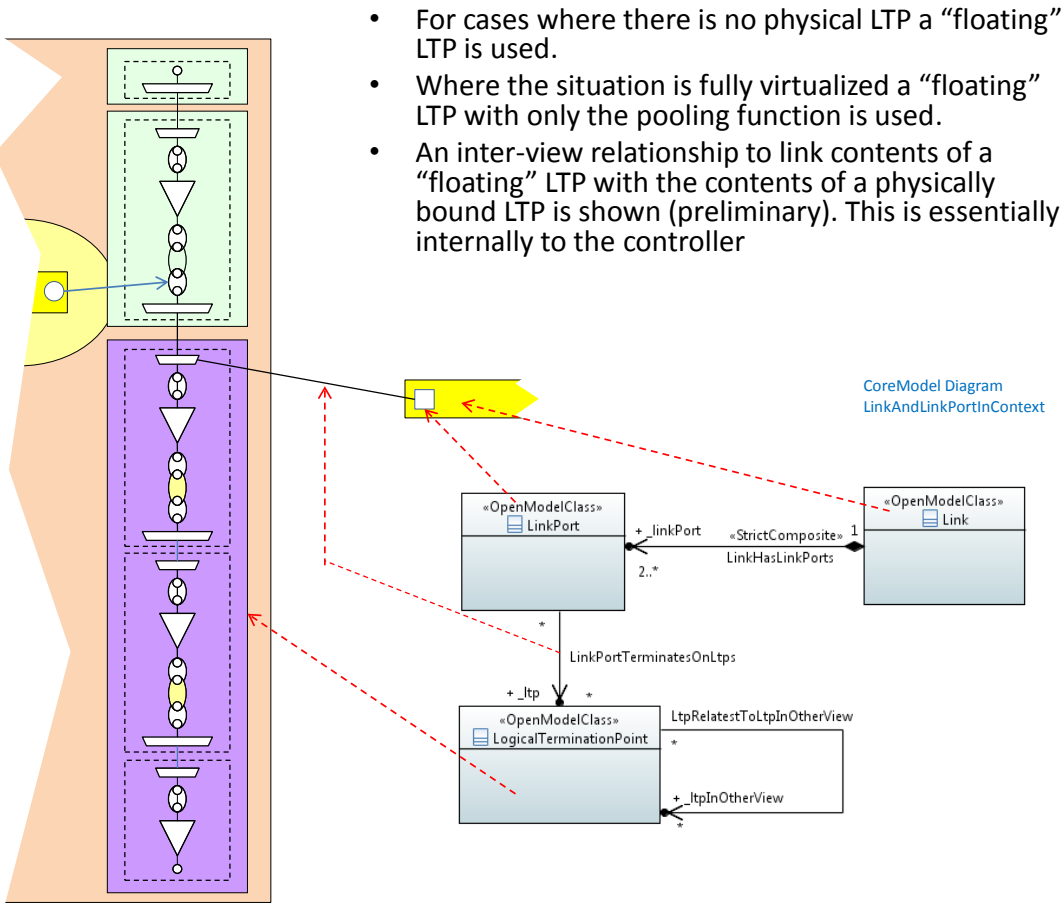


Figure 4-8 LTP "pooling" client LTPs

The figure above shows how the Link terminates on the LTP via the LinkPort.

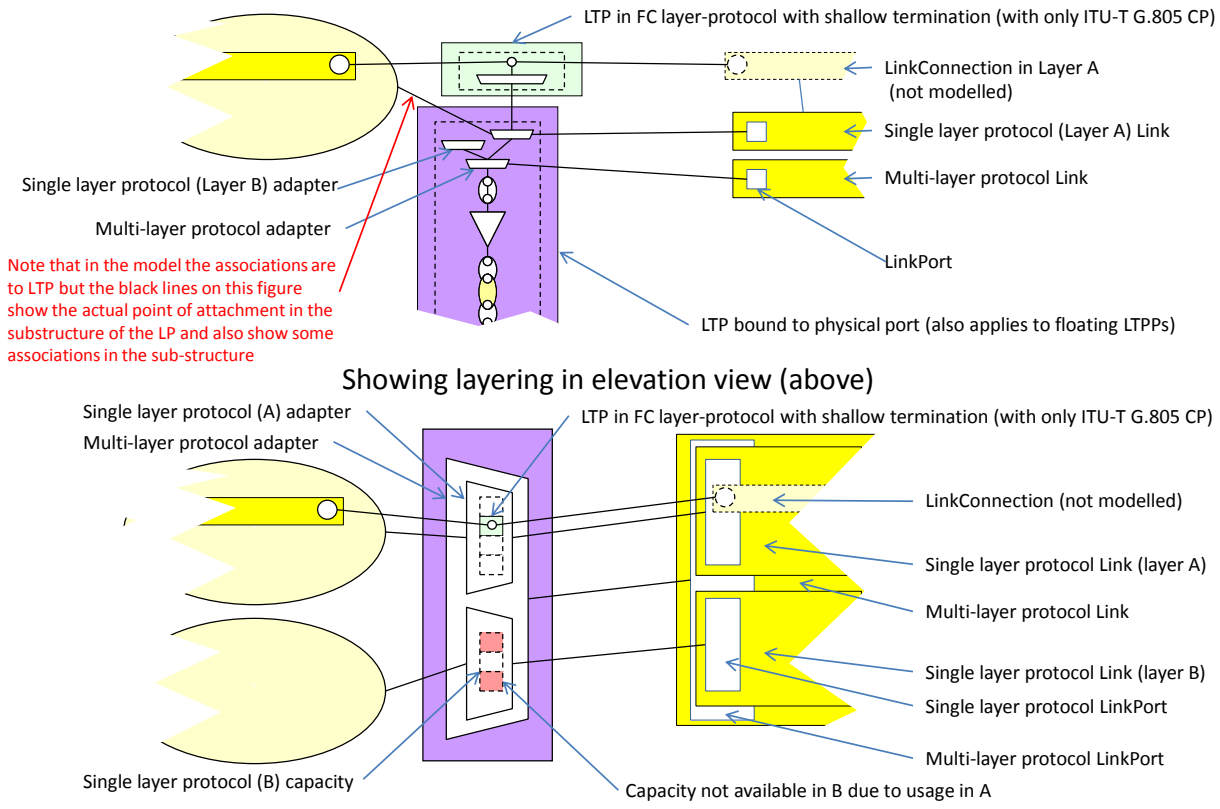


Figure 4-9 Views of Link, LinkPort and LTP showing LTP pooling

The LTP may have the capability[12] 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-ported 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.

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

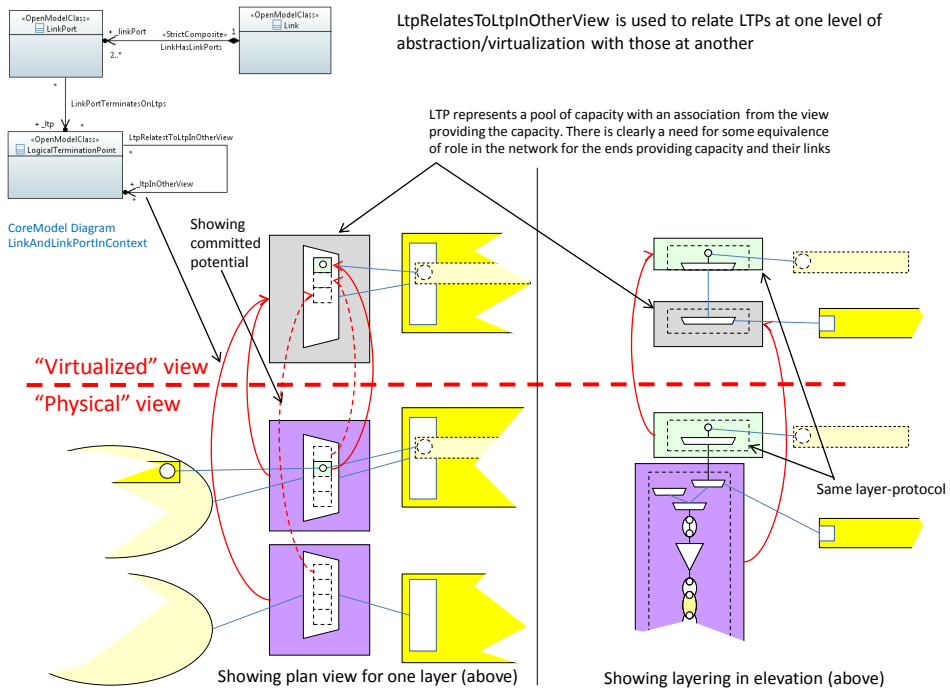


Figure 4-10 Views of "virtualization" [13] of LTPs with server side LTP representing a pool

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.

[13] 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 a physical equipment slot in a device.

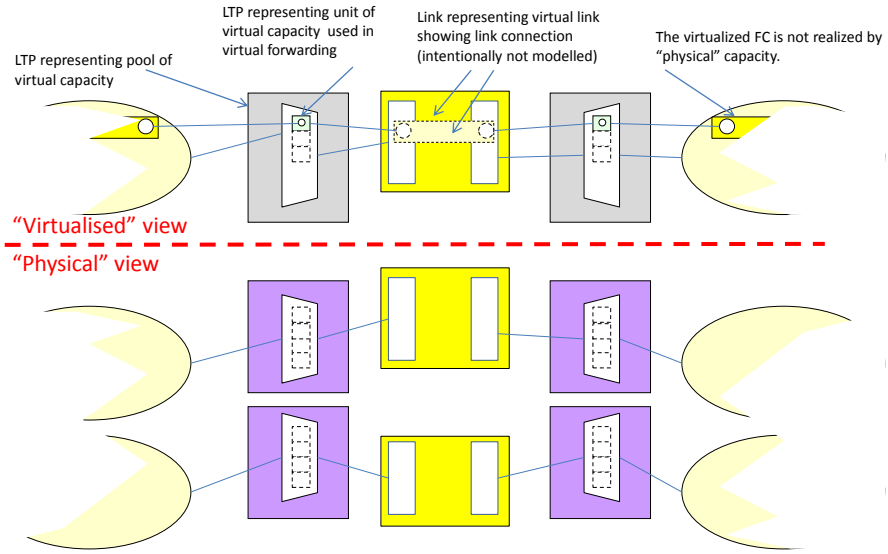


Figure 4-11 Starting condition

The operator then chooses to allocate capacity 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.

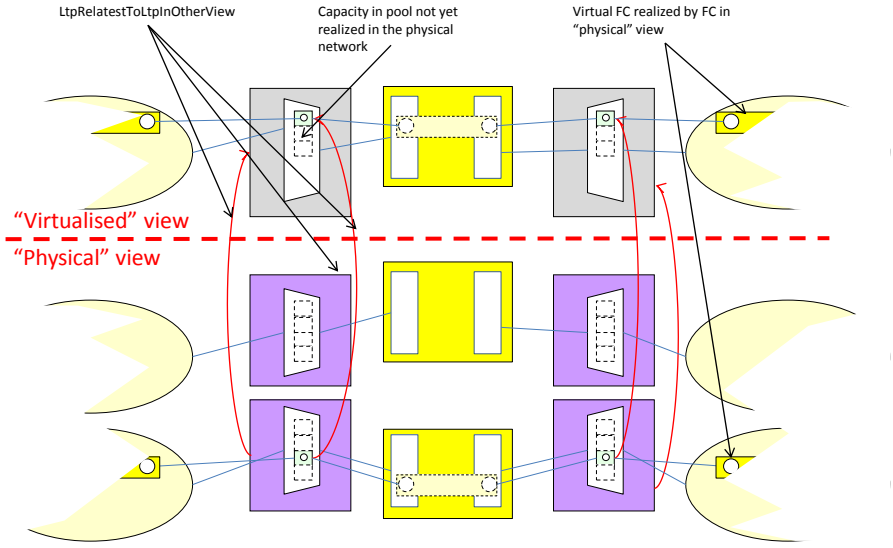


Figure 4-12 Resource allocation

The figure above 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.

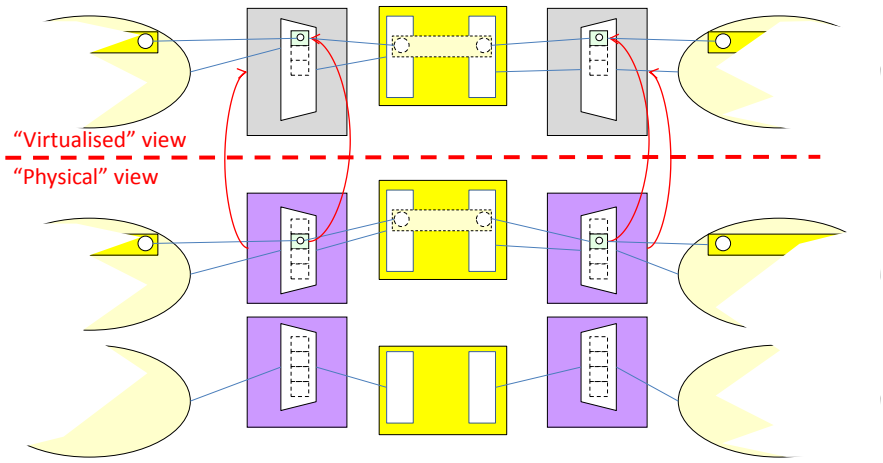


Figure 4-13 Move of allocation with no change to "Virtualized" view

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

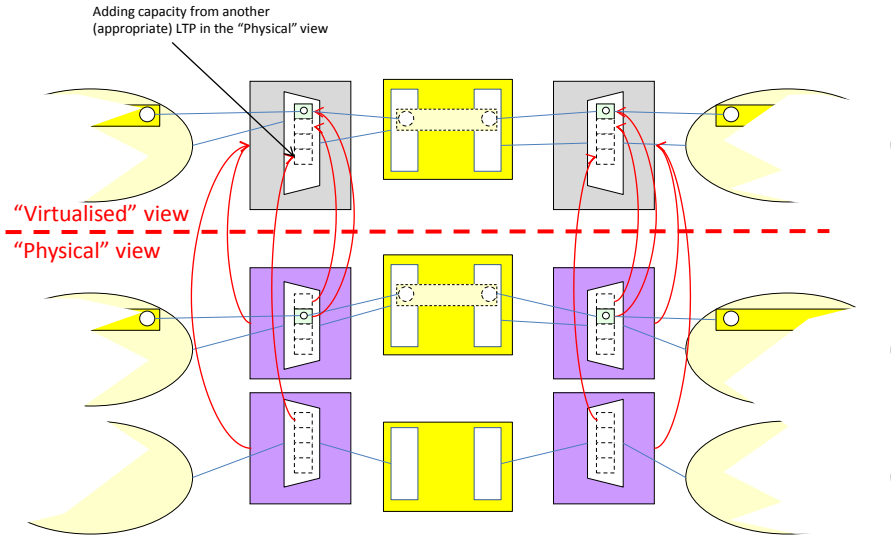


Figure 4-14 Capacity from server LayerProtocol Server LTPs

In the case set out in the figure above, there are two "Physical" view LTPs that are in the server layer of FCs shown. These two LTPs are associated with a single "Virtualized" view LTP representing a pool of capability to support the layer of the FCs shown.

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

4.4 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. In a common 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 shown 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. Then 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.

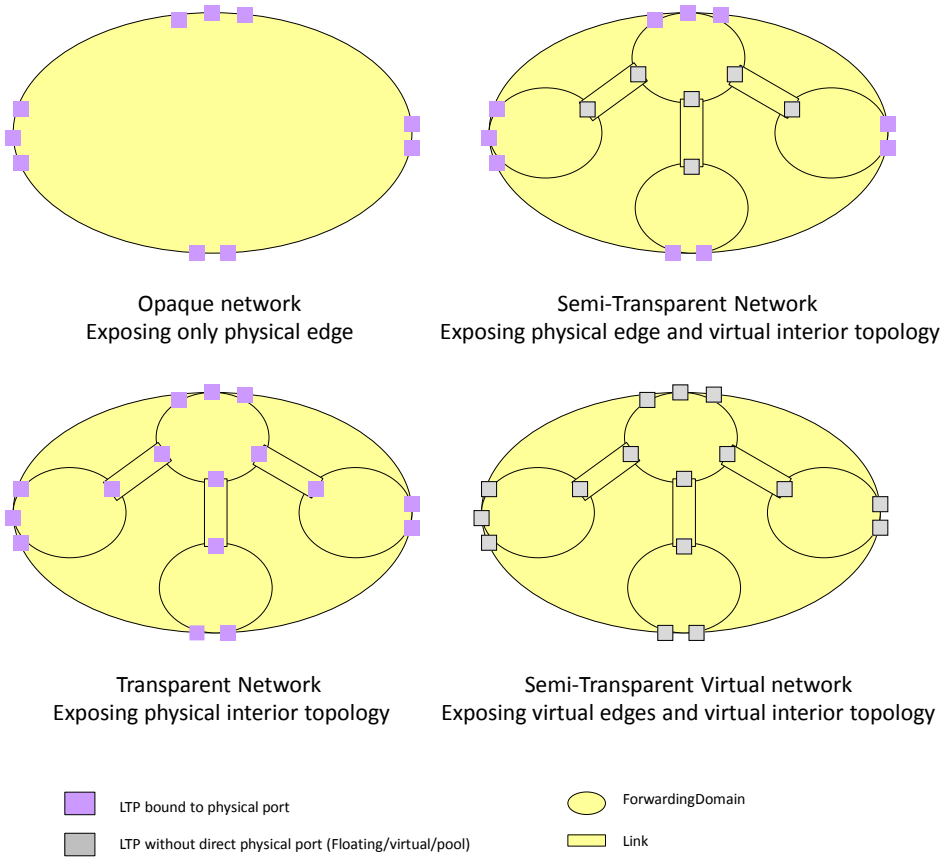


Figure 4-15 Various view boundaries

4.5 The FdPort

FdPort improves the representation of asymmetric FD capability and aligns the FD with the general Component-System pattern (see TR-512.A.2). For example an FD could have FdPorts with root role, FdPorts with leaf role and FdPorts that can be either such that only FCs that are Root-Leaf can be created and only in conformant orientations.

Limited use has been made of the FD port at this stage.

4.6 More on views and names/identifiers – The FC representing a Call

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. An entity may have several names and several identifiers. 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 figure below covers forwarding services. For other more complex services ProcessingConstruct will be required (see TR-512.11).

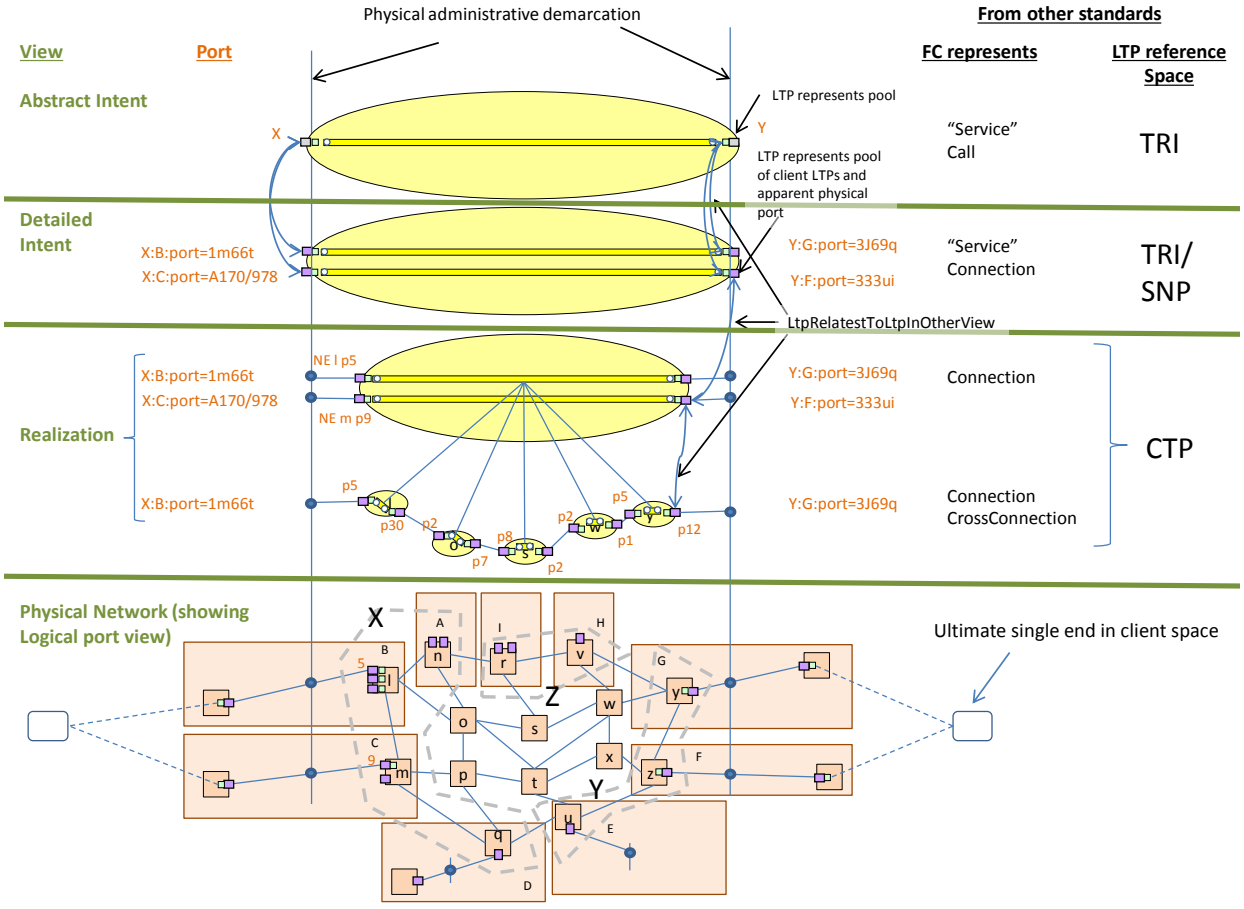


Figure 4-16 Various interrelated network views in a multi-party context

The most abstracted view (Abstract Intent[14]) shows the FC bounded by the network demarcation represents a "Service"[15] or Call[16,17] [ITU-T G.8081] in any state, from most abstract stage to fully operational state. The Call represents the intention to provide service and sometimes it can exist

[14] The term "Intent" is being used loosely here
[15] The usage of the term "Service" is intentionally vague here.
[16] 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).
[17] "A call is an association between two or more users and one or more domains that support an instance of a service through one or more domains." Based on the definition, the service used here only refers to connectivity service.

without connection. When the Call is installed and enabled, it can be used to provide forwarding service and function as an operational FC[18].

The Call tends to be specified in terms of relatively abstract constraints although it could be specified precisely if the shared model is suitably detailed. The cross connections are considered as being specified precisely but there can be optional parameters and the definition of the cross connections could be considered as constrains with respect to the underlying device.

The FC is, as usual, terminated by LTPs, which in this case are at the actual physical edge of the administration of the network. There is a two level hierarchy of LTPs shown where the lower (grey) LTP represents the pool of physical network access ports and the upper (green) LTP represents the per-"Service"/Call forwarding termination[19]. 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 Transport Resource Identifier (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. which is used to represent the parallel decomposition of the FC or "Service". 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[20].

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

Note: similar to the Call, crossconnection (XC) in the realization level can exist without supporting hardware in place, hence Call, Service and crossconnection are intention. If and only if all the XCs of FC are installed and operational, the FC is operational.

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.

[18] A Call can have different parameters from an underlying FC (e.g., resilience and quality. A Call in planning stage (e.g. in early stage of lifecycle) may not have a fully defined end point.
[19] The layering of the lower LTPs depends upon the variety of accesses and will not be discussed further here.
[20] This level may be decomposed into “connection” segments and there may be many recursions of abstraction decomposition.

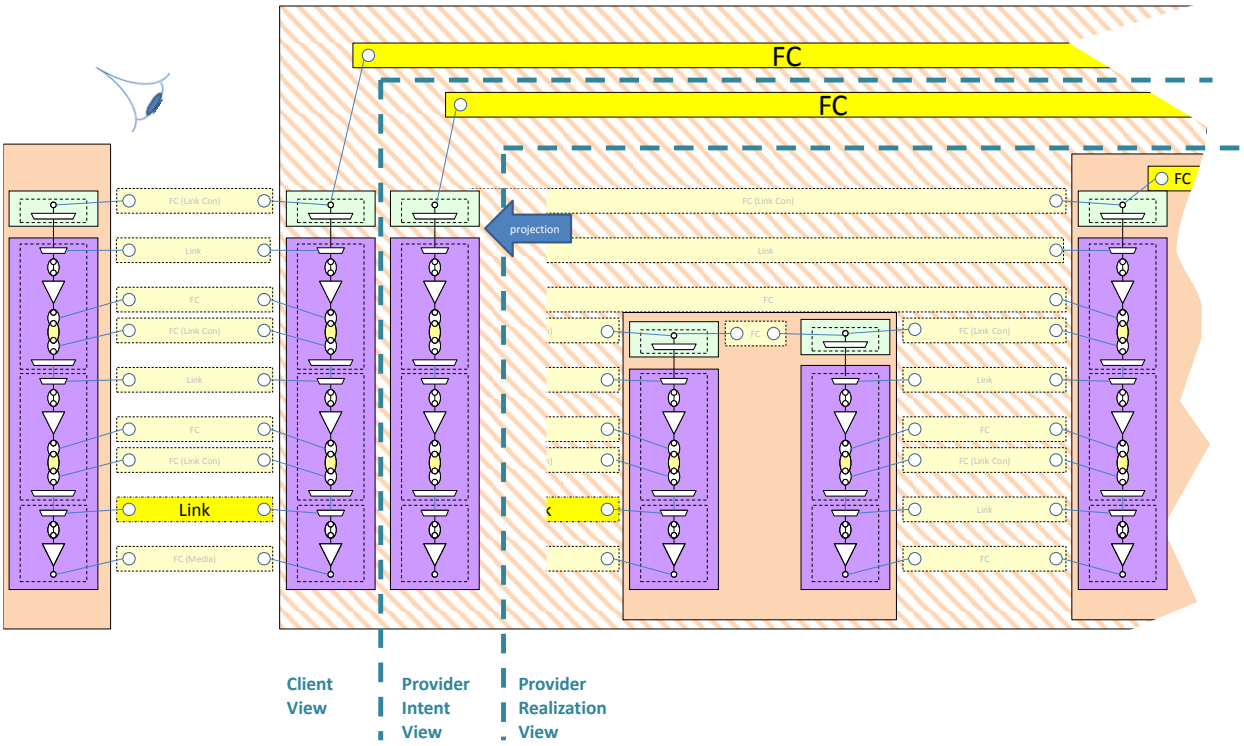


Figure 4-17 Complex network edge[21]

4.6.1 Call, Service, the Resource-Service Continuum and the Capability Continuum

The true service is the outcome/experience as perceived by the user e.g. the experience of adjacency. Providing a service is causing an outcome/experience for a client/user. The service achieved by a transport network is apparent adjacency such that the experience is that what is actually remote information appears local. Providing the service is achieving that outcome for the client/user.

The user could request an outcome/experience but actually usually requests provision of a capability to enable them to achieve an outcome/experience. Ethernet Private Line is a definition of capability to achieve the outcome of perceived adjacency. What is called Service is usually the capability to achieve the desired outcome and not a statement of the outcome.

Hence, the usual agreement is for a capability. The call/service is the capability described in the agreement between a client and a provider. The client will perceive the provided capability as a resource. As client-provider this has recently been called the Resource-Service Continuum. In current usage call, service and FC are all definitions of capability.

[21] The boundary could be considered as a MEF UNI.

The requested capability is itself achieved in terms of other capabilities where those capabilities are assembled in a structure. The structure is a system and the capabilities can be perceived as components making up that system.

There appears to be a generalized underpinning model of capabilities represented in terms of components and assemblies of components to form a system that can itself be viewed as a component (this is explored in TR-512.A.2).

Associated with the providing of Service is billing and a level of security. Both of these aspects are relevant to some degree for an FC at any level of view but are most relevant at a commercial boundary. So at a commercial boundary there is an expression of capabilities and what outcomes/experiences they may be used to achieve, there is an associated statement on pricing and also an appropriate statement on security.

The axis of the consideration is Capability. Both the resource models and the service models are refactored capability models. This leads to the notion of a Capability Continuum where the intention is to provide capability at each relevant client-provider demarcation.

Performance measures are measurements of achieved capability (in the context of the network technology). Performance measures do not necessarily relate to user perception. Mean Opinion Score (MOS) is a statement of the user experience. An individual only experiences a degradation of service if it the performance issue is apparent within their capability of perception and it occurs when something relevant is being transferred. A performance issue that is within perception capability of the user when something relevant is being transferred will impact the MOS[22].

4.7 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)[23] in the LinkPort.

[22] For example, in an audio call, a human has a particular round trip delay they can detect (in terms of 10s of ms)
[23] Note that the attribute in the model is «Experimental»

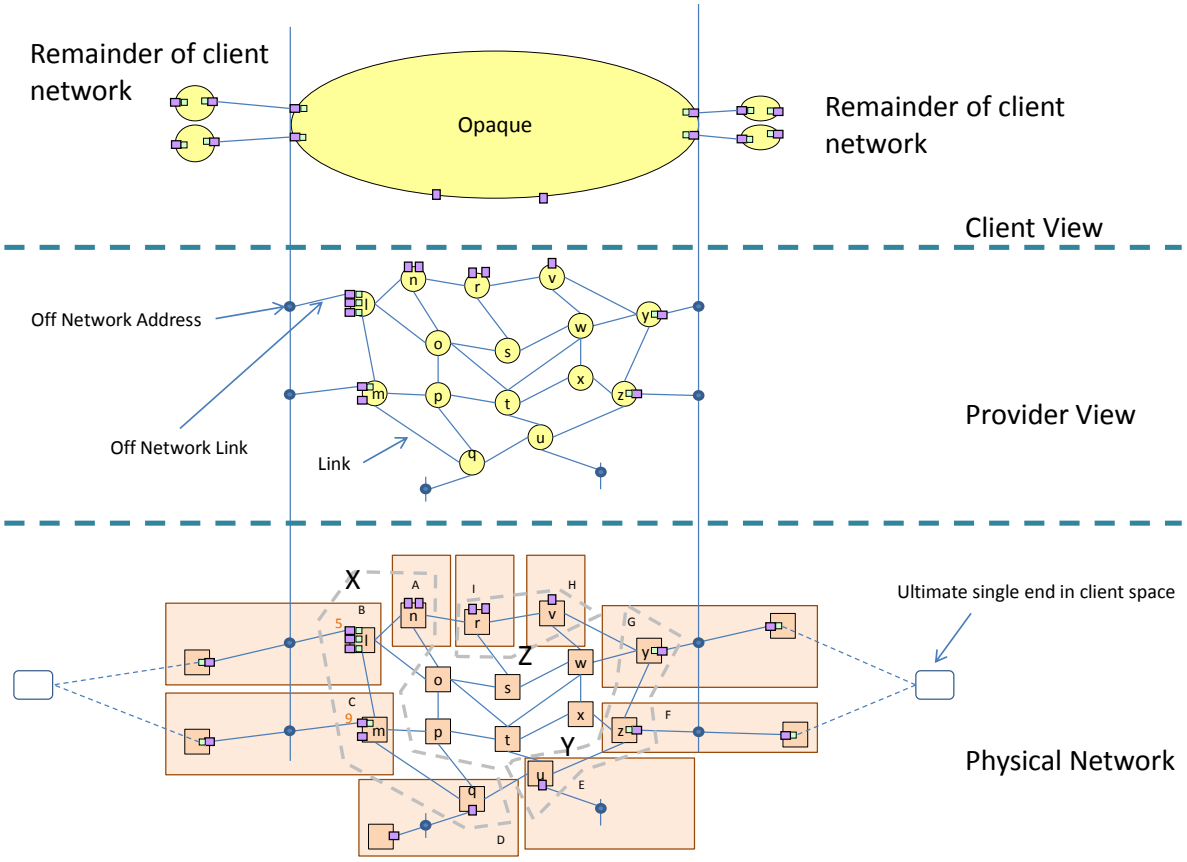


Figure 4-18 Complex network edge

4.8 Serial-Compound Links

Essentially the serial-compound Link is formed by a serial assembly of Links and Forwarding Domains. However there are significant complexities that need to be discussed.

From the user's perspective a serial compound Link is a Link. Serial compound describes the realization. A serial compound Link is an adjacency that is realized by a number of server layer forwarding constructs which terminate to the client layer at various intermediate points. The arrangement is potentially complex although for a simple point to point Links the server layer Links form a chain.

In a serial compound Link, the client layer terminations are encapsulated in the exposed Link. For the Link to be realized, these client layer terminations must bound ForwardingDomains in such a way that a mesh of Links and ForwardingDomains is formed between the bounding points of the Link and such that forwarding can be enabled between the appropriate points of the Link. There could be complex protection etc. encapsulated.

For the exposed Link to be used, appropriate client layer forwarding constructs need to be created in the ForwardingDomains so as to provide the enabled adjacency. Where there are multiple potential clients in addition to the ends needing configuration the intermediates will also need configuration of adapters as well as configuration of intermediate forwarding constructs.

The figure below shows an opaque network between two access points in a client's view, where the network is controlled up to and including the client device, i.e. the controller of the network (depicted as a cloud) had control of the adapters at the two access points and can determine that a connection has been made in the FD. The client is assumed here to be operating with the same layer protocol, LP-X, at both ends.

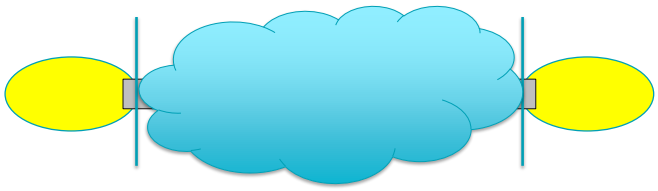


Figure 4-19 Opaque network between two access points

The client layer may require an adjacency between the two access points as shown below and may wish to view this as a Link.

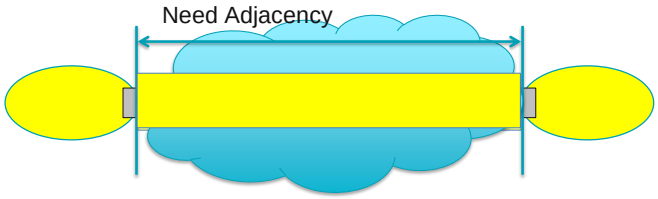


Figure 4-20 Simple adjacency between two access points viewed as a Link

But the network between the two access points operates at the same layer protocol, LP-X as the client such that the network combination looks as shown in the figure below where all FDs are in LP-X.

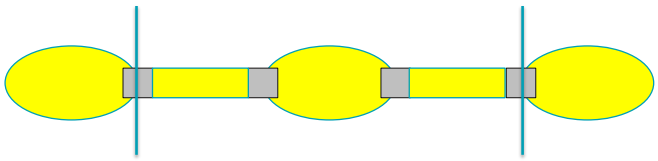


Figure 4-21 Network between two accesses in the same layer protocol as the access

To achieve the apparent adjacency it is necessary to create FCs in the intervening FD and configure the LTPs. In this example there is one FD but there could be many and the separate

FCs could take different, potentially resilient, routes across the network so long as the effect at the access was as if there was a Link. The one restriction is that the signal instance identifier (channel, VID, Wavelength etc.) must appear unchanged.

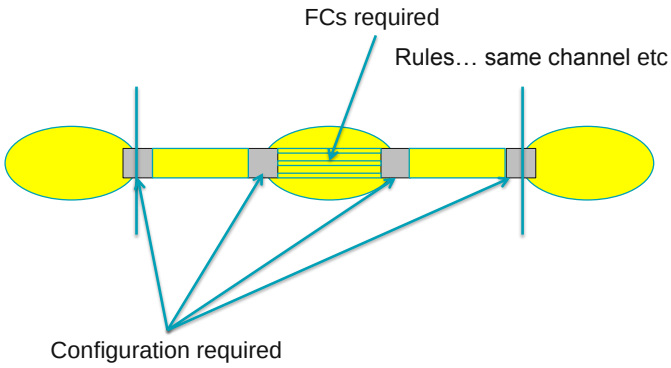


Figure 4-22 The realization of an apparent Link

In normal usage the FCs would be defined and determined in an initial negotiation for Link capacity. These FCs would then be set up and activated in the network. The FCs would remain unchanged through usage providing dedicated capacity that is always available for the client. The client could negotiate additional Link capacity up to the capacity limit of the access points (assuming the network has sufficient capacity to support this. Clearly the client could also negotiate a reduction etc.

Monitoring at intermediate points on the FCs supporting the serial compound Link will depend upon application of client traffic. The provider of the Link will not be aware of when traffic is being applied and when not. Some form of partial span maintenance (e.g. Tandem Monitoring, MEPs/MIPs etc.) may be necessary to provide meaningful and continuous measures

In this case, as the provider's controller of the Link can gain information directly from the access points, it can potentially determine when traffic is applied and hence when to set up measures. In addition, it is possible that the information available could allow the controller of the Link to determine when to set up the FCs in the network to support the Link so that the client sees the desired Link as shown below, although capacity supporting the Link is not committed until a corresponding client FC is applied at the access point.

This dynamic capability clearly requires "real time" behavior from the controller but could allow more efficient usage of network resources and Serial Compound Link the configuration of a ForwardingConstruct will require the use of a (possibly trivial) routing function. Note that his dynamic capability is not defined in [ITU-T G.800].

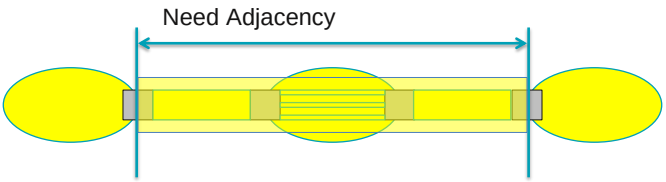


Figure 4-23 The Link resulting from the underlying configuration

The figure below shows the classes and associations that are relevant in the modeling of the serial compound Link (the key associations are highlighted in red and blue).

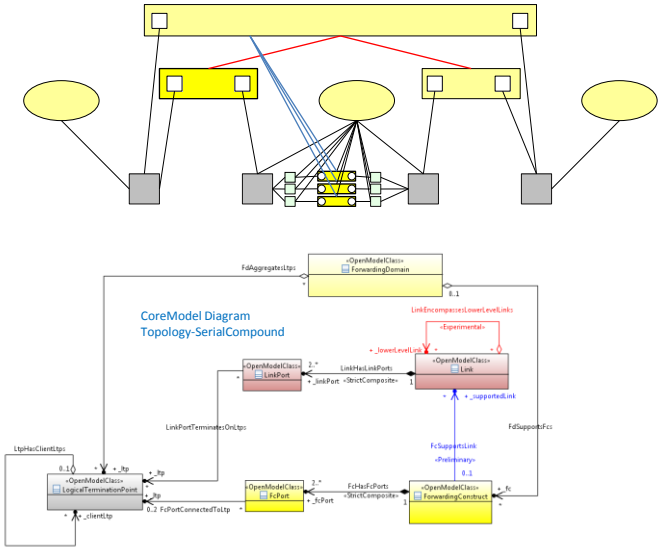


Figure 4-24 Serial compound Link showing model

In the case above it was assumed that the controller of the network had information from the access points through direct control. In the case below it is assumed that there is a more common interconnect where the demarcation is at some point along a cable and not inside a device.

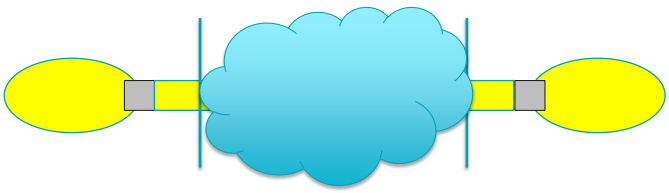


Figure 4-25 Network demarcation at some point along a cable

Again, the network can be considered as below but on this occasion with the demarcation cutting the access Links.

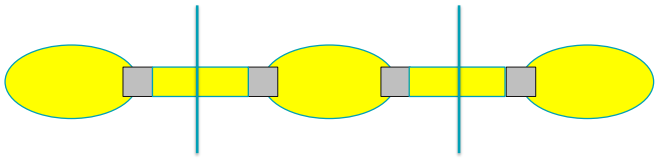


Figure 4-26 Network demarcation showing network detail

As discussed above, to achieve the appearance of a Link to the client at the access points it is necessary to configure the FCs and LTPs within the network as shown below.

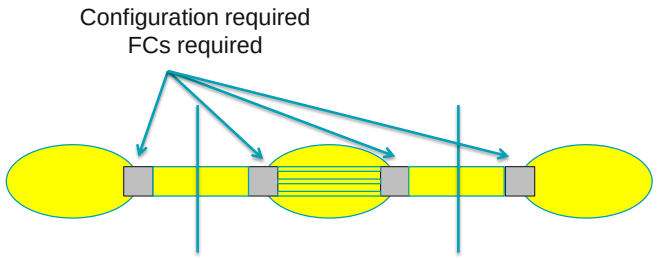


Figure 4-27 The realizing of an apparent Link in a mid-cable demarked network

As can be seen in the figure above, both the client and the network operator need to apply configuration to cause the client perception of a Link.

However, considering the dynamic case, the network operator is unaware of when the client has applied configuration unless there is some form of signaling from the client to the operator to cause the configuration to be applied and the network to provide an apparent link to the client (as shown below.

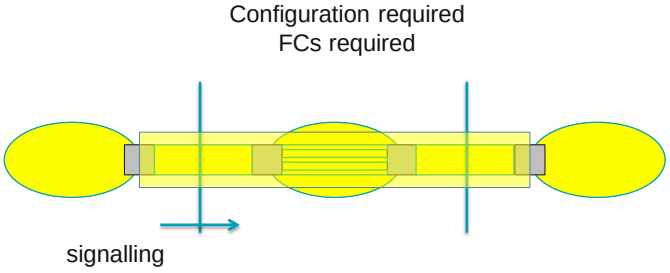


Figure 4-28 Signaling is necessary to provide dynamic operation of the apparent Link

BUT there is no signalling defined for dynamic serial compound Link support. On that basis, only a static serial compound Link is achievable. All FDs in the layer of the service must have been committed and configured to achieve the Link. This appears to apply to all layers.

Whilst it may be useful for path computation to use the serial compound Link without a pre-configured ForwardingDomain to determine a route, it is not clear that serial compound Links should be used directly for evaluation of dynamic forwarding control. To use a serial compound Link requires the forwarding control application to have knowledge of the resources/topology that supports the Link via the server layer. It appears that the "raw" form rather than a serial compound Link (i.e. the concatenation of Links and ForwardingDomains) must be exposed directly for the purposes of connection management.

If dynamic behavior is required with no pre-allocation of resources, it is recommended that:

- The network is represented to the client as a ForwardingDomain with short access Links
- A mechanism is provided to configure the forwarding service via the creation of ForwardingConstructs
- The ForwardingDomain is exposed with constraints such as:
  - Same channel
  - Whole port forwarding group
  - Limited capacity
- In complex cases a mesh of ForwardingDomains and Links is exposed to detail the complex constraints
- The client may be offered a self-service facility via a management-control interface

The figure below shows a sketch of the network exposed to the client in the flexible case. In the figure the client has configured 3 FCs.

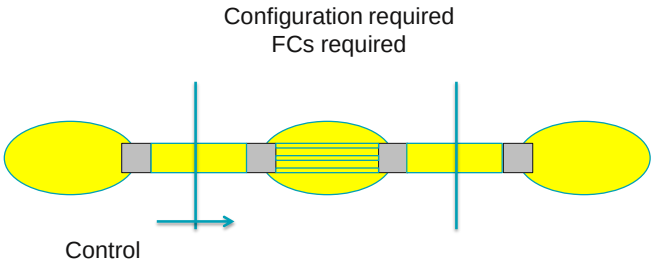


Figure 4-29 Network exposing configurable FD to client and providing self-service control

4.9 Transitional Links

As previously discussed, a topology is formed from FDs and Links. A topology represents capacity for forwarding for a single layer-protocol. The Links in one layer are supported by forwarding in a server layer. The interconnection between topologies at a layer-protocol boundary is represented by an LTP. When the server layer is not committed there is no Link. There may be uncommitted LTPs that provide access to the server layer that could provide capacity.

Where there is a need for additional capacity, due to usage, the traditional approach would be for the controller of the layer with the need for capacity to request capacity between points in the server layer. The server layer may present some form of cost matrix for interconnection between different server access points. Whilst this may provide a solution, the approach is suboptimum in many cases and requires significant preparatory calculation in the server layer to account for potential demand that may not materialize. In resilient dual homed scenarios there may be a need to recalculate costs regularly.

An alternative is to view several topologies for different layer-protocols that are interconnected by LTPs as one single multi-layer-protocol topology. This can provide more efficient calculation that provides results that are closer to optimum. Like all network topologies the multi-layer topology view is formed only from FDs and Links. On that basis, all LTPs at relevant layer-protocol boundaries will need to be transformed into corresponding Links when forming the view. The type of Link that is formed in place of an LTP is called a Transitional Link as it represents one or more layer boundary transitions. The Transitional Link has special properties related to the layer-protocol transition.

The following figures work briefly through the use of a Transitional Link in a path computation context.

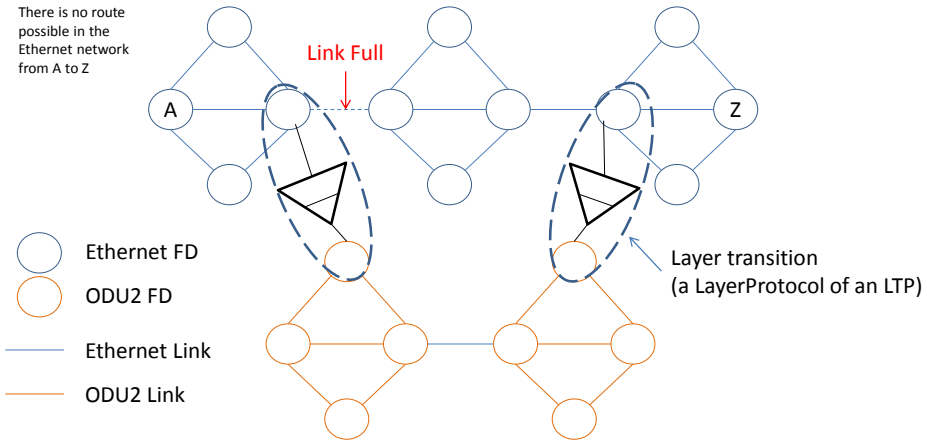


Figure 4-30 Sketch of two network layers

In the figure above there is no available route between A and Z but there is capacity in the server layer.

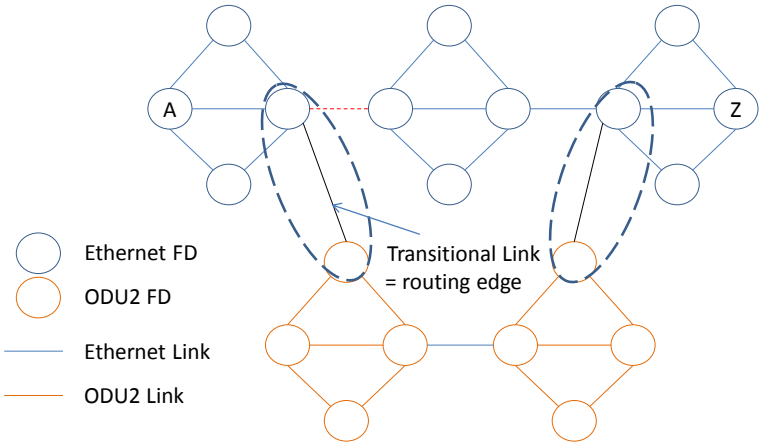


Figure 4-31 Forming the Transition Links

All LTPs between the two layers are converted, in the view, into Links with the layer transition properties (i.e. Transitional Links) as shown above for the two LTPs in this trivial network.

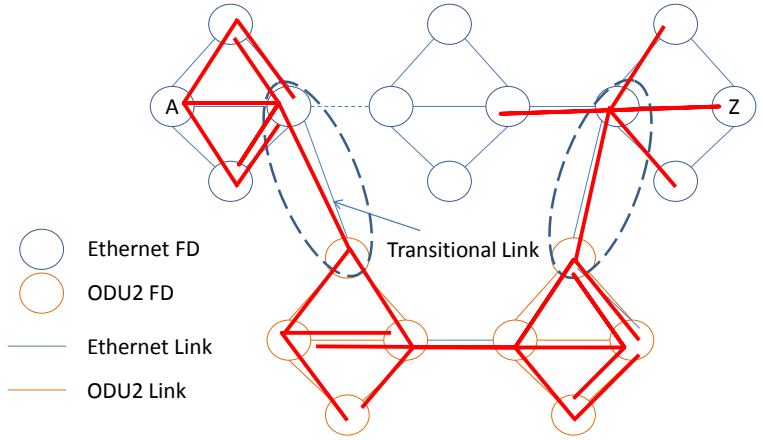


Figure 4-32 Path computation finds available routes through the topology

A path computation algorithm is run to explore the network of Links and FDs as shown in the sketch above. It finds available routes as shown in the sketch below.

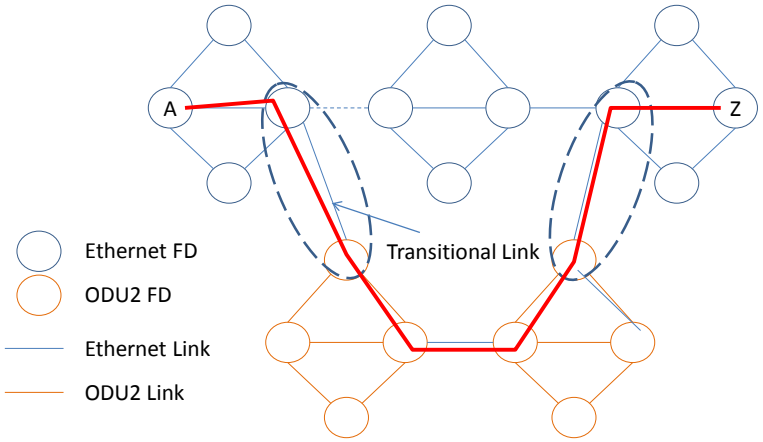


Figure 4-33 Path computation finds available routes through the topology

The Transitional Links, in the view, are converted back to LTPs and the details for the resulting paths in the two layers are deployed in terms of FCs etc.

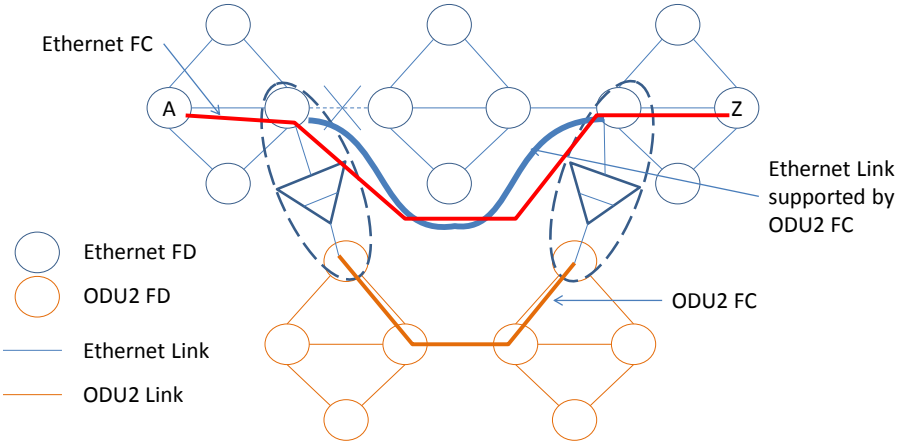


Figure 4-34 Two FCs are created, one in the ODU2 layer and one in Ethernet

The discussion above deals with relatively simple single layer Transitional Links that are within one device. More complex Transitional Link cases are for further study (see section 5.5).

4.10 Multi-Port Link

A Link may have more than two LinkPorts. The QinQ over PBB example below (derived from material in [TMF TR215]) shows a number of FCs, all of which are symmetric. Note that there is per flow behaviour associated with the points that ensures the traffic is forwarded correctly (this would be explained in the definition of "C" FcPorts).

The lower part of the figure shows the multi-ported Link (Fb) supported by the B MAC layer FC (which is not shown but is coincident with the Link).

The key focus of the example is the FC between Device C, D and E which represents the link connection. This is a client (QinQ) granularity FC that is narrower than the server Link (shown

in the lower figure) where the narrowing is controlled by the intervening ISID identifier (which is at the QinQ granularity).

The blue dashed arrows show the relevant server LTP to client LTP relationships. There are many client LTPs per server LTP. The figure below shows an example of one set of instances. The client "link connection" is only between C, D and E and hence nodes X, Y and Z are not visible in the specific QinQ instance above.

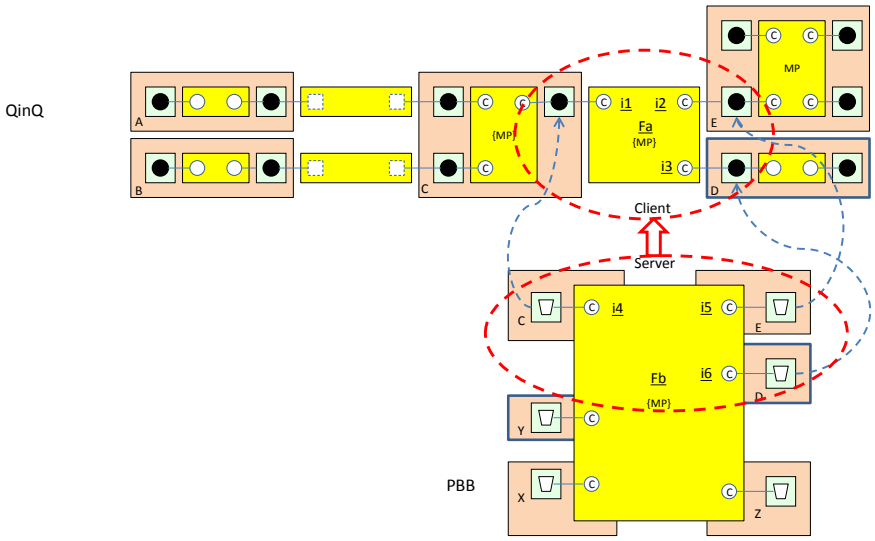


Figure 4-35 Multi-ported Link supporting an FC

This complex case in the figure below derived from material in [TMF TR215]) shows Interconnect ({IC}) protection with roles Resilient, Bridge and Protection and Double Add-Drop ({DAD}) with roles Main and Standby (where the roles are in pairs (left M/S pair and right M/S pair)).

The figure below shows the Higher Order (HO) path across the Shared Protection Ring connecting the terminations in B, C, D and E that support the Lower Order (LO) LTPs shown in the upper part of the figure.

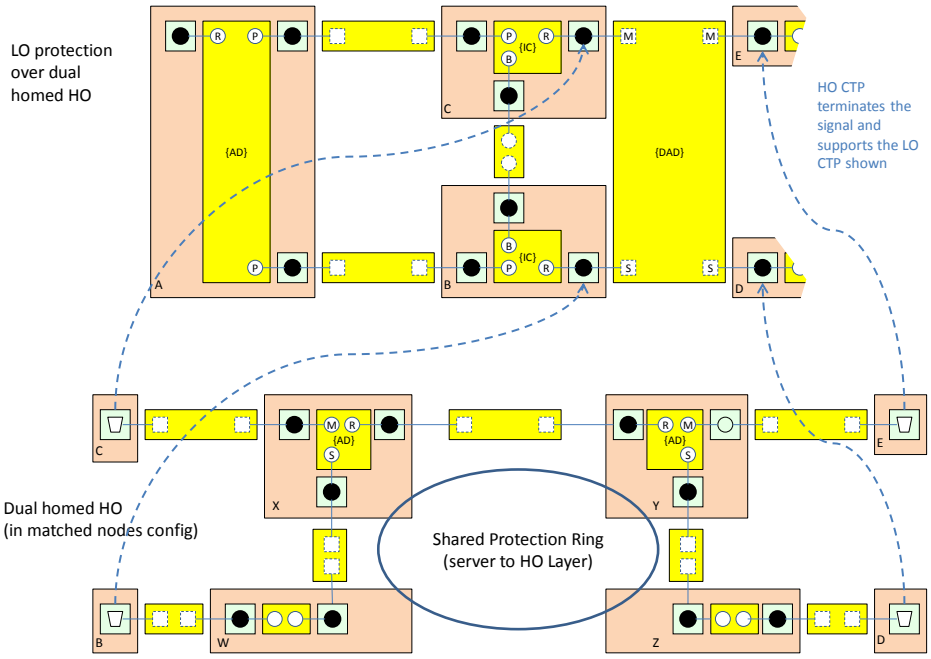


Figure 4-36 Multi-ported Link supporting supported by a complex server configuration

The key feature of the figure above is the Multi-Ported Link, "{DAD}", that reflects the characteristic of the server FC (B&C-D&E). In this example the adapters in B, C, D and E are assumed to have the same capability. In the general case the Link capability is determined by the intersection of the capabilities of the adapters at the ends of the server FC.

The application of the parameters from the ForwardingEntity is for further study.

4.11 State Dependency

The client-server aspect of Topology, Termination and Forwarding dictates essential dependency between the Lifecycle states of entities. This can be explained with the figure below that shows the effect of serial elements in the hierarchy. The arrows in the figure show the direction of direct dependency (the arrows point at the dependent entity).

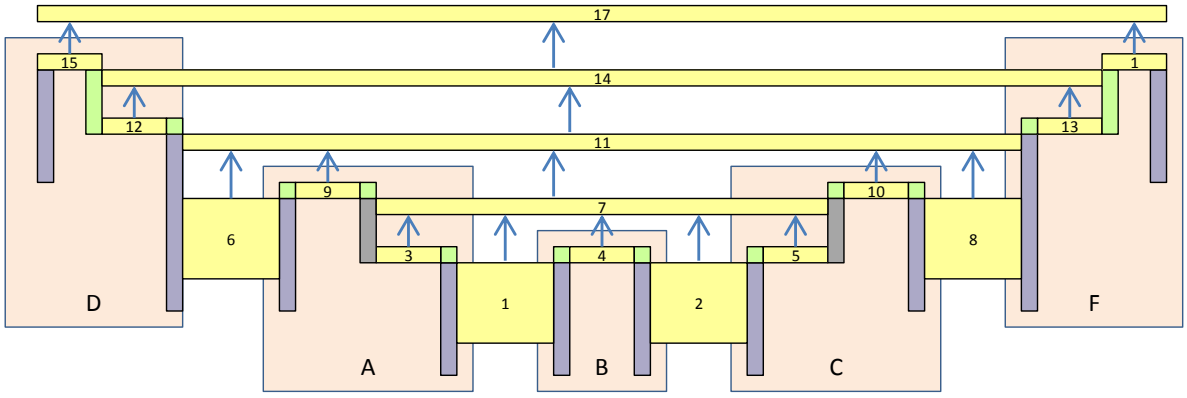


Figure 4-37 Lightweight sketch of a multi-layered network

The following table provides a view of state dependencies (see TR-512.3).

Table 14: Client-Server State interaction

Server State			**Dependent Client State**		
Lifecycle	**Admin**	**Operational**	**Lifecycle**	**Admin**	**Operational**
PLANNED	LOCKED	DISABLED	PLANNED	LOCKED	DISABLED
POTENTIAL_AVAILABLE	UNLOCKED	DISABLED	POTENTIAL_AVAILABLE/ POTENTIAL_BUSY/ PLANNED	Any	DISABLED
POTENTIAL_AVAILABLE	SHUTTING_DOWN	DISABLED	POTENTIAL_BUSY/ PLANNED	SHUTTING_DOWN/ LOCKED	DISABLED
POTENTIAL_AVAILABLE	LOCKED	DISABLED	POTENTIAL_BUSY/ PLANNED	LOCKED	DISABLED
POTENTIAL_BUSY	LOCKED	DISABLED	POTENTIAL_BUSY/ PLANNED	LOCKED	DISABLED
INSTALLED	UNLOCKED	ENABLED	Any	Any	Any
INSTALLED	UNLOCKED	DISABLED	Any	Any	DISABLED
INSTALLED	SHUTTING_DOWN	ENABLED	Any	SHUTTING_DOWN/ LOCKED	Any
INSTALLED	SHUTTING_DOWN	DISABLED	Any	SHUTTING_DOWN/ LOCKED	DISABLED
INSTALLED	LOCKED	DISABLED	Any	LOCKED	DISABLED
PENDING_REMOVAL	SHUTTING_DOWN	ENABLED	PENDING_REMOVAL	SHUTTING_DOWN/ LOCKED	Any
PENDING_REMOVAL	SHUTTING_DOWN	DISABLED	PENDING_REMOVAL	SHUTTING_DOWN/ LOCKED	DISABLED
PENDING_REMOVAL	LOCKED	DISABLED	PENDING_REMOVAL	LOCKED	DISABLED

4.12 Inverse Multiplexing

It is sometimes necessary to carry a single information flow that has a particular characteristic rate over a network where the bearers are too small to carry that rate of information transfer. Under these circumstances it is necessary to use a mechanism that divides the information flow into parts to be conveyed over several of the bearers in parallel such that it can be reassembled at the far end of the bearer into a flow that is indistinguishable from the original.

The dividing of an information flow into parts is called Inverse Multiplexing. There are a number of different schemes for inverse multiplexing (Link Aggregation Group (LAG), Virtual Concatenation (VCAT) etc.). Some schemes take advantage of other characteristics of the information flow such as the packet nature. The scheme provides distinct properties and also distinct measures. Regardless of the specific scheme the essential model is the same.

In the case of the LAG it is possible to use some of the bearers to protect others by simply over-provisioning. Again, this does not change the essential model but may change the encapsulation and certainly affects the parameters and measures.

In the figure below:

- The "Expanded Representation" diagram shows a view of the essential model of Inverse Multiplexing as an arrangement of basic generalized functions.
    - The FC is shown with a selector that operates at signal rate selecting fragment by fragment from different inputs (where the fragments may be packets, frames, frame fragments) and feeds this as a stream towards the client.

  This form is overly complex and there is opportunity for simplification
- The "Encapsulated FC and CSC" diagram shows the chosen simplified form where the C&SC and the FC have been encapsulated in the LTP

- This encapsulation could be exposed within the spec of an LP of the LTP or could be summarized as attributes of the LP of the LTP

This is the model for Inverse Multiplexing

- There are two specific cases shown dealing with different multiplicities
  - 'n clients and n "channels" on the server' shows the use of the full "Encapsulated.." model
  - '1 client and 1 "channel" on the server' shows the most reduced form

The most likely case is 'n clients and 1 "channel" on the server.

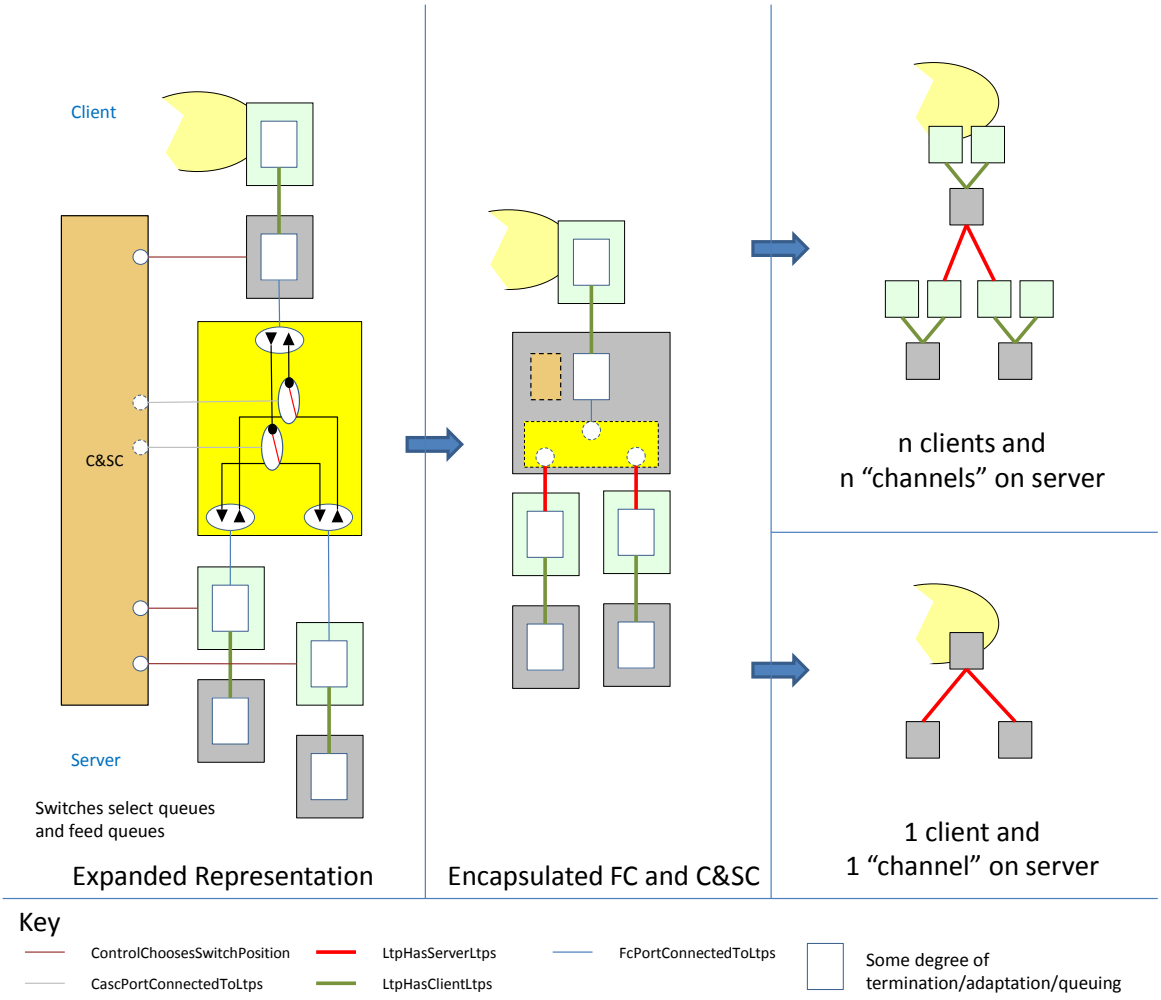
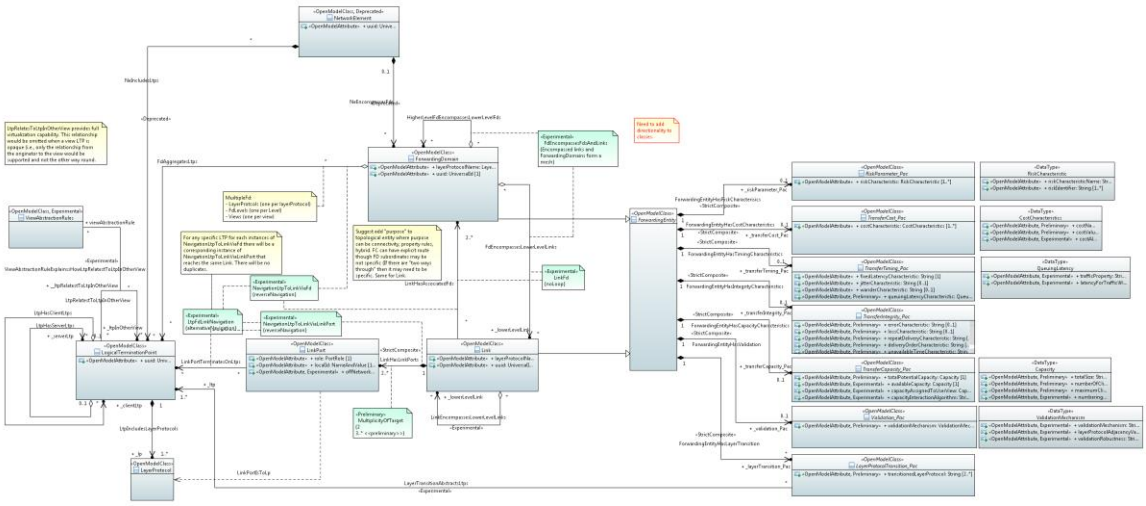


Figure 4-38 Representing Inverse Multiplexing

5 Work in progress (see also TR-512.FE)

5.1 Detailed properties of Topology

The topological components are assembled into systems and then encapsulated into topological components. The rules related to these assemblies are explored in the following figure. This also relates to discussion in TR-512.A.2. This area needs to be developed further.



CoreModel diagram: Topology-DetailWithRules

Figure 5-1 Topology details with rules

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

5.2 Cost algorithms

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).

5.3 FC/Link Convergence

Aim to fully align the two models and look for improved derivation.

5.4 NearEnd/FarEnd, Input/output and ingress/egress

Need to consider aligning input/output used in directionality with ingress/egress used in the switch spec model and to consider developing a model of Near/Far end.

5.5 Complex Transitional Links

The figure below shows two forms of Transitional Link.

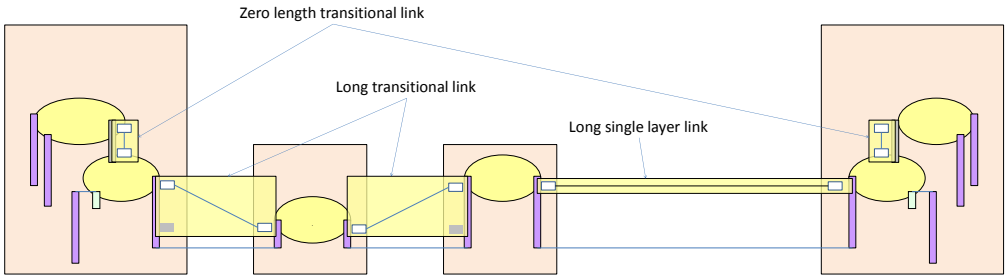


Figure 5-2 Two forms of complex Transitional Link

The figure below shows a particularly complex partial configuration where some of the server capacity has been committed to the client but further capacity remains. There are several transitions within one Link.

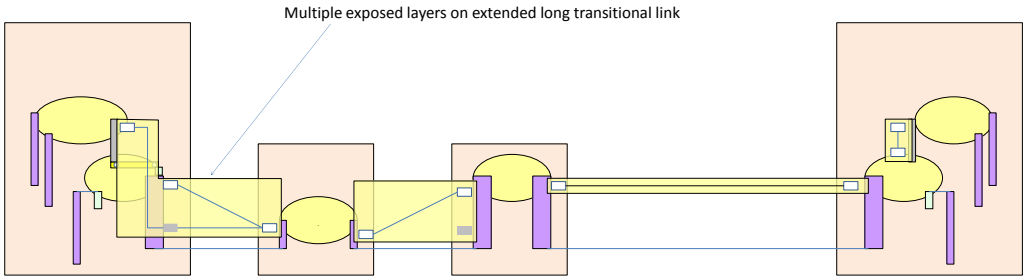


Figure 5-3 A particularly complex multi-layer multi-port Transitional Link

The figure below shows an off-network Transitional Link

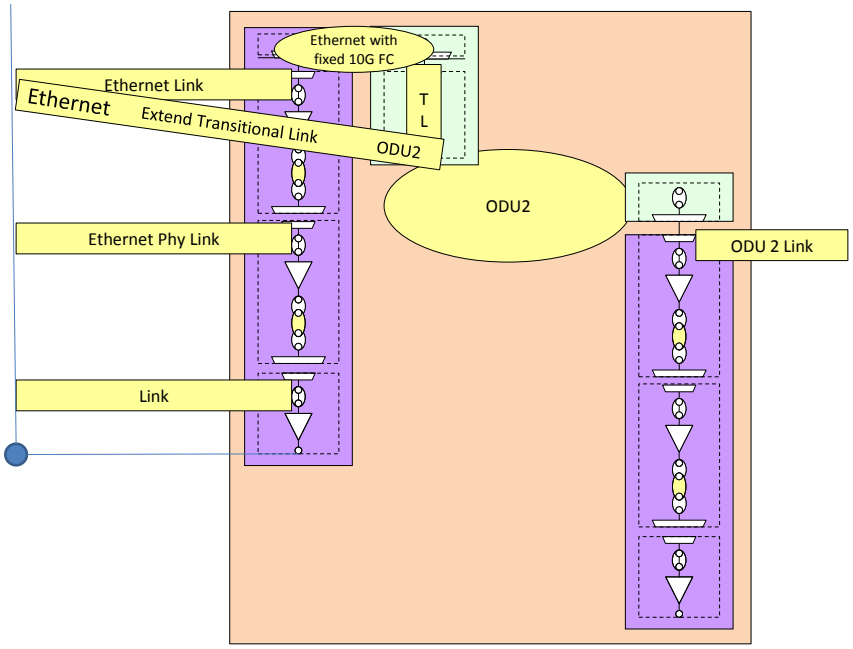


Figure 5-4 Off-network Transitional Link

The above cases are for further study. The model would appear to support the cases shown above.

5.6 Non-orthogonal FDs

The model does not impose a single hierarchy of FD. An FD is not decomposed into smaller parts, instead an FD is an aggregation of smaller parts. Several FDs may have some common FDs that they aggregate. This allows for a traditional single hierarchy but also allows more sophisticated structure. A more detailed explanation of this needs to be added to this document.

End of document