OPEN NETWORKING FOUNDATION

Core Information Model (CoreModel)

TR-512.2

Forwarding and Termination

Version 1.2
September 20, 2016

OpenFlow

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

Disclaimer

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

Table of Contents

List of Figures

Document History

Version	Date	Description of Change
1.0	March 30, 2015	Initial version of the base document of the "Core Information Model" fragment of the ONF Common Information Model (ONF-CIM).
1.1	November 24, 2015	Version 1.1
1.2	September 20, 2016	Version 1.2 [Note Version 1.1 was a single document whereas 1.2 is broken into a number of separate parts]

1 Introduction

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

1.1 References

For a full list of references see TR-512.1.

1.2 Definitions

For a full list of definition see TR-512.1.

1.3 Conventions

See TR-512.1 for an explanation of:

- UML conventions
- Lifecycle Stereotypes
- Diagram symbol set

1.4 Viewing UML diagrams

Some of the UML diagrams are very dense. To view them either zoom (sometimes to 400%) or 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 Core Network Model

The focus of this document is the key parts of Core Network Model of the ONF-CIM. The Core Network Model covers the essentials for modeling of the Network providing all of the key classes.

The CoreNetworkModel encompasses all aspects of Termination and Forwarding. The focus of this document is:

- Termination aspects of the CoreNetworkModel covering the modeling of the processing of transport characteristic information, such as termination, adaptation, OAM, etc.
- Forwarding aspects of the CoreNetworkModel covering the details of forwarding entities

The Core Network Model also encompasses a number of other areas which are covered in detail in related documents:

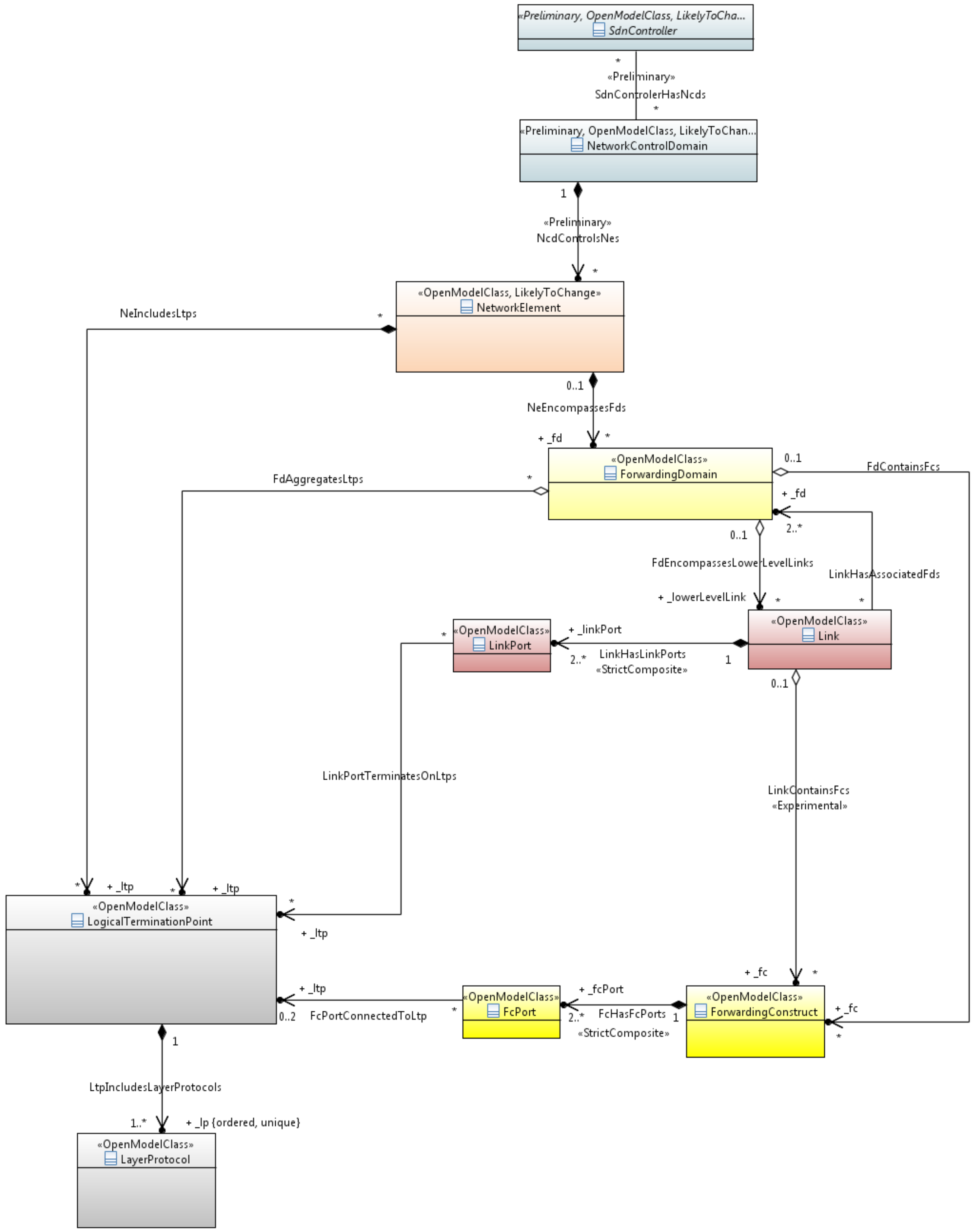
- Topology (see TR-512.3) covering the modeling of network topology information in detail[1] and describes the attributes relevant when working with multi-layered network topology.
- Resilience (see TR-512.4) covering the modeling of switches and configuration/switch control

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

3 Forwarding and Termination model detail

The Forwarding and Termination model is at the heart of the CoreModel. The figure below provides a view of the structure of the model. Further structure related to other aspects of the model is provided in other sections (especially relevant areTR-512.4 ONF Core IM – Topology and TR-512.5 ONF Core IM – Resilience. The diagram below highlights key interrelationships between key classes defined in the CoreNetworkModule of the CoreModel. The classes are colored to help recognize key groupings in the model. The colors are chosen to match the key entity colors in the diagram symbol set referenced in section 1.3 Conventions on page 5 (with the Link in the alternative color for clarity). This color scheme for class diagrams is used in some of the later figures.

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



CoreModel diagram: Forwarding-SkeletonOverview

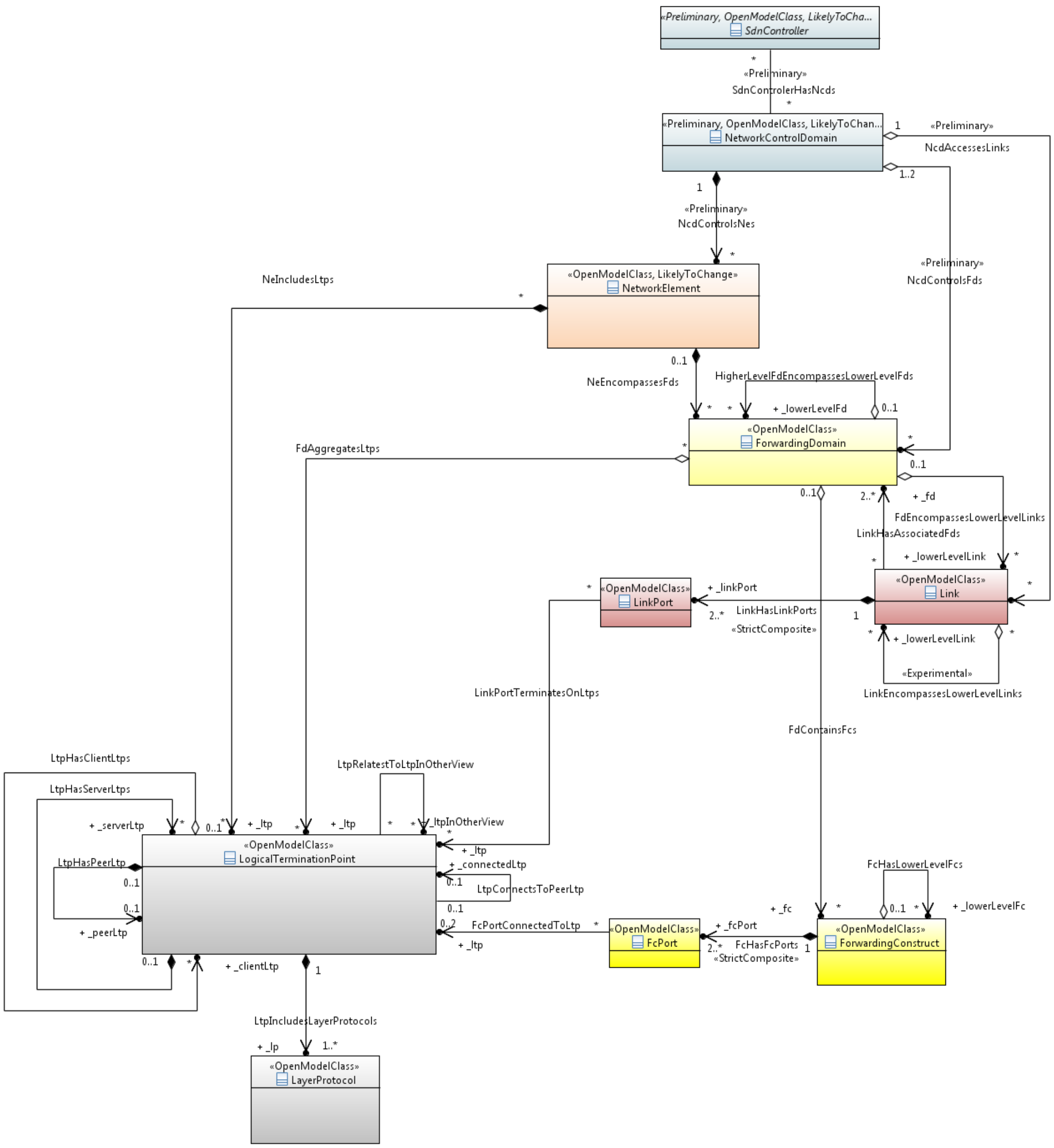
Figure 3-1 Skeleton Class Diagram of key object classes

The model in the figure above provides the basic structure for the information represented over an interface. When applying the information model to a specific interface, only a subset of the overall information model may be needed. Depending on the scope of the interface, pruning of the information model may be necessary, such as excluding a whole class or part of a class. In addition, re-factoring of the selected model artifacts may be necessary to meet the specific-purpose needs. However, re-factoring of the model artifacts should not add semantics beyond those defined in the information model. The Pruning and Refactoring method is described in [ONF TR-513].

It should be noted that the classes SdnController, NetworkControlDomain and NetworkElement[2] are being reassessed and will be remodeled in the next release.

The figure below provides more detail highlighting peer and interlayer associations between LTPs. The figures in section 4.2 Termination on page 18 explain the uses of the associations using simple pattern examples.

[2] The Network Element scope of the direct interface from a SDN controller to a Network Element in the infrastructure layer is similar to the EMS-to-NE management interface defined in the information models [ITU-T G.874.1] (OTN), [ITU-T G.8052] (Ethernet), and draft [ITU-T G.8152] (MPLS-TP).



CoreModel diagram: Forwarding-LtpInterLayerSkeletonOverview

Figure 3-2 Skeleton Class Diagram of key classes showing layering

Details of FC to link layering and other FC, Link and FD considerations are provided in[TR-512.4 ONF Core IM - Topology](#).

Note that not all attributes are shown for the classes below. Only those attributes that are relevant for this document are shown.

3.1 Termination model

3.1.1 LogicalTerminationPoint (LTP)

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 1: Attributes for LogicalTerminationPoint

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

An explanation of the structure and usage of the specification referenced by "_ltpSpec" is provided in TR-512.7 ONF Core IM - Specification. Rules for forming and interrelating LTP instances are provided in section 4.2 Termination on page 18.

3.1.2 LayerProtocol (LP)

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::LayerProtocol

The projection of an LTP into each transport layer is represented by a LayerProtocol (LP) instance. A LayerProtocol instances can be used for controlling termination and monitoring functionality. It can also be used for controlling the adaptation (i.e. encapsulation and/or multiplexing of client signal), tandem connection monitoring, traffic conditioning and/or shaping functionality at an intermediate point along a connection. Where the client – server relationship is fixed 1:1 and immutable, the layers can be encapsulated in a single LTP instance. Where there is a n:1 relationship between client and server, the layers must be split over two separate instances of LTP.

Inherits properties from:

- LocalClass

Table 2: Attributes for LayerProtocol

Attribute Name	**Lifecycle Stereotype (empty = Mature)**	**Description**
layerProtocolName		Indicate the specific layer-protocol described by the LayerProtocol entity.
_lpSpec	Experimental	The LpSpec identifies the interna structure of the LP explaining internal flexibilities, degree of termination and degree of adaptation on both client and server side.
lpDirection	Preliminary	The overall directionality of the LP. - A BIDIRECTIONAL LP will have some SINK and/or SOURCE flows. - A SINK LP can only contain elements with SINK flows or CONTRA_DIRECTION_SOURCE flows - A SOURCE LP can only contain SOURCE flows or CONTRA_DIRECTION_SINK flows
terminationState	Experimental	Indicates whether the layer is terminated and if so how.

Transport layer-protocol[3] specific properties (such as technology specific termination and adaptation properties) are not modeled directly in LayerTermination. These attributes are defined in specifications (see TR-512.7 ONF Core IM - Specification) that are used to augment the model. Where a technology specific termination has a complex structuring of internal parts, these parts will be modeled in the specification

[3] The specific transport technology Characteristic Information (see [ITU-T G.805])

3.2 Forwarding

3.2.1 ForwardingDomain (FD)

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

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

Inherits properties from:

- GlobalClass
- ForwardingEntity

Table 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 aggregation of lower level FDs into higher level FDs as views rather than FD partition, and supports multiple views. Aggregation allow reallocation of capacity from lower level FDs to different higher level FDs as if the network is reorganized (as the association is aggregation not composition).
_fc		An FD aggregares one or more FCs. A aggregated FC connects LTPs that bound the FD.
_ltp		An instance of FD is associated with zero or more LTP objects. The LTPs that bound the FD provide capacity for forwarding.
_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.

3.2.2 ForwardingConstruct (FC)

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::ForwardingConstruct

The ForwardingConstruct (FC) class models enabled constrained potential for forwarding between two or more LTPs at a particular specific layerProtocol. Like the LTP, the FC supports any transport protocol including all circuit and packet forms. It is used to effect forwarding of transport characteristic (layer protocol) information. An FC can be in only one FD. The ForwardingConstruct is a Forwarding entity. At a low level of the recursion, a FC represents a cross-connection within an NE. It may also represent a fragment of a cross-connection under certain circumstances. The FC object can be used to represent many different structures including point-to-point (P2P), point-to-multipoint (P2MP), rooted-multipoint (RMP) and multipoint-to-multipoint (MP2MP) bridge and selector structures for linear, ring or mesh protection schemes.

Inherits properties from:

- GlobalClass
- ForwardingEntity

Table 4: Attributes for ForwardingConstruct

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
layerProtocolName		The layerProtocol at which the FC enables the potential for forwarding.
_lowerLevelFc		An FC object supports a recursive aggregation relationship such that the internal construction of an FC can be exposed as multiple lower level FC objects (partitioning). Aggregation is used as for the FD to allow changes in hierarchy. FC aggregation reflects FD aggregation. The FC represents a Cross-Connection in an NE. The Cross-Connection in an NE is not necessarily the lowest level of FC partitioning.
_fcPort		The association of the FC to LTPs is made via FcPorts (essentially the ports of the FC).
_fcSpec	Preliminary	References the specification that describes the capability and internal structure of the FC (e.g. The arrangement of switches for a particular instance is described by a referenced FcSpec). The specification allows interpretation of FcPort role and switch configurations etc.
forwardingDirection		The directionality of the ForwardingConstruct. Is applicable to simple ForwardingConstructs where all FcPorts are BIDIRECTIONAL (the ForwardingConstruct will be BIDIRECTIONAL) or UNIDIRECTIONAL (the ForwardingConstruct will be UNIDIRECTIONAL). Is not present in more complex cases.

3.2.3 FcPort

Qualified Name: CoreModel::CoreNetworkModel::ObjectClasses::FcPort

The association of the FC to LTPs is made via FcPorts. The FcPort class models the access to the FC function. The traffic forwarding between the associated FcPorts of the FC depends upon the type of FC and may be associated with FcSwitch object instances. In cases where there is resilience, the FcPort may convey the resilience role of the access to the FC. It can represent a protected (resilient/reliable) point or a protecting (unreliable working or protection) point. The FcPort replaces the Protection Unit of a traditional protection model. The ForwardingConstruct can be considered as a component and the FcPort as a Port on that component.

Inherits properties from:
- LocalClass

Table 5: Attributes for FcPort

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_ltp		The FcPort may be associated with more than one LTP when the FcPort is bidirectional and the LTPs are unidirectional. Multiple Ltp - Bidirectional FcPort to two Uni Ltps Zero Ltp - BreakBeforeMake transition - Planned Ltp not yet in place - Off-network LTP referenced through other mechanism
role		Each FcPort of the FC has a role (e.g., working, protection, protected, symmetric, hub, spoke, leaf, root) in the context of the FC with respect to the FC function.
fcPortDirection		The orientation of defined flow at the FcPort.

3.2.4 Link

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

The Link class models effective adjacency between two or more ForwardingDomains (FD). In its basic form (i.e., point-to-point Link) it associates a set of LTP clients on one FD with an equivalent set of LTP clients on another FD. Like the FC, the Link has ports (LinkPort) which take roles relevant to the constraints on flows offered by the Link (e.g., Root role or leaf role for a Link that has a constrained Tree configuration).

Inherits properties from:
- GlobalClass
- ForwardingEntity

Table 6: Attributes for Link

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
layerProtocolName		The Link can support multiple transport layer protocols via the associated LTP object. For implementation optimization, where appropriate, multiple layer-specific links can be merged and represented as a single Link instance as the Link can represent a list of layer protocols. A link may support different layer protocols at each Port when it is a transitional link.

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_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 formed from subordinate links (similar FD formations from subordiate FDs). This association is intended to cover concepts such as serial compound links.
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.
_fdRuleSet		The rules related to a Link.

- At this point the model supports point to point links fully.
    - The model allows multi-point but anything above 2 (i.e., 3..*) is preliminary
- A Link may offer parameters such as capacity and delay (see TR-512.4 ONF Core IM - Topology).
    - These parameters depend on the type of technology that supports the link.

3.2.5 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 7: Attributes for LinkPort

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

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

3.3 NetworkElement, NetworkControlDomain and SdnController

- These three classes offer a rudimentary controller model that does require some advancement.
- The Network Element concept is well known in the industry and it is normal practice to represent it as in this model. However there would appear to be a number of potential issues with this traditional representation. These potential issues will be explored in a future release and there may be changes made to this entity. Because of the familiarity it has NOT been marked preliminary.
- There is work underway to improve the model in this area including development of a model for controller continuum and dismantling the NE into coherent parts (see "Future CoreModel work areas" in TR-512.1 ONF Core IM - Overview)

4 Explanatory Figures

This section provides figures that illustrate the application of the model to various network scenarios. The section covers both forwarding and termination. The forwarding views are relatively lightweight. More sophisticated forwarding views are provided in TR-512.4 ONF Core IM – Topology and TR-512.5 ONF Core IM – Resilience).

For an explanation of the symbol set being used in the figures see section 1.3 Conventions on page 5.

4.1 Forwarding

4.1.1 Basic Forwarding

The basic forwarding model, described in previous sections, offers the capability to enable constrained forwarding between LTPs. The figure below provides a basic nodal view.

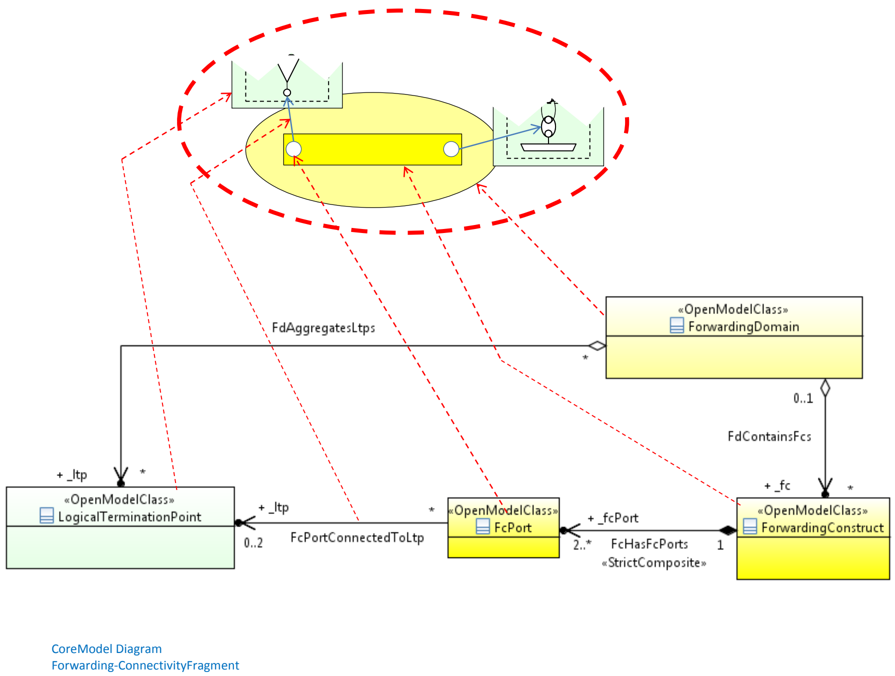


Figure 4-1 Forwarding fragment in a nodal view

The pictorial form in the figure above shows the ForwardingConstruct (FC) in the context of two LTPs. The FC defines the enabled constrained forwarding between the LTPs (in the figure it is point to point). The FcPort of the FC is shown within the FC, emphasizing the strict whole-part relationship and lifecycle dependency of the FcPort on the FC. The FcPorts are effectively FC component ports. The FC shown has two FcPorts but the model allows for two or more FcPorts [2..*] where in some cases the FcPort could be selected as a source or destination for switching. The protection switching capability is explained elsewhere in this document.

The [0..2] multiplicity of _ltpRefList (at the end of the association "FcPortConnectedToLtp" allows for a bidirectional FcPort to associate with two unidirectional LTPs.

4.1.2 Forwarding the topology

The FC defining the enable constrained forwarding between a set of LTPs can be considered in the context of a network topology offering capacity.

The figure below shows a network for a single layer protocol in terms of the basic topology of FDs, Links and LTPs (grey) that provide capability and capacity for the layer protocol and the signal forwarding using FCs (X, Y and Z) and LTPs (green) enabling information flow for the layer protocol.

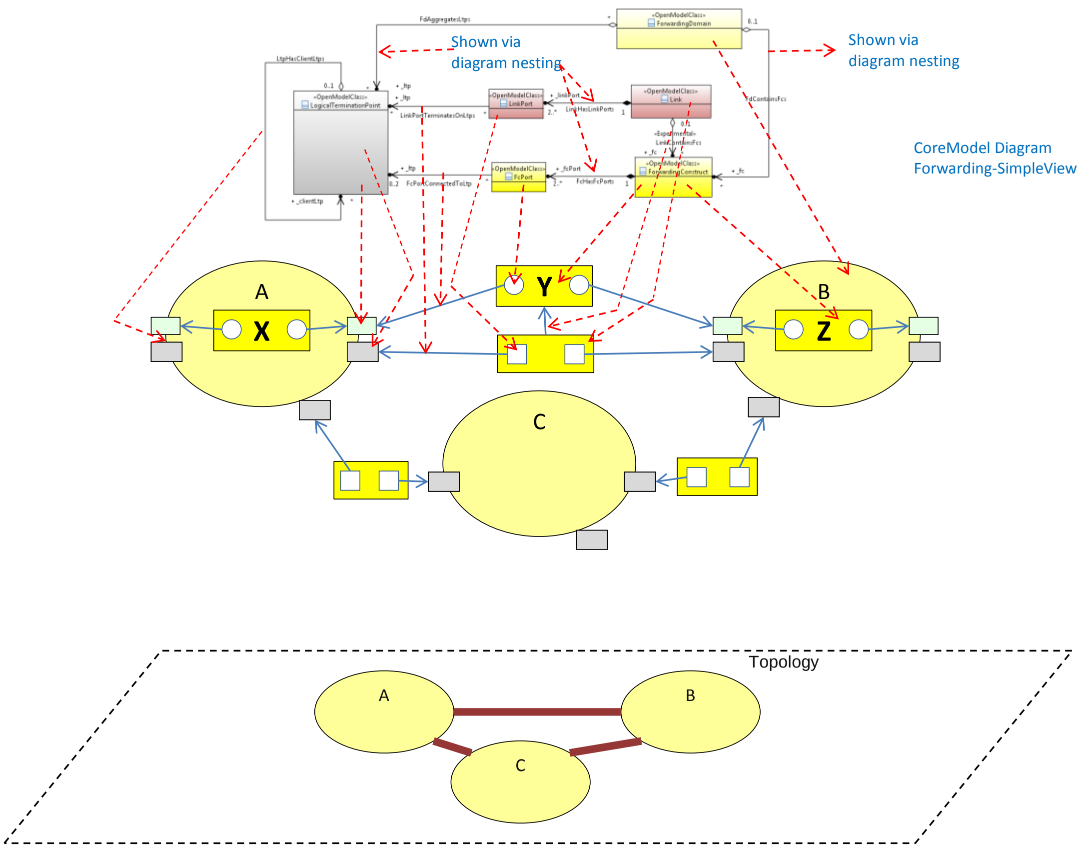


Figure 4-2 Forwarding in a single layer

The following section deals with LTP layering considered in the context of singe FDs. More sophisticated multi-layer multi-FD and multi-view considerations are covered in detail in TR-512.4 ONF Core IM - Topology.

4.2 Termination

In some of the figures the LP is depicted with a view of the internal details. The following figure shows the cases illustrated in figures. In a realization the LP detail structure would be expressed by a specification as described in TR-512.7 ONF Core IM - Specification.

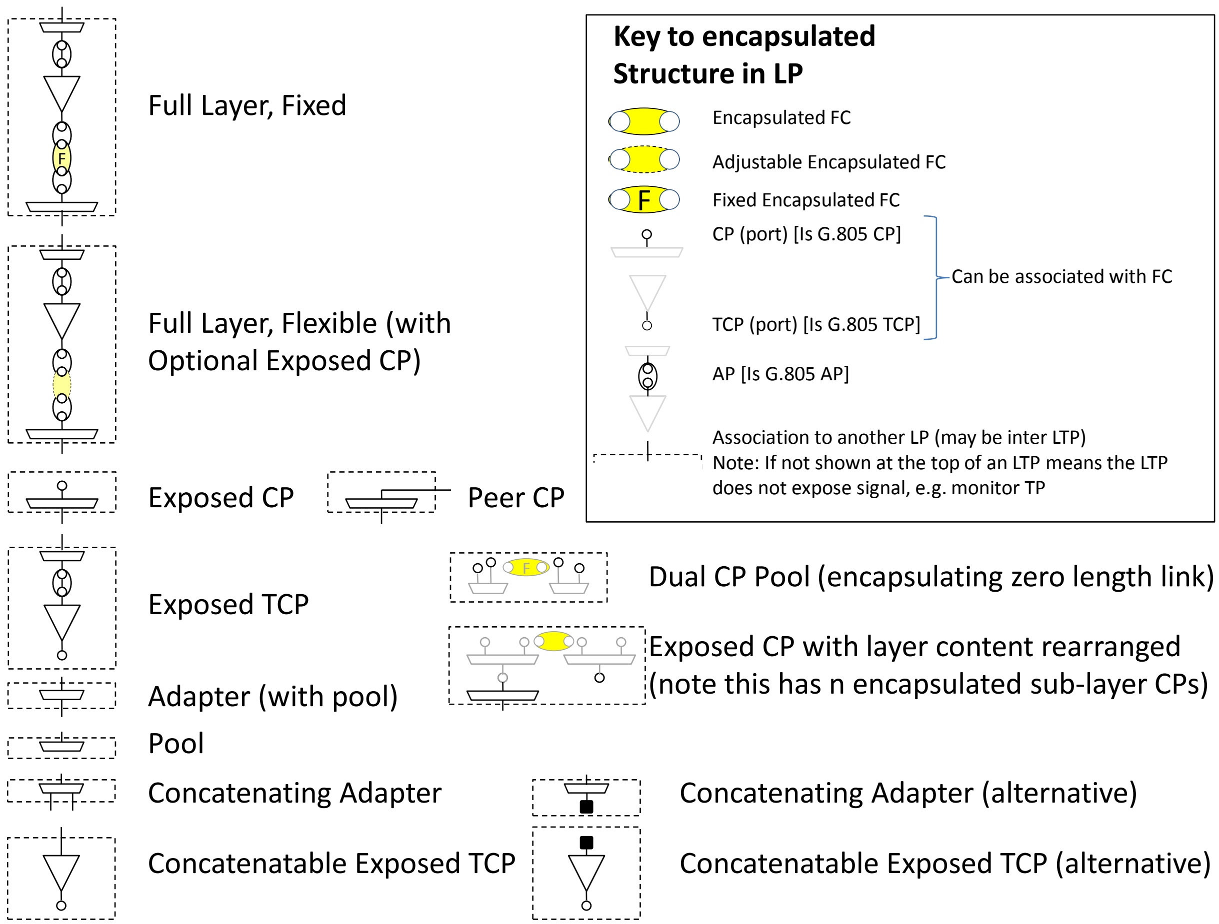


Figure 4-3 LP Cases

The relationship between some of the entities in the ONF-CIM and other familiar models are shown in the next figure. The figure also provides a key to some additional symbols. Further mappings are provided in TR-512.9 ONF Core IM - Terminology Mapping.

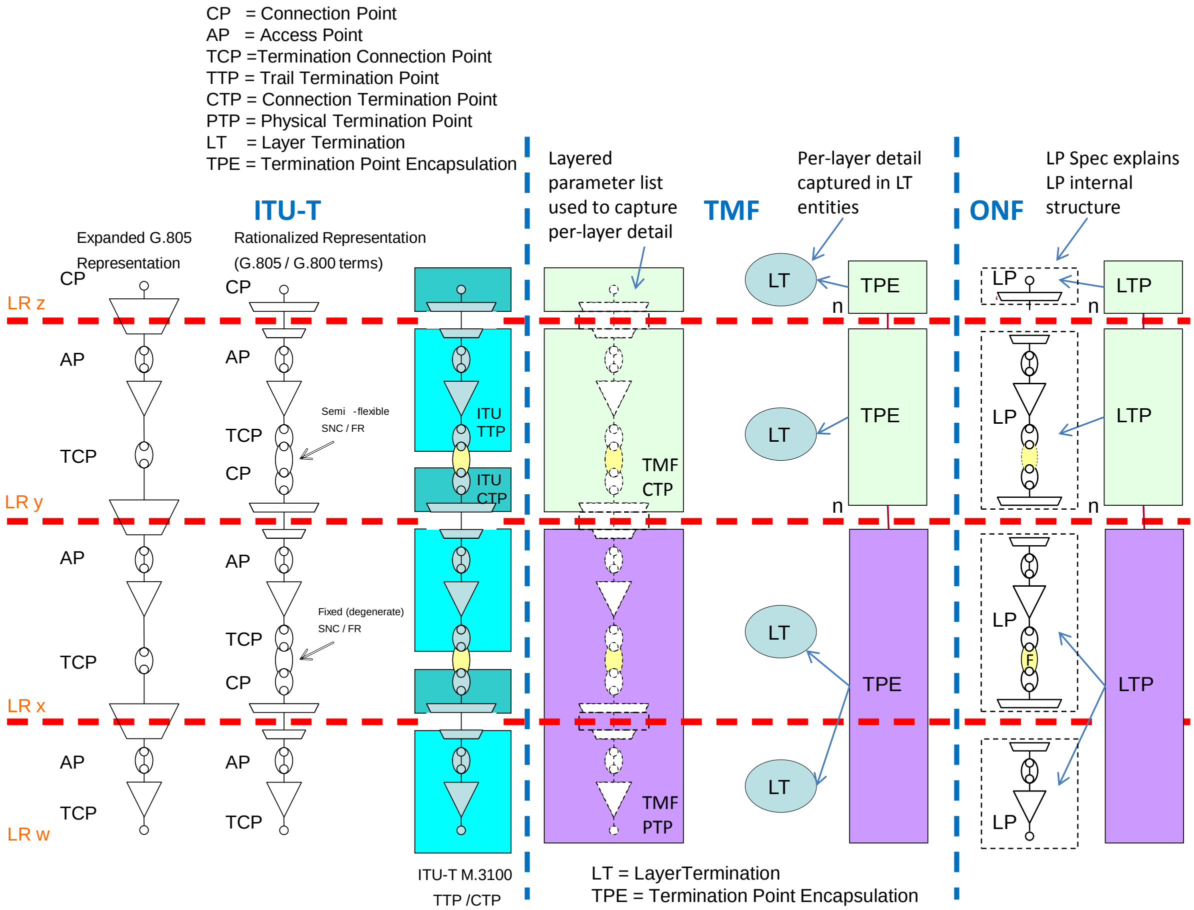


Figure 4-4 Mapping from ITU-T and TM Forum Termination models to the ONF Core[4]

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

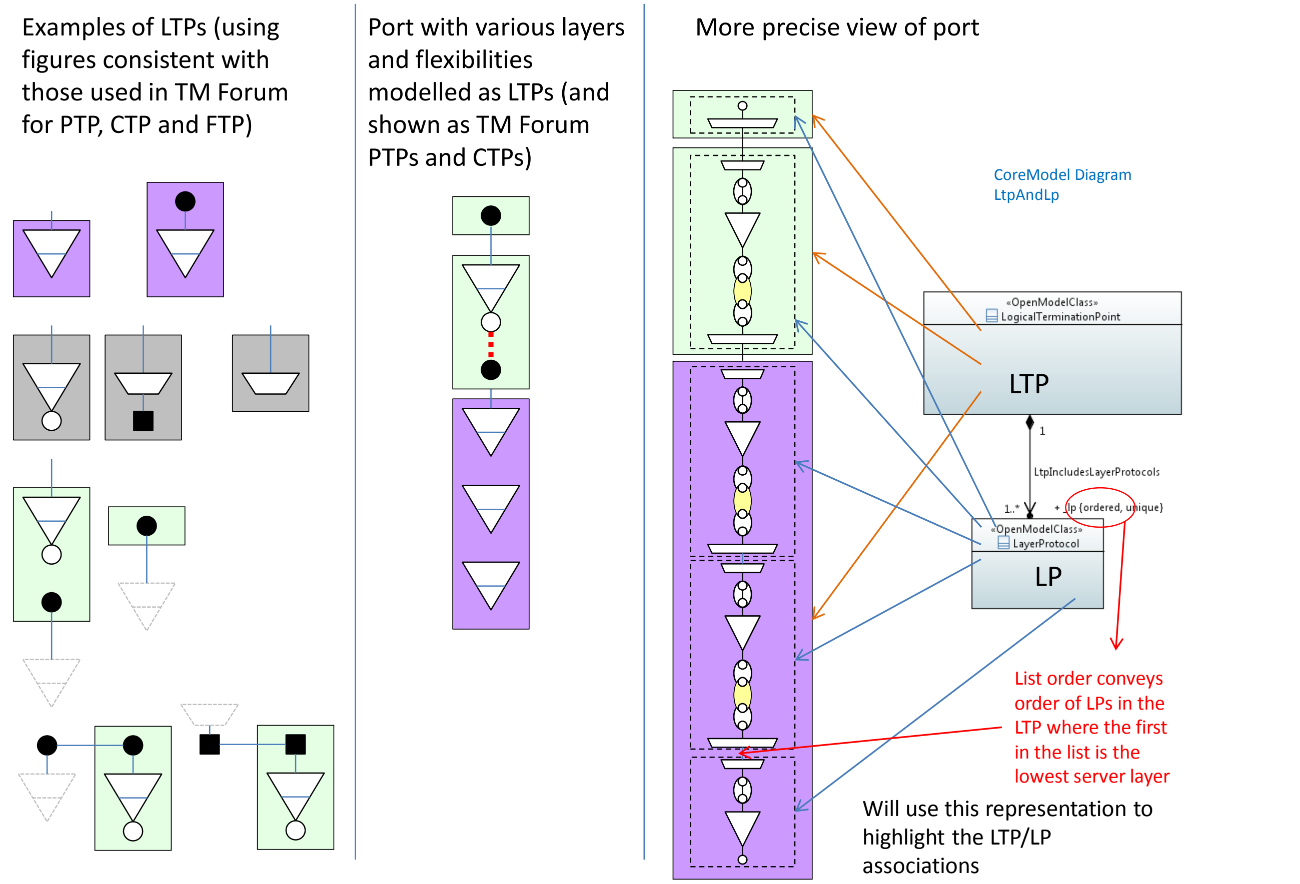


Figure 4-5 Representations of LTPs

In the figure above, the pictorial form shows a number of representations of LTPs (purple, grey and green) representing the layering associated with physical ports (purple), their connectable clients (green) and floating LTPs (grey). The right most pictorial form shows the relationship between the LTP and the LP in terms of a detailed symbol derived from work by TM Forum and ITU-T.[5] An LP instance represents all aspects of termination of a single layer-protocol. An LTP is composed of 1 or more LPs, where the LPs represent the stack of terminations relevant to the LTP as depicted in the pictorial view. A termination stack may spread across several LTPs. The reason for this split includes multiplicity, connection flexibility and flow orientation transitions (see also 1.3 Conventions on page 5 for reference to the diagram keys etc).

In the model:

- The flow of signal through the aspects of the LP shown in the figure is not currently formally represented,
    - The LTP specification work (see [TR-512.7 ONF Core IM - Specification]) which is currently experimental provides the basis for formal representation in a following release.

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

- The flow between LPs within an LTP is represented via list order (see the note on the figure above)
- The flow between LPs in different LTPs in a hierarchy is represented by the specific LTP relationship (see Figure 4-7 LTP relationships illustrated in a simple Network Element context on page 23) and the corresponding LP list order in the LTP
  - In the figure above the Sink[6] signal flowing from the top of the upper LP of the purple LTP (i.e. the last entry in the LP list of that LTP) passes to the bottom of the LP in the associated green LTP

There are a number of different cases of LTP which are depicted in the figure below.

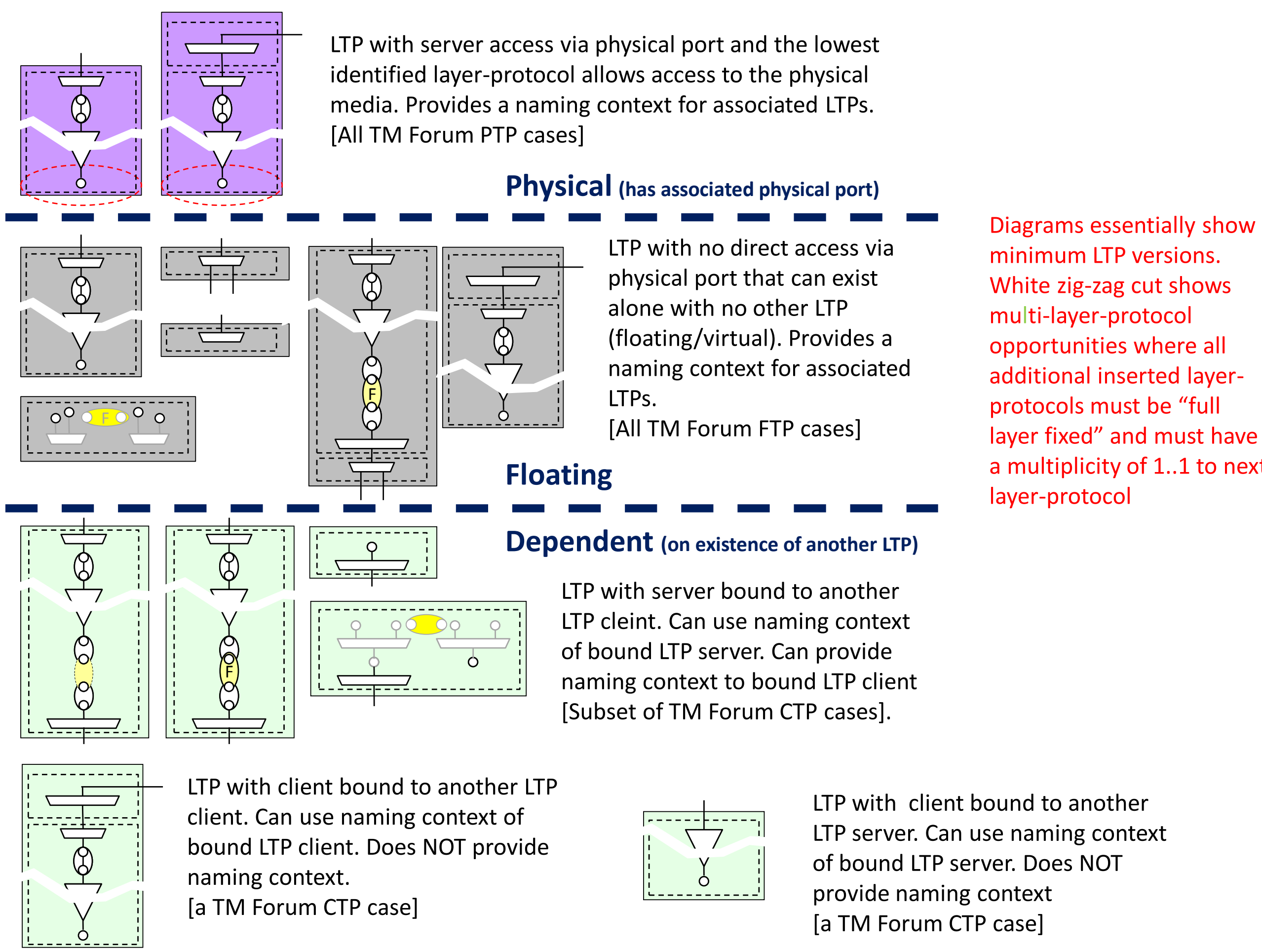


Figure 4-6 LTP Cases

[6] See section 4.3 Directionality on page 56.

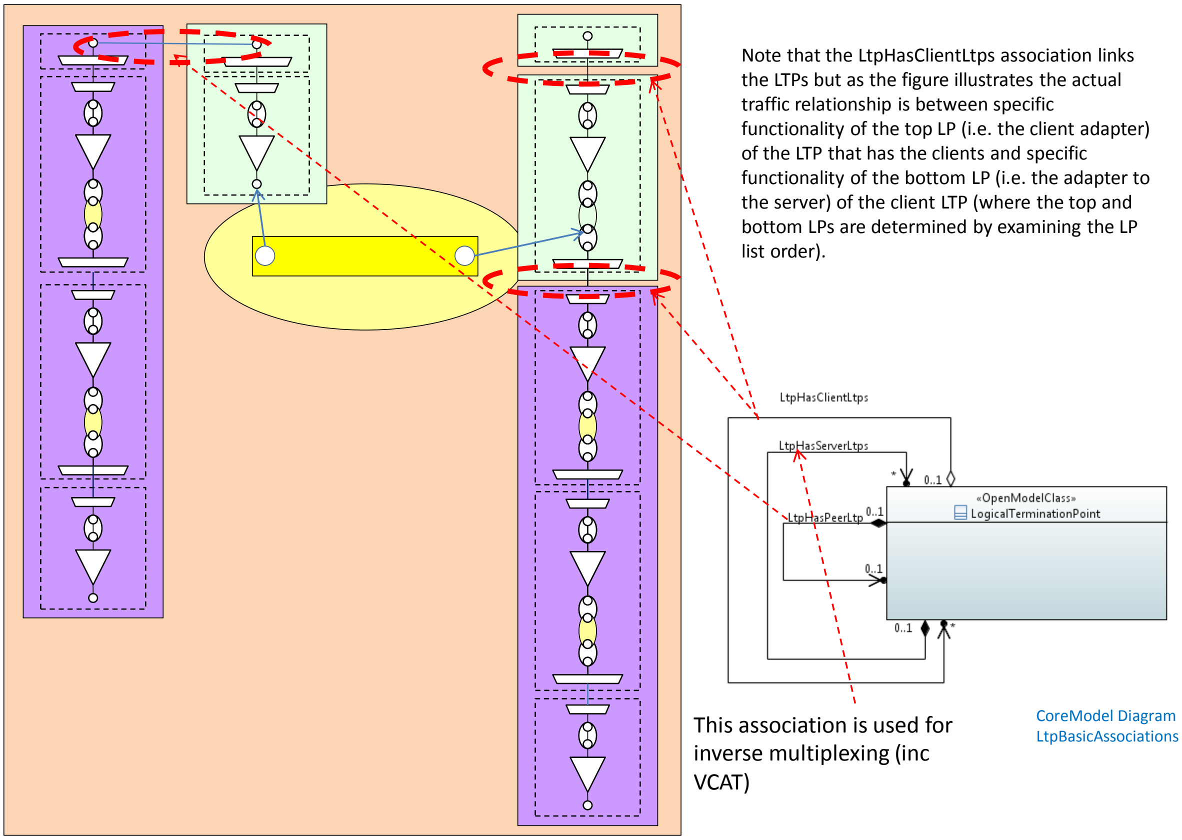


Figure 4-7 LTP relationships illustrated in a simple Network Element context

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

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

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

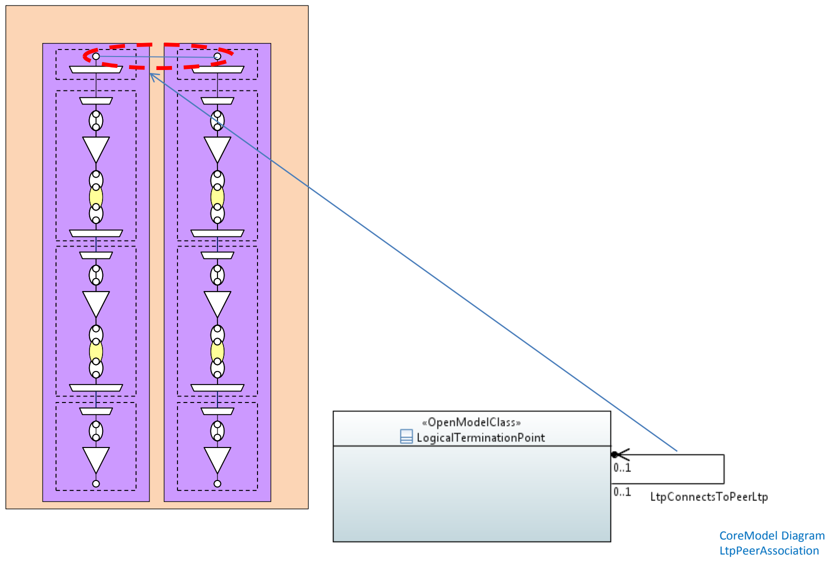


Figure 4-8 LtpConnectsToPeerLtp illustrated in an Amplifier/Regenerator context

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

The following figure shows a standard case of an FC between two LTPs (green) which are clients of LTPs (purple) where those LTPs support multiple clients.

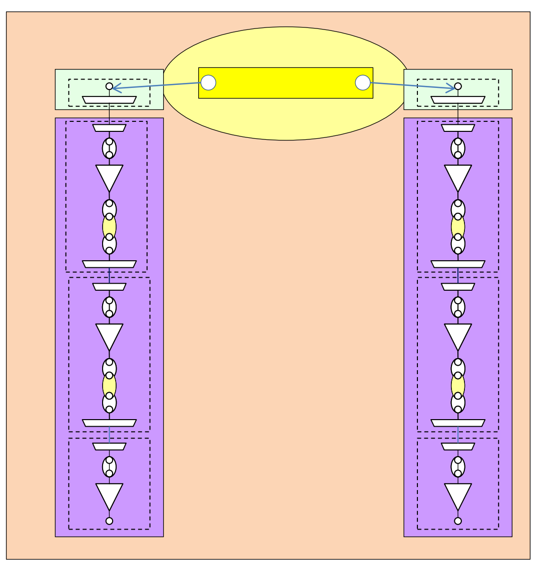


Figure 4-9 FC between LTPs

The following figure shows a standard case of an FC between two LTPs (purple) where there is forwarding flexibility but the LTP supports only one signal flow.

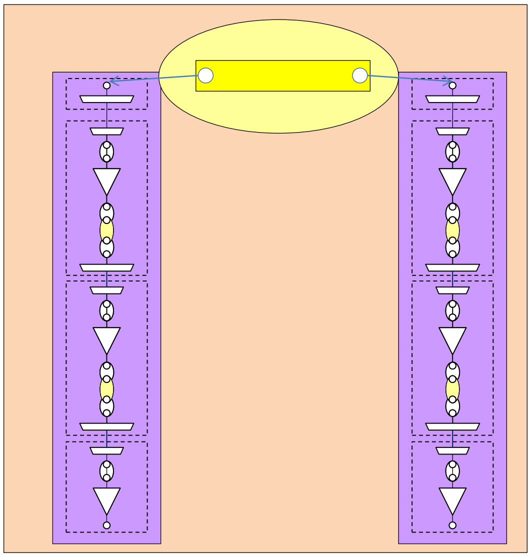
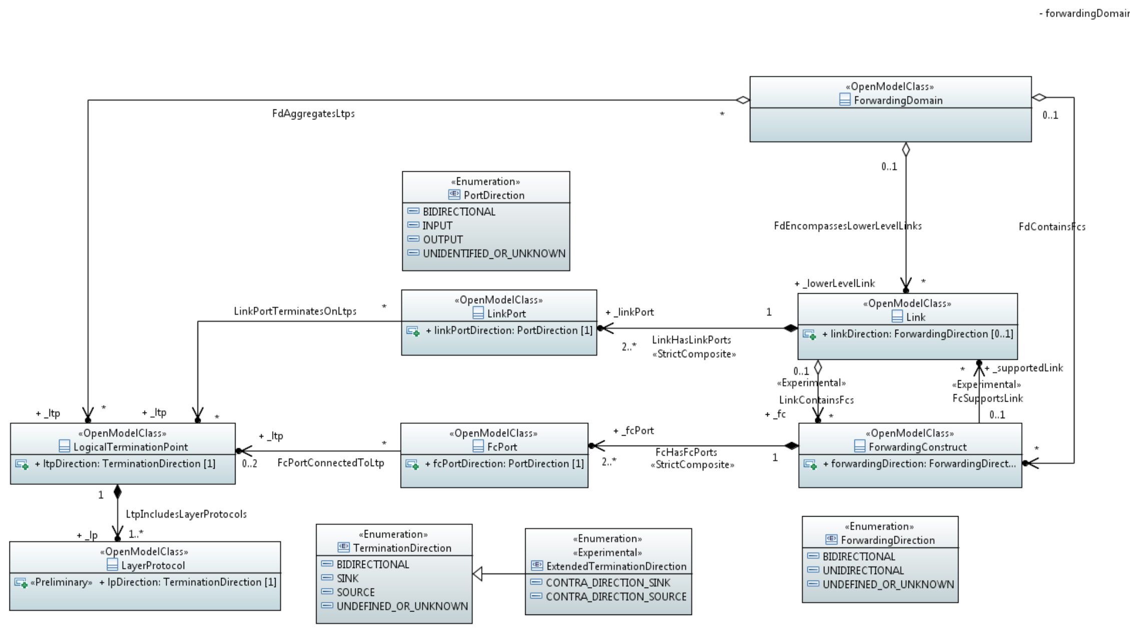


Figure 4-10 FC between LTPs supporting only one flow

4.3 Directionality

The model supports bidirectional, unidirectional and mixed directionality constructs. The following figure shows the directionality attributes and data types.



CoreModel diagram: ForwardingConnectivityFragmentWithLtAndDirection

Figure 4-11 Model highlighting directionality

The following figure shows in pictorial form the meaning of the key direction attributes in the model.

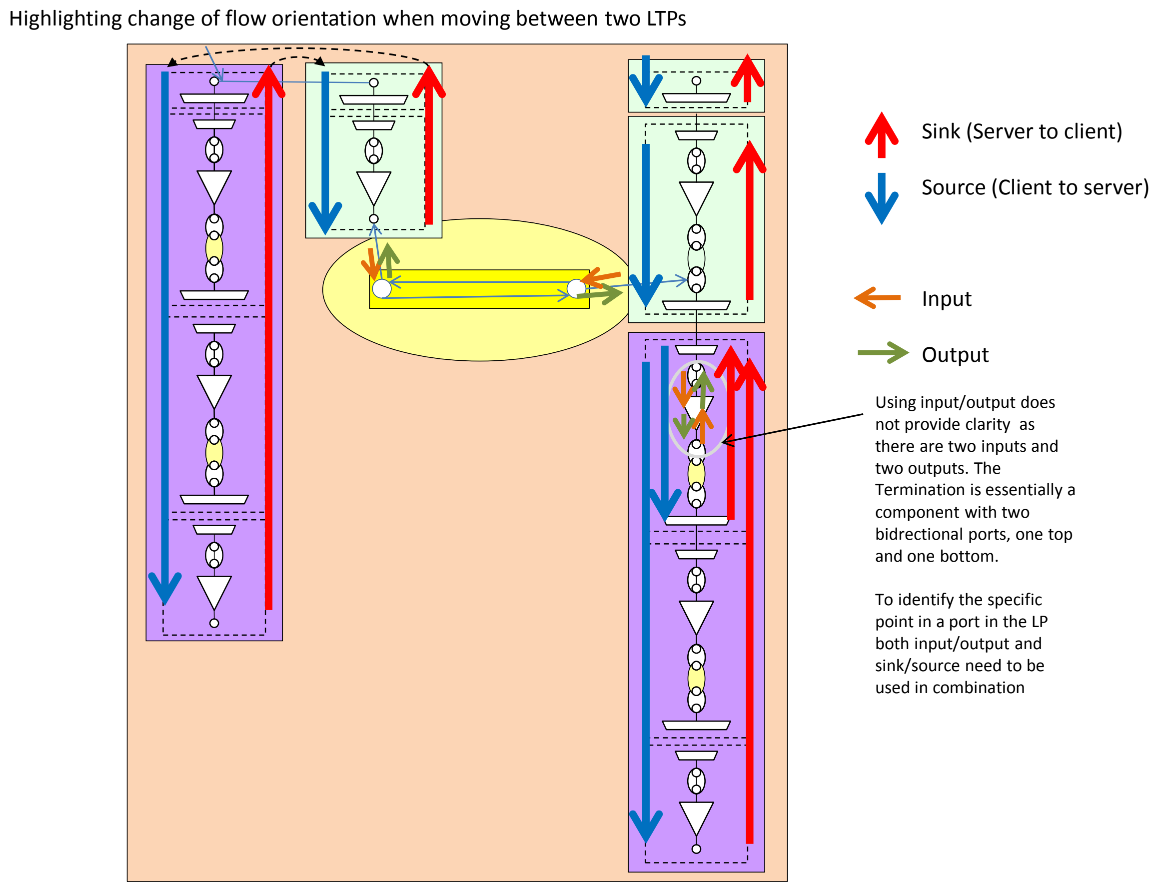


Figure 4-12 Interpreting the direction attributes

The figure above shows bidirectional LTPs and an FC in an NE context. It should be noted that the terms Sink and Source are consistent with Input and Output at the base of the LTP/LP (but counterintuitive at the top of the LTP/LP (where a Sink outputs signal). The specific terminology is aligned to that used in ITU-T. Sink/Source are defined in terms of "flow orientation" in the layer stack (i.e. client to server or server to client).

There are a number of legal combinations of bidirectional and unidirectional LTPs and FCs. The following sequence of figures provides an overview.

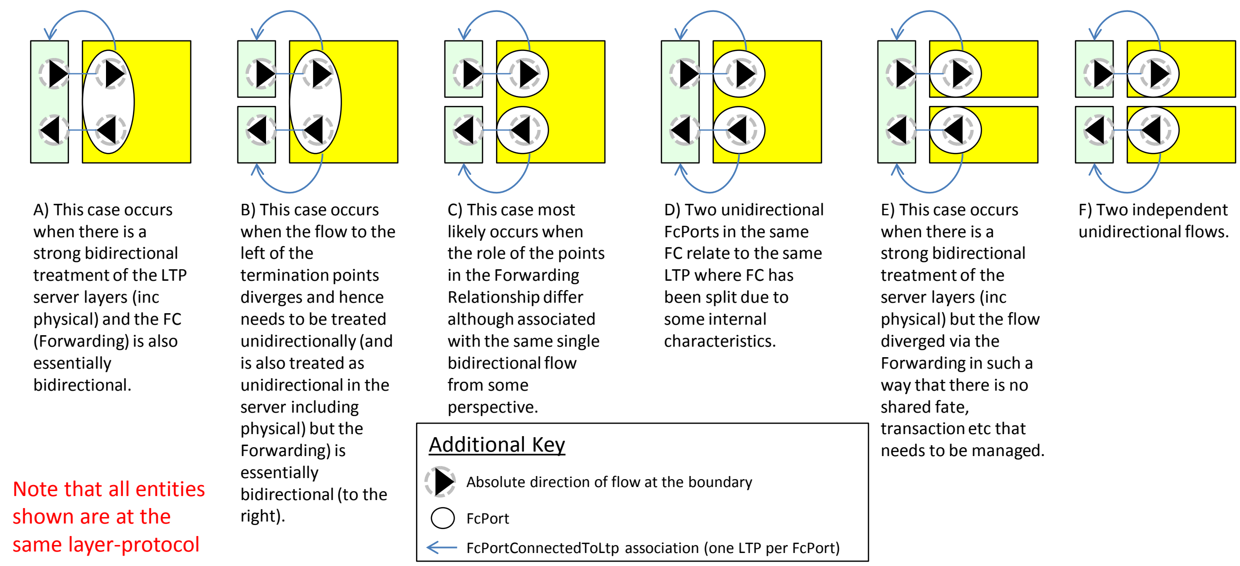


Figure 4-13 Various mixed directionality forms

The following figure shows how to relate two unidirectional LTPs to a single FC where the two LTPs are intended to carry the same traffic. The pattern also applies to bidirectional LTPs and FCs.

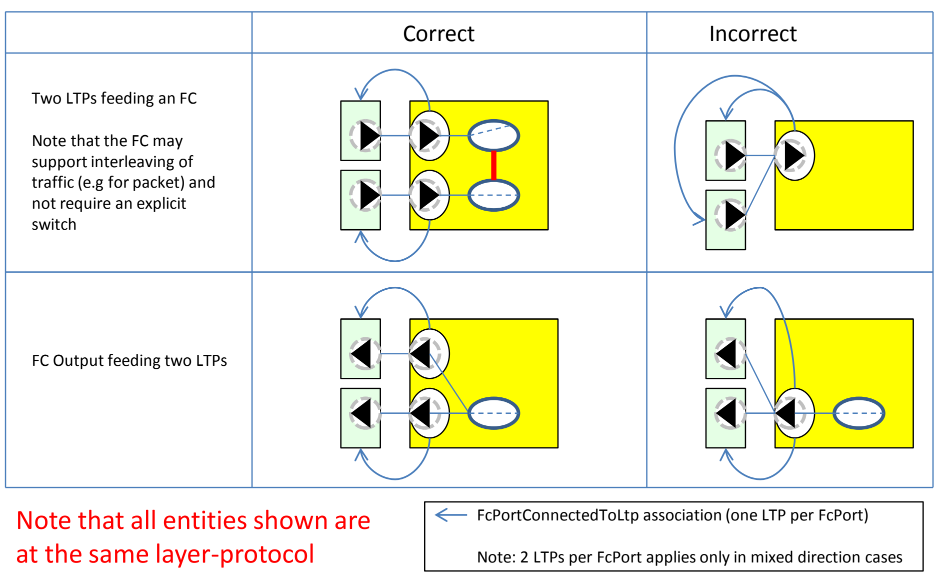
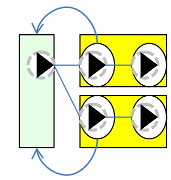
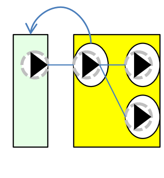
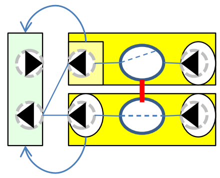
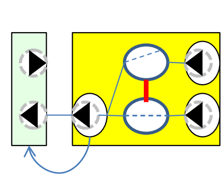


Figure 4-14 Interrelationship between a pair of unidirectional LTPs and a unidirectional FC

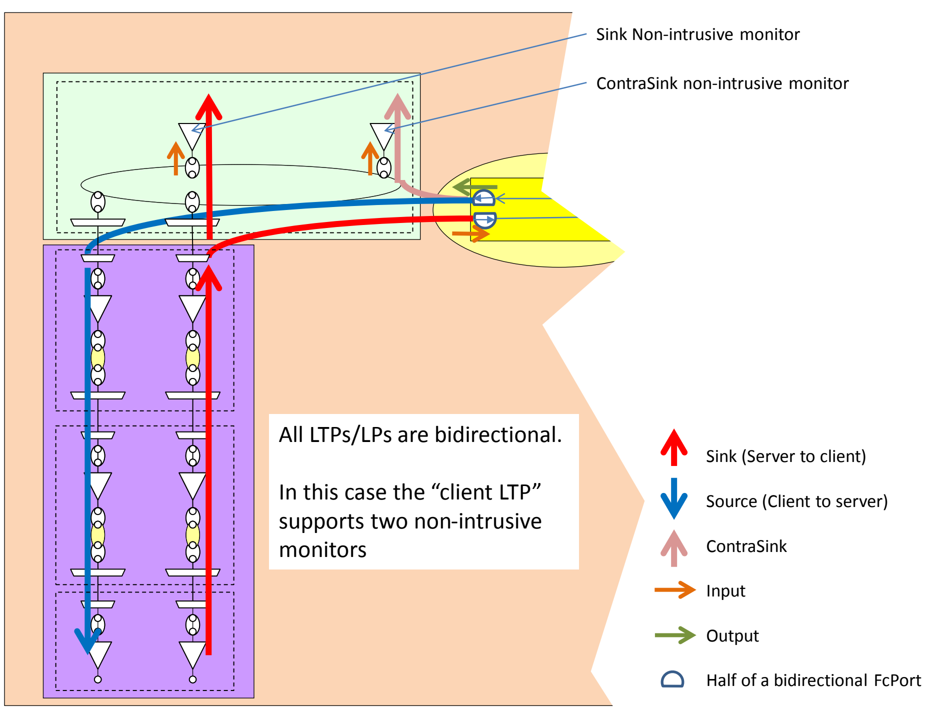
The following figure shows how to relate two unidirectional FCs to a single LTP where the two FCs are intended to carry the same traffic. The pattern also applies to bidirectional LTPs and FCs.

Shows access to FC	Overlay form	Encapsulated form
One LTP feeding two FC that have different remote destinations (or the same remote via different routes)		
Two FCs feeding one LTP where the FCs have the same remote sources (only legal if flows switched or interleaved)		

Note that all entities shown are at the same layer-protocol

Figure 4-15 Interrelationship between a pair of unidirectional FCs and a single LTP

In some network cases the LP encapsulates several terminations functions with the same essential orientation of flow. The figure below shows a case with non-intrusive monitoring in an LTP (green)[7]. In that LTP, the two cases of Sink flow are distinguished by recognizing that one is in the normal orientation (red flow) with respect to standard traffic flow, i.e. the signal passed from the Server LTP is further terminated, whereas the other is in a non-normal orientation, i.e. the signal that would be expected to be encoded by (multiplexed etc.) by the server LTP is actually terminated (blue going to brown flow). The non-normal orientation is called ContraSink.



[7] The measures for the non-intrusive monitor are no different from the measures for the corresponding Termination. The Termination is embedded in the LP… hence so is the non-intrusive monitoring.

Figure 4-16 Contra-directionality showing monitors

The same logic applies to the Source terminations as depicted in the following figure where the LTP has both non-intrusive monitoring (as in the previous figure) and the potential for active test signal injection in an LTP (green)

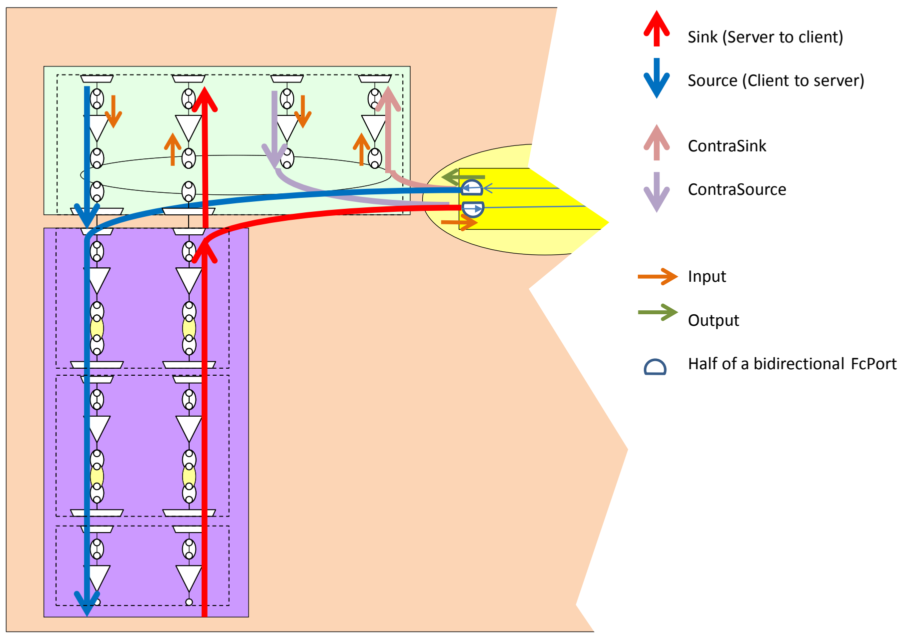


Figure 4-17 Contra-directionality showing monitors and signal sources

The Client LTP has one LP (which is considered simply as Bidirectional) which has four termination functions (where two are contra-directional). As a consequence there are four inputs to the termination functions, these are distinguished as follows:

- Source Input
- ContraSource Input
- Sink Input
- ContraSink Input

It is expected that the LP directly include the Source and Sink attributes and a composed part of the LP would include the ContraSource and ContraSink attributes (this is for further study[8]).

In the following example where there is a deep inspection capability dealing with two layers of inspection. It is assumed that the forwarding technology is such that the server layer supports only one client. Although the LTPs are bidirectional, the upper LP of the green LTP is unidirectional Sink. This illustrates one case where an LTP directionality is different from the directionality of an included LP.

[8] The measures etc for the SINK and CONTRA_DIRECTION_SINK are likely to be the same hence the need to partition the CONTRA_DIRECTION_SINK measure etc into a composed part (to avoid name clashes)

The Client LTP (which is considered simply as Bidirectional) has two termination functions in for the layer-protocol of the FC (where one is contra-directional). As a consequence there are two inputs to termination functions. These are distinguished as follows:

- Sink Input
- ContraSink Input

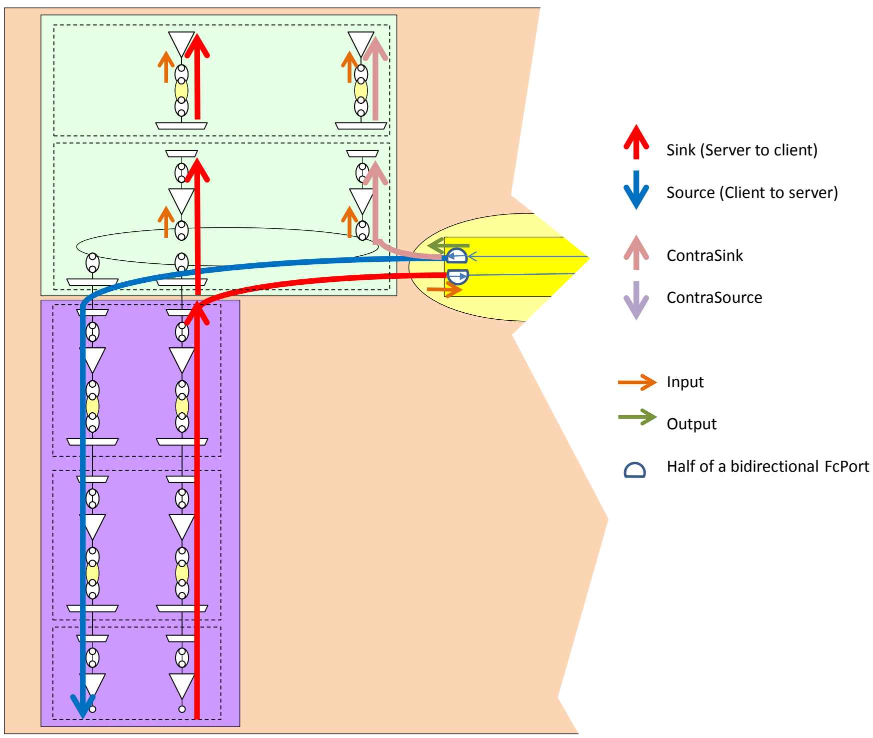
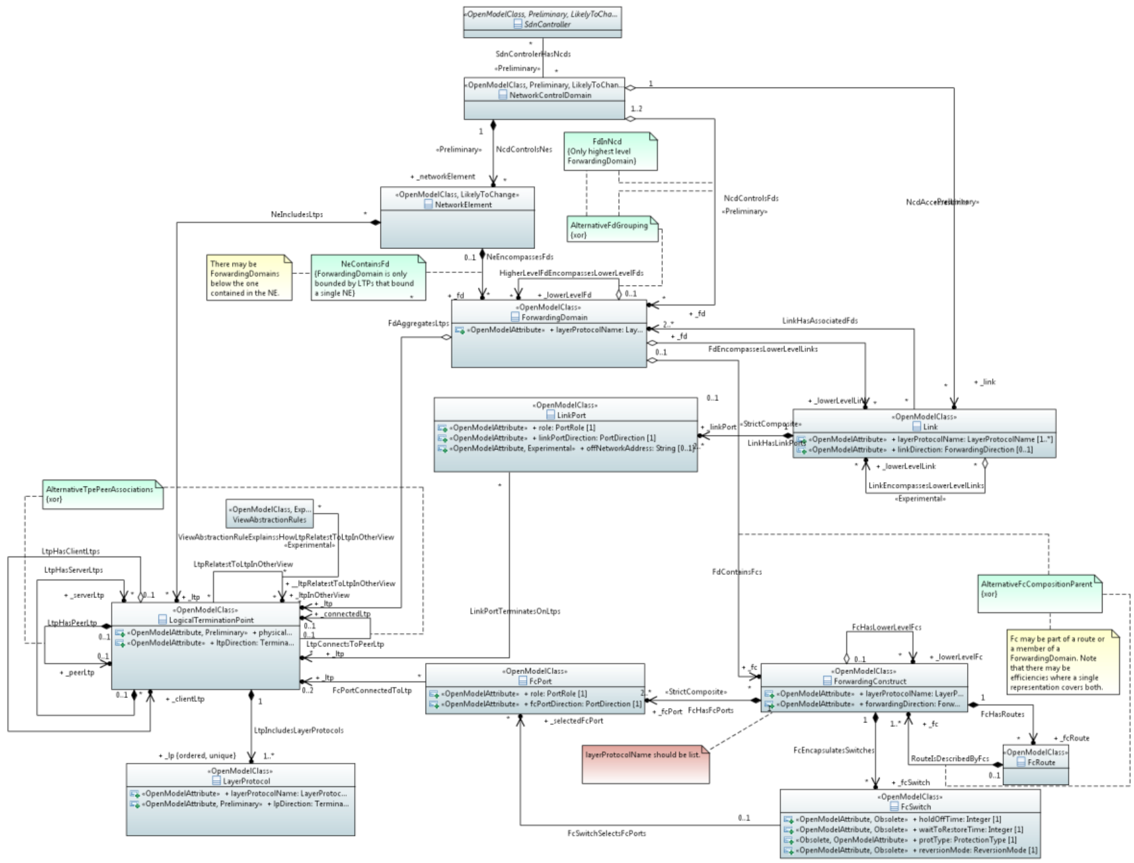


Figure 4-18 Contra-directionality showing deep inspection

5 Work in progress

The figure below shows some constraints on the associations in the model. Further work is being carried out on how to most appropriately represent constraints. The figure also shows some classes related to other parts of the model covered in other documents (see TR-512.4 ONF Core IM – Topology and TR-512.5 ONF Core IM – Resilience).



CoreModel diagram: HighLevelDetail

Figure 5-1 Class Diagram of all key classes showing attributes and constraints

The above diagram shows owned attributes of the key classes in the model. Not all classes are shown and the classes in the diagram have additional attributes related to associations to those classes as well as some inherited attributes and some experimental attributes.

End of Document

