Core Information Model (CoreModel)

TR-512.8

Control

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

List of Figures

Document History

Version	Date	Description of Change
		This document was not published prior to Version 1.3
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 as well as 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 Control model

As explained in TR-512 V1.2 the classes SdnController, NetworkControlDomain and NetworkElement[1] have been reassessed and deprecated and new classes have been developed in

[1] 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).

this release to replace them. It has been recognized that a uniform recursive model of control can be developed that provides a consistent treatment of what were previously seen as completely different things.

This document describes a general model of control suitable for representation of the capabilities that control the network and for representation of the relationship to the model of the network from the control perspective. The document also discusses the dismantling of the NE and recasting aspects of it as Control.

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

3 Model of control component and views

3.1 Background

The ONF Architecture [reference] talks of a recursion of control aligning well with the more general concept of the Management-Control Continuum from [TMF IG1118]. The control model in [ONF TR-512 V1.2] showed a traditional hierarchy rather than a generalized recursion.

Over many years it has become apparent that the traditional representation of Network Element (NE) and of Managed Element (ME) was not correct. It is clear that from one perspective the Network Element is simply a lower level member of the Management-Control Continuum. It is also apparent that all other aspects of the NE are covered by other parts of the model.

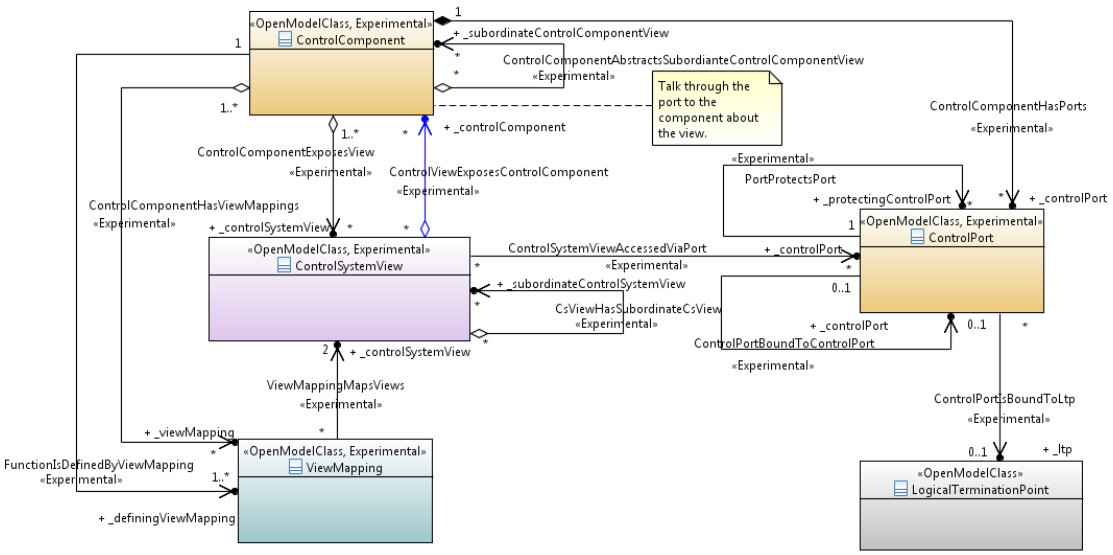
It was concluded that the NE should be remodeled as simply a control capability and that that capability should be generalized so that it could handle all aspects of the Management Control Continuum.

The model chosen for the Control functions is derived from the Component-System pattern (see TR-512.A.2) and the ProcessingConstruct (see TR-512.11 and TR-512.A.9). It was then clear that as a controller controls components then the components of the controller that deal with controlling other things also needed to be controlled (as is explained in the Management Control Continuum).

The following sections set out the model in this context.

3.2 The control model in the context of the core classes

The figure below shows the core of the Control model.



CoreModel diagram: Control-ControlComponentAndControlViewCore

Figure 3-1 Core Control Model

The classes are described in the section below. Some aspects of the model described below are shown in figures in sections 3.3, 3.4 and 3.5.

3.2.1 ControlComponent

Qualified Name: CoreModel::GeneralControllerModel::ObjectClasses::ControlComponent

Represents control capability/functionality. The functionality is emergent from the running software.
ControlComponents communicate with other ControlComponents through ControlPorts about things expressed in the corresponding ControlSystemViews.
Examples of higher level aggregate ControlComponents are:
- the controller in the Network/Managed Element e.g. an SNMP agent).
- an SDN Controller.
- an EMS.
- an NMS.

Examples of lower level ControlComponents are:
- Path Computation.
- Problem Analytics.

This specific model follows a subset of the Component-System Pattern as that is all that is necessary for this aspect of expression.

This class is Experimental.

Table 1: Attributes for ControlComponent

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_controlSystemView	Experimental	See referenced class

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_definingViewMapping	Experimental	ControlComponent behavior is defined in part by view mappings.
_controlPort	Experimental	See referenced class
_subordinateControlComponentView	Experimental	A ControlComponent that is part of an abstract view of the system that supports the referencing ControlComponent and hence describes part of the behavior of the referencing ControlComponent.
_viewMapping		ControlComponent uses the referenced ViewMapping to produce one view from another.

3.2.2 ControlSystemView

Qualified Name: CoreModel::GeneralControllerModel::ObjectClasses::ControlSystemView

A view of the things controlled by a control system.
A ControlSystemView is structured presentation of the underlying controlled things (the "actual" entities) for some purpose.
The ControlSystemView is constructed by pruning and refactoring the models of the underlying controlled things.
Is an aggregation of "views of things" where a "view of things" is represented in abstraction by an Object Class in the model.
The ControlComponent is itself controlled and presents itself in terms of ControlComponents (subordinate) in a view.
At one extreme a ControlSystemView may expose all underlying details of everything controlled with no adjustment from the presentation provided by the controlled things.
A ControlSystemView may expose a subset of the controlled things that focuses on a particular aspect (e.g. only the ControlComponents).

This class is Experimental.

Table 2: Attributes for ControlSystemView

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_ltp	Experimental	LTP in ControlSystemView
_fd	Experimental	FD in ControlSystemView
_link	Experimental	Link in ControlSystemView
_fc	Experimental	FC in ControlSystemView
_pc	Experimental	PC in ControlSystemView

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_cd	Experimental	CD in ControlSystemView
_equipment	Experimental	Equipment in ControlSystemView
_subordinateControlSystemView	Experimental	A ControlSystemView that is a subset of this ControlSystemView providing more detail.
_controlPort	Experimental	Port of ControlComponent through which the referencing ControlSystemView is accessed.
_controlComponent	Experimental	A ControlComponent (subordinate) that is in this ControlSystemView (superior) which was provided by another ControlComponent (superior). The ControlComponent (subordinate) will have exposed its own ControlSystemViews (original subordinate). These ControlSystemViews (original subordinate) will be pruned and refactored by the ControlComponent (superior) to be presented in the ControlSystemView (superior) under the view of the ControlComponent (subordinate) as a ControlSystemView (pruned and refactored subordinate). The ControlSystemView (pruned and refactored subordinate) will have other components represented that may include ControlComponents. The process is recursive.

3.2.3 ControlPort

Qualified Name: CoreModel::GeneralControllerModel::ObjectClasses::ControlPort

The access to the ControlComponent following the normal Component-Port pattern (i.e. the functions of a component are accessed via ports).
Is assumed to usually be bidirectional.

This class is Experimental.

Table 3: Attributes for ControlPort

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_protectingControlPort	Experimental	A simple representation of resilience where one ControlPorts are identified as providing equivalent information.
_controlPort	Experimental	Control Ports may be used to associate controllers in a hierarch and as peers. Peer controllers are assumed to both the subordinate of each other.
_ltp	Experimental	The LTP through which the control messaging/signaling flows.

3.2.4 ViewMapping

Qualified Name: CoreModel::GeneralControllerModel::ObjectClasses::ViewMapping

The rules that relate one view to another.

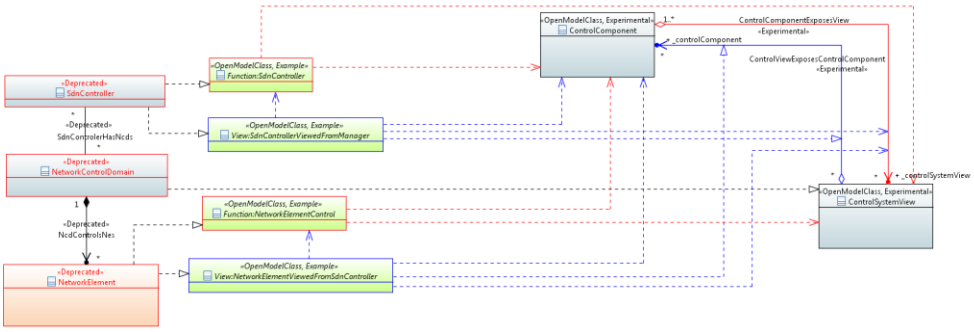
This class is Experimental.

Table 4: Attributes for ViewMapping

Attribute Name	Lifecycle Stereotype (empty = Mature)	Description
_controllerInternalView	Experimental	See referenced class
_controlSystemView	Experimental	See referenced class

3.3 Relationship to TR-512 V1.2 model

The relationship between the V1.2 classes (that have been deprecated) and the new V1.3 classes is depicted in the figure below.



CoreModel diagram: Control-MappingToControlComponentAndControlView

Figure 3-2 Mapping Core Control Model to traditional view

The V1.2 classes are shown with (red text and a red border). These are related to the V1.3 classes (shown with black text and a black border) via some explanatory classes (shown with a green fill). The relationships are purely pictorial.

The explanatory classes show (via the black dashed associations) that:

- The SdnController class (of V1.2) represents both the SDN Controller function and a view of that function as seen through an interface provided by a manager of the SDN Controller
- The NetworkControlDomain (of V1.2) represents the view of the network controlled by the SDN Controller as presented by the SDN Controller
- The NetworkElement (of V1.2) represents the embedded Network Element Control function presented to the SDN Controller as well as a view of that function as seen through an interface provided by the SDN Controller controlling the NE

The dashed associations, red for Functions and blue for views, highlight (roughly) that in the V1.3 model:

- The NetworkElementControl function is represented by a ControlComponent and corresponding ControlSystemView which will have:
    - LTPs, FCs and other abstract representations of NE functions
    - Any relevant ControlComponents that make up the control functions of the NE, such as log managers and alarm queue functions, of the NE to be exposed[2]
- The SdnController function is represented by a ControlComponent and corresponding ControlSystemView. The ControlSystemView will have:
    - ControlComponents representing the Network Elements controlled by the SDN Controller (see NetworkElementViewedFromSdnController below)
    - LTPs, FCs and other abstract representations of network functions abstracted from the assembly of NE level functions
    - Any relevant ControlComponents that make up the control functions of the SDN Controller, such as log managers etc.
- The NetworkElementViewedFromSdnController view will include:
    - A ControlComponent representing the NE as relevant to the specific view provided by the SDN Controller
    - A ControlSystemView which will have:
        - LTPs, FCs and other representations of NE functions
        - Any relevant ControlComponents that make up the control functions of the NE, such as log managers and alarm queue functions, of the NE to be exposed

      Where the instances in the view are all abstractions (pruned and refactored forms) of those provided by the actual NE
- The SdnControllerViewedFromManager view will include:
    - A ControlComponent representing the SDN Controller as relevant to the specific view provided by the Manager (seen through an interface provided by the manager managing the SDN Controller)
    - A ControlSystemView which will have:
        - LTPs, FCs and other abstract representations of network functions (see SdnController above)
        - Any relevant ControlComponents that make up the control functions of the SDN Controller (see SdnController) above
        - ControlComponents representing the Network Elements controlled by the SDN Controller (see NetworkElementViewedFromSdnController below)

Where the instances in the view are all abstractions (pruned and refactored forms) of those provided by the actual SDN Controller

Clearly the above is recursive and hence a Manager could present the following via the same mechanism:

- A ControlComponent representing the manager itself

[2] The model does not provide explicit representations for such ControlComponents. The generalized ControlComponent class should be used decorated appropriately.

- A ControlComponent representing each subordinate manager
- A ControlComponent representing each SDN Controller subordinate to each subordinate manager
- A ControlComponent representing each NE controlled by each SDN Controller…

A complex NE could represent subordinate parts again through the same mechanism leading to a deep Component-View hierarchy.

The classes listed here are provided in the model to assist in the understanding of the mapping from ManagedElement, SdnController and NetworkControlDomain.

3.3.1 Function:NetworkElementControl

Qualified Name:
CoreModel::GeneralControllerModel::ObjectClasses::ExplanatoryClasses::Function:NetworkElementControl

Traditional model of the NE equivalent to an aspect of the NetworkElement class from v1.2.
This class should not be implemented.

This class is abstract.

This class is Example.

3.3.2 Function:SdnController

Qualified Name:
CoreModel::GeneralControllerModel::ObjectClasses::ExplanatoryClasses::Function:SdnController

Traditional model of the SDN controller equivalent to the SdnController class from v1.2.
This class should not be implemented.

This class is abstract.

This class is Example.

3.3.3 View:NetworkElementViewedFromSdnController

Qualified Name:
CoreModel::GeneralControllerModel::ObjectClasses::ExplanatoryClasses::View:NetworkElementViewedFromSdnController

Traditional model of the view of the NE controller as seen from a SDN Controller equivalent to an aspect of the NetworkElement class from v1.2.
This class should not be implemented.

This class is abstract.

This class is Example.

3.3.4 View:SdnControllerViewedFromManager

Qualified Name:
CoreModel::GeneralControllerModel::ObjectClasses::ExplanatoryClasses::View:SdnController ViewedFromManager

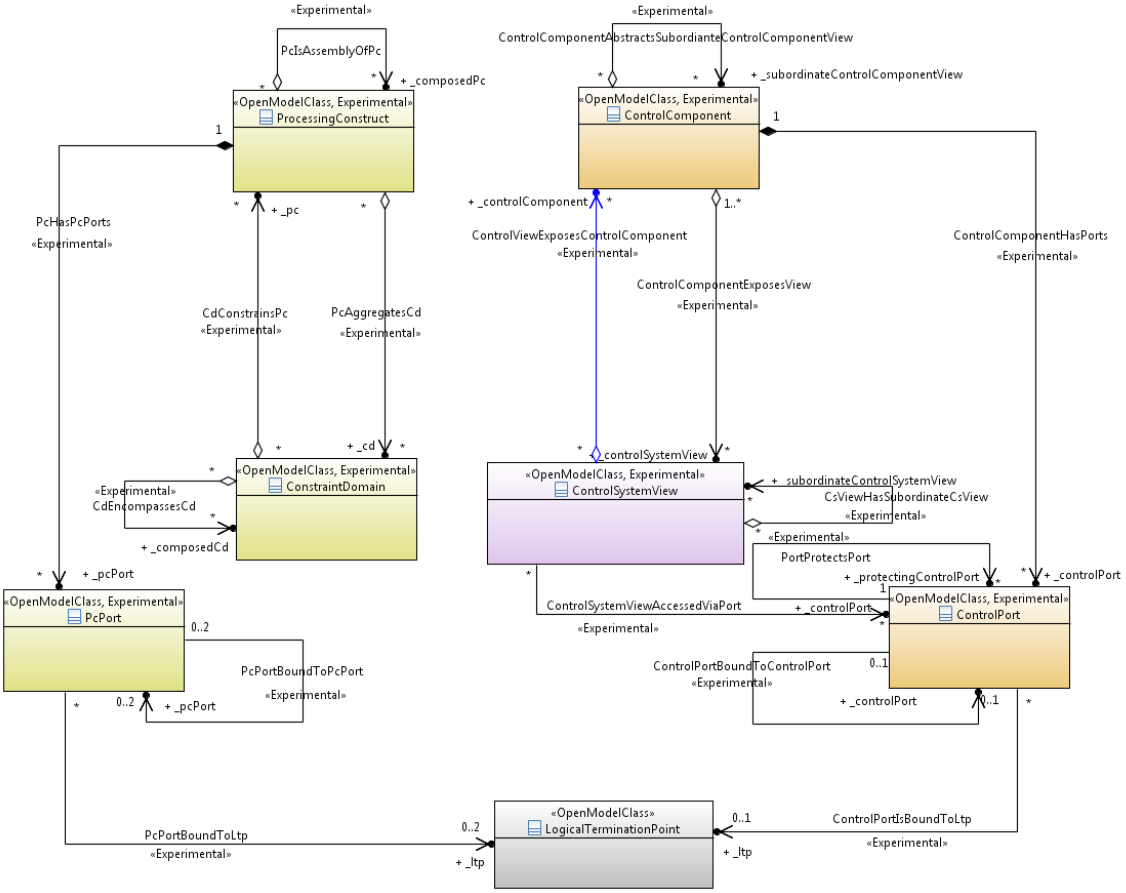
Traditional model of the view of the SDN controller as seen from a manager .
No equivalent in v1.2.
This class should not be implemented.

This class is abstract.

This class is Example.

3.4 Relationship to ProcessingConstruct

The following figure shows the relationship between the Processing Construct (see TR-512.11 and TR-512.A.9) and the key Control model classes. The relationship is illustrated by positioning as there is no formal inheritance (to avoid confusing clutter of inherited associations in the Control classes). The relationship is essentially the adoption of the pattern.



CoreModel diagram: Control-ControlComponentAndControlViewPattern

Figure 3-3 Relationship of Control Model to ProcessingConstruct

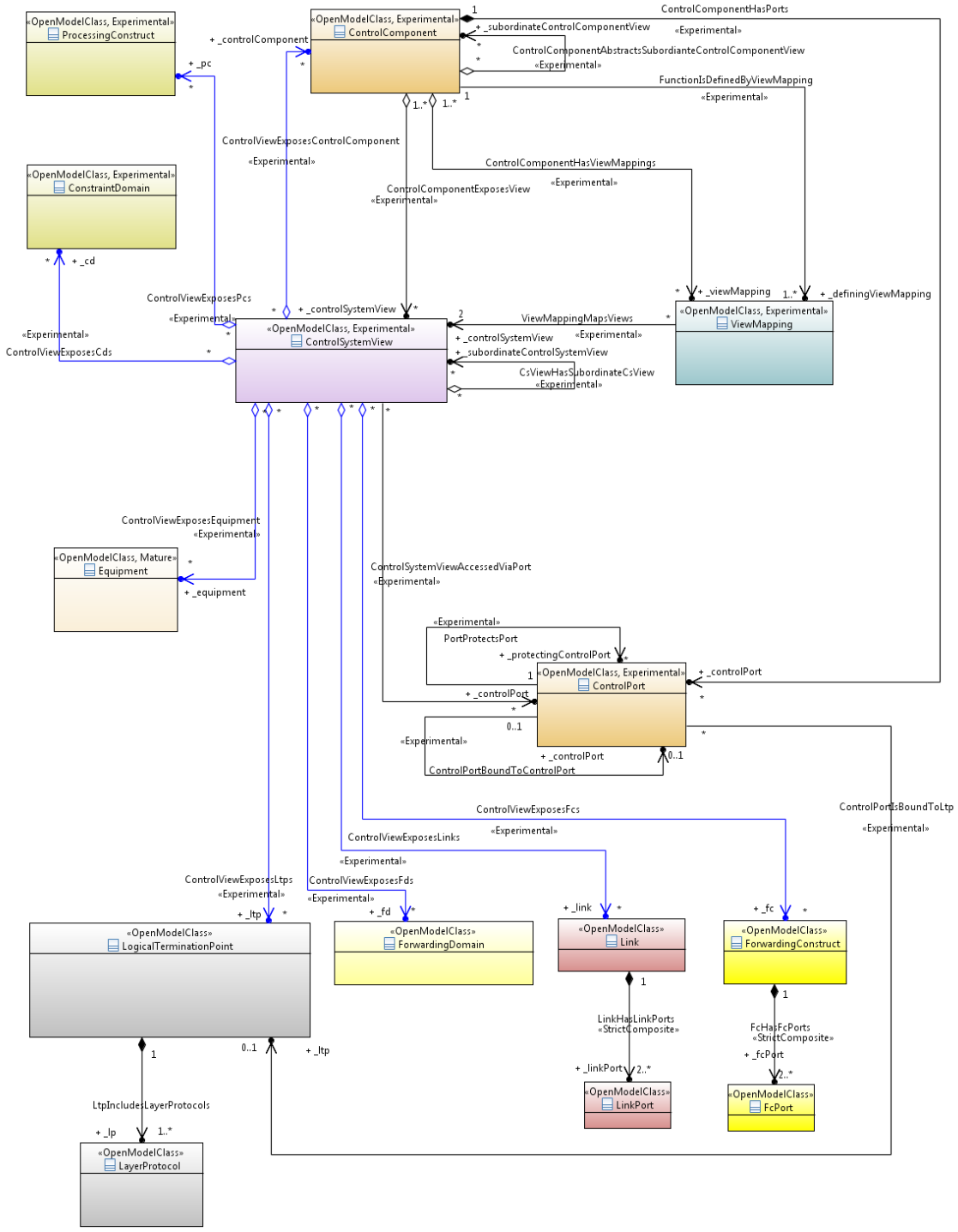
The following key observations should be considered:

- ControlComponent[3] is essentially a type of ProcessingConstruct:
  - _subordinateViewControlComponent is essentially _composedPc but the emphasis is on the view of behaviour rather than actual construction. _composedPc supports both view and actual[4].
- ControlPort is essentially a type of PcPort
  - ControlPort has an additional statement on simple protection which in the PC model would be explicitly modelled (as it is for FC – see TR-512.5)
- ControlSystemView[5] is essentially a type of ConstraintDomain (see TR-512.11):
  - It has an association to ControlPort explaining where it can be acquired from
  - The emphasis is on exposing a constrained set of information and operations
  - It cannot exist in the absence of a ControlComponent

Considering the importance of the Control aspect of the model and the subtle specializations required it has been chosen to use explicit classes for Control as discussed in this document.

3.5 Model in context – directly controlled things

[3] ControlComponent could perhaps be better named ControlConstruct to improve consistency with the remainder of the model. The name of this class is likely to change in the next release.
[4] I.e. there are two distinct usages, the apparent construction as perceived by an external user and the actual construction as perceived by the system presenting the view. _composedPc focusses on actual construction where as _subordinateViewControlComponent focusses on apparent construction.
[5] ControlSystemView could perhaps be better named ControlDomain, however the ControlSystemView is as presented over an interface from a particular viewpoint and on that basis the domain of control is almost always broader than the ControlSystemView. In another sense this is the domain of control available to the viewer.



CoreModel diagram: Control-ControlComponentAndControlViewFullModel

Figure 3-4 Control Model showing Controlled Entities

3.6 General discussion

The key consideration here is that the ControlComponent exposes one or more ControlSystemViews (the replacement for the NetworkControlDomain etc.) which include, via

aggregation, all relevant controlled entities (where a controlled entity is allowed to be in many ControlSystemViews).

For example, the SDN Controller function, which is represented by a ControlComponent (essentially the SdnController in V1.2), exposes its behavior as a set of subordinate ControlComponents. It also exposes the Network Elements it controls as ControlComponents each with one or more ControlSystemViews that include the NE ControlComponents (i.e. the control aspects of the Network Element – the NetworkElement in V1.2) and the aggregation of the subset of the entities from the SDN Controller ControlSystemView that the NE controls. These are presented in the terminology and naming of the SDN Controller.

The subordinate ControlSystemView represented by the superior ControlComponent (e.g. SDN Controller) may be Pruned & Refactored from the ControlSystemView presented by the subordinate ControlComponent (e.g. Network Element). The SDN controller presents some but not all of the capabilities of the Network Element and the capabilities presented are represented by the SDN controller using the ONF CIM but are represented by the managed element using some other model such as that of TL1 or OpenFlow[6].

If a control function (a ControlComponent) in a device is tasked with the control of a function terminating a stream of packets (a termination function) the control function will present a ControlSystemView which includes an LTP that in part represents the termination function. The control function will also represent its own capabilities (perhaps a capability to notify) via other view entities, not detailed here but represented as ControlComponents, aggregated by the ControlSystemView. An example of such a control function is a Network Element SNMP agent (see section 4.1 Rationale on page 18).

As discussed a ControlComponent representing an SDN Controller can present a network level ControlSystemView of the functions of the network of Network Elements that it controls. This may include the LTPs that were presented in the ControlSystemViews by the ControlComponents representing the Network Elements functionality. Depending upon the degree of Pruning & Refactoring, the LTP may be identical in the network view to that presented in the subordinate ControlSystemView and hence the same LTP instance can be aggregated by the ControlSystemView of the ControlComponent representing the SDN controller and the ControlSystemView of the ControlComponent representing the Network Element.

If the Network Element is also controlled by another ControlComponent (along with other managed elements), that ControlComponent will present the Network Element as a ControlSystemView as noted above where the ControlSystemViews from each of the ControlComponents will probably have some entity instances in common.

As any entity can be in many views, as can any subordinate view, the model accounts for controller resilience and control migration. Several different ControlComponents can present the same information at the intersection of overlapping views. The UUIDs in the instances of objects presented in the views provided by the ControlComponents will allow reconciliation[7]. A

[6] The CIM should be used at all levels of view of networking capabilities. Clearly legacy devices will use traditional representation forms.

[7] Each ControlComponent instance has a distinct and different UUID but some of the object instances presented in one view may have the same UUID as object instances presented in another view as they are representations of the

ControlComponent can present the same information in several views. A ControlComponent can present the same information through several ports.

Any representation of a thing in a view could be known to be a fragment (e.g. an FD could represent a fragment of the whole domain where forwarding is possible). This may be determined as a result of explicit or implicit off-network (out of view) relationships within the entity.

For example, an LTP may expose two layers but it is known that there are more layers represented by another controller. It is expected that a superior controller will assemble (union) the fragments to form a coherent single entity using whatever matching criteria are appropriate. If a representation is a fragment, then appropriate match criteria and combination rules will need to be used to identify which fragments to combine to form the whole and what process to use to form the whole.

In a realization of the model it would be possible to subsume a subordinate ControlSystemView in the superior ControlSystemView so that there is a simple aggregation recursion.

4 Understanding the control component and view model

The world of networking has changed as computing and networking converge. It is clear that the implications are significant and there is an opportunity to take advantage of patterns that are apparent when taking a holistic view.

Traditionally Network Element, or a similar concept, has been used to represent a 'logical device'. This concept was easy to understand, especially when a 'device' had only one major function (like a SDH ADM or a PDH channel multiplexer).

As 'devices' have become more complex and multi-functional, the usefulness of the Network Element concept has decreased. For example, initially packet routers and Ethernet switches performed complementary functions. Now we have routers with inbuilt switches and layer2/3 switches, blurring the distinction between them.

Another point of confusion is where the management plane scope and the functional scope were mixed in concepts such as 'Managed Element' or 'Managed Network Element'. This scope confusion is especially problematic when 'devices' are logically partitioned or grouped to form 'distributed devices'.

The key to understanding the way forward is to understand that in a multi-functional 'device', we need to focus on the functions. In hindsight, NetworkElement was just a container, that grew too complex and tried to encapsulate everything and ended up causing a lot of issues.

Reexamining the way of representing networking functionality leads to the Component-System pattern, the Processing Construct and the approach to representation of control discussed in this document. In addition, the model of physical things set out in TR-512.6 cleanly separates genuinely physical things that can be measured with a ruler, from logical concepts. The general

---

same thing. For example an LTP instance in one view may have a UUID of 27 and an LTP instance in another view may also be UUID 27.

approach is careful separation of conceptually distinct concerns into functional, physical and informational and then to further separate functional into control and networking etc.

4.1 Rationale

The ONF Architecture [ONF TR-521] shows a recursion of control. This aligns with the ideas from [TMF IG1118] which:

- Developed the concept of the Management Control Continuum (MCC)
- Emphasized that automation is essentially about closing the control loop
- Explained the recursion of control loops where a control element may participate in one or more loops
- Developed the Component-System pattern
- Emphasized that a Component exposes views
- Explained how a ControlComponent exposed views of itself and what it is controlling to its client (which were potentially simply control components with broader scope)
- Highlighted recursive functional abstractions, where a selection of functional components offered by providers are taken by a client, pruned to give useful function, assembled into a system and the capabilities of that system are offered to clients in various pruned and refactored functional component forms. Offered functional components are then taken by a client and the process is repeated
- Explained that all functional capabilities viewed are abstractions of an underlying system with greater detail and complexity, and are, as a consequence, also virtualized within the scope of the provider system

An SDN Controller will be realized using compute, storage and communications capabilities. Clearly the traditional SDN Controller just like the traditional Network/Managed Element will have communication ports. These communication ports have functionality that is no different from any other function terminating a stream of packets. The functions of communication ports of the SDN Controller are represented using the LTP class. Hence a control device and a transport NE are essentially the same. All such devices are balances of compute, storage and communications capabilities (it is just the specific balance that is different).

4.2 Implications

Three classes from the V1.2 model are obsoleted and replaced:

- SdnController becomes ControlComponentView of a ControlComponent
- NetworkControlDomain is a ControlView related to the ControlComponent that represents the SDN Controller
- NetworkElement becomes:
    - ControlComponentView of a ControlComponent (where the control component is the NE applicable when accessing the NE from a SDN Controller) and the rest of the ControlView (i.e. the LTPs etc.) related to the ControlComponent that is the NE
    - SubordinateExposedViews in the ControlView of the SDN Controller which provides a further ControlView that includes the NE ControlComponentView (i.e. the control aspects of the NE – the NetworkElement in V1.2) and the aggregation

of the subset of the entities from the SDN Controller ControlView that the NE controls. These are presented in the terminology and naming of the SDN Controller.

The relationship between the ControlView and the things in the view is aggregation and not composition as it was in a traditional model of an NE.

In addition the C&SC is essentially a specialized ControlComponent.

We can use ControlComponentView to represent :

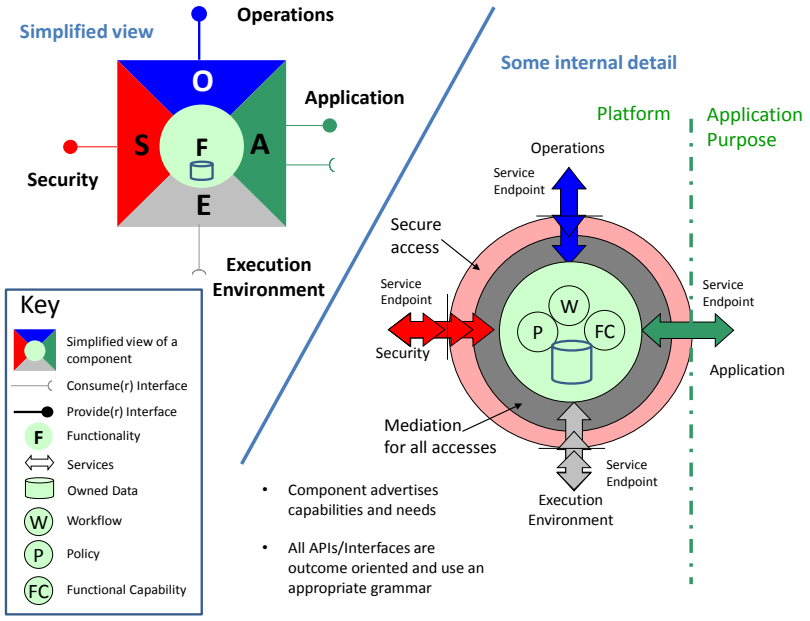
- a logical scope that aligns to a physical inventory boundary (especially useful for 'device partitions' and 'distributed devices')
- a management scope (which may differ from the physical and functional scope)
- a general functional scope that can be used for grouping and scope boundaries

While the move to replace NetworkElement with ControlComponent and ControlSystemView was prompted by issues in representing 'traditional devices', it can be seen that (along with the existing decoupling of functional and physical viewpoints) this now gives a neat and consistent representation of SDN and NFV implementations, where the NetworkElement concept is largely irrelevant anyway.

4.3 The patterns behind the model

As for all components, the ControlComponent has ports. The ports provide access to the ControlViews and allow control of the ControlComponent.

A helpful view of this is provided by [TMF IG1118] as shown below.



[TMF IG11118] Figure 1 The FMO component interface and structural overview

Figure 4-1 A Controllable Component

A Component has an Operations port through which it may be controlled/managed[8] and an Application port through which it exposes it purposeful behavior. The purposeful behavior of a Control Component is related to the controlling of other Components, A Control Component has an Operations port through which it is controlled.

As discussed in TR-512.A.2, all functional capabilities of the network are represented in the form of Components (FC, LTP, PC etc.). Likewise, the functional capabilities of the control system can be represented in the form of Components (e.g. C&SC).

The ports of the control components used for signaling can be represented using LTPs and the Control Functions that terminate the signaling can be represented by Control Components such as C&SC. Where appropriate, the signaling itself can be represented via a protocol definition perhaps using the Generalized operations pattern (see TR-512.10).

4.4 Identifiers, naming and addressing

In general, there is a need for separate spaces of identifiers/addressing for:

- Ports
- Control functions
- Management-Control views
- Functions (Virtual)
- Physical things
- Mixed assemblies of Functions and Physical things
- Places

When a controlled thing does not have a native UUID that can be used consistently across Control Views, there needs to be some directory service to provide consistent identification.

4.5 Resilience in the Control System

By separating the identifier spaces for Control from the spaces of the things being controlled and by loosening the association from composition in a traditional model to aggregation, the Control model is then set up appropriately to allow for well identified instances of controlled things to appear in more than one ControlView. As a consequence, various controller resilience schemes are readily supported.

4.6 Controller view considerations

The figure below highlights the pattern of talking through a port to a controller about a controlled system where that system:

- Includes the controller itself
- Is represented in terms of components
- Is represented via some pruning & refactoring transform

[8] A component provides a façade through which it can be controlled…. This essentially provides access to an embedded controller which is at the lowest level of "visible" recursion (degenerates to a transistor gate etc).

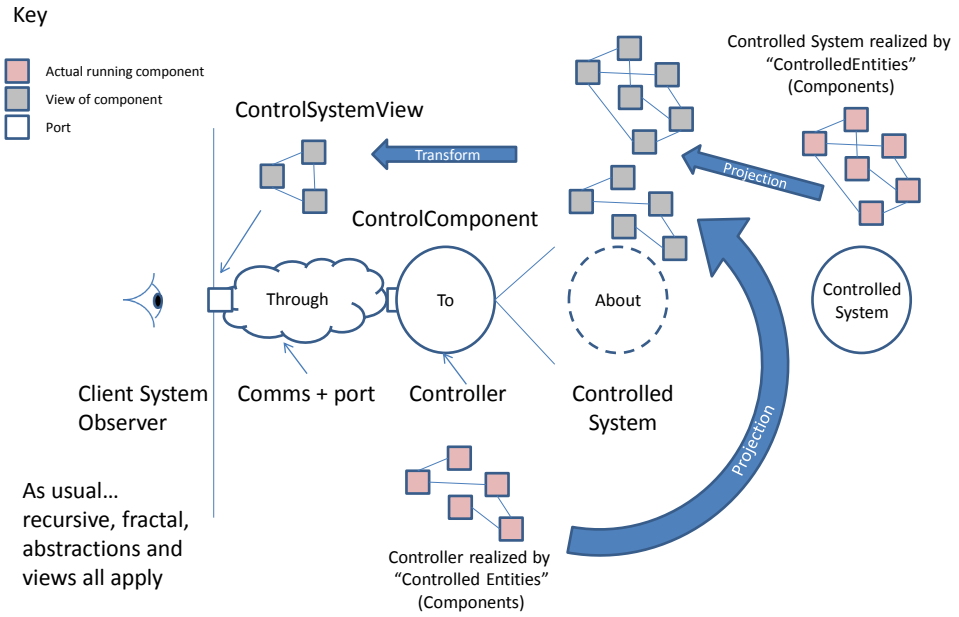
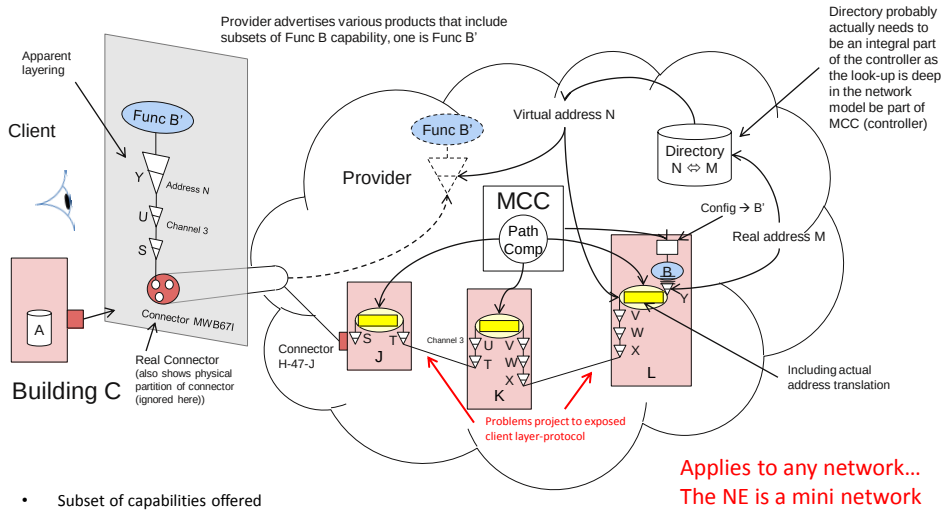


Figure 4-2 Through, To, About…

The figure below shows the perception of a complex network as viewed by the Client. The ControlSystemView will include precisely the functional components perceived by the Client. The perceived functions are an abstraction of the actual network and are also virtualized in that the Client does not know, or care, where the functions actually are. The figure shows a network that has a function "B" that is exposed as "Func B' " to the Client.

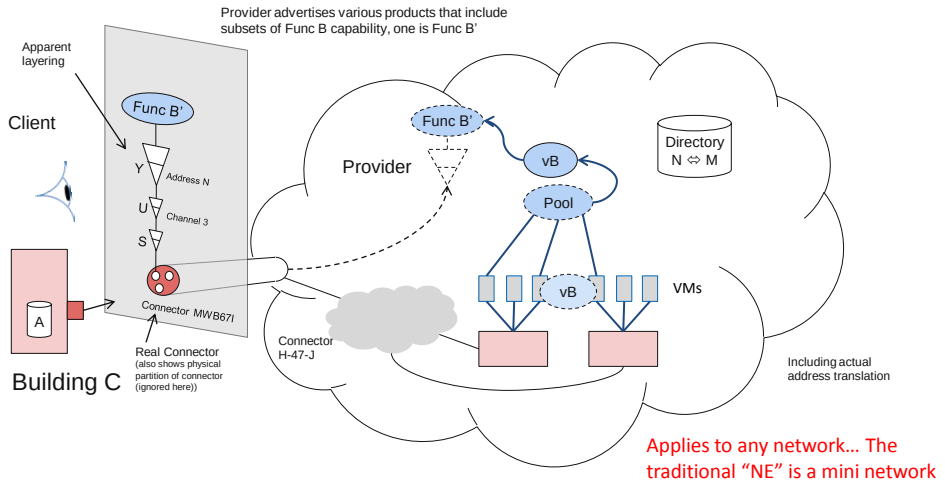


- Subset of capabilities offered
- Address translation necessary and measures need to map via address translation
- Alternative instance of function may be selected after network restoration. Apparent address remains the same and function appears continuous to the observer
  - Performance data must "move" with route and function move
- Problems will project to client layer-protocols (U and Y)
- For retrospective diagnosis and analytics details of positions and moves must be maintained

Figure 4-3 Simple network view mapping

The figure below shows a network that has a virtual function "B" (virtual) that is exposed as "Func B' " to the Client. The view provided to the client is the same as in the previous figure although the realization in the network is quite different

View mappings – function running on a VM



As previous… In addition

- Function may move to different platform whilst apparent address remains the same and function appears continuous to the observer
  - Performance data must "move" with function move
- Problem in server is reflected through effect of VM on function in pool

Figure 4-4 View mapping for functions on a VM

The figure below shows a client view of various control interfaces related to a particular simple network service. The same pattern applies at all levels and as a consequence the same model can be applied at all levels. Traditionally different models have been applied.

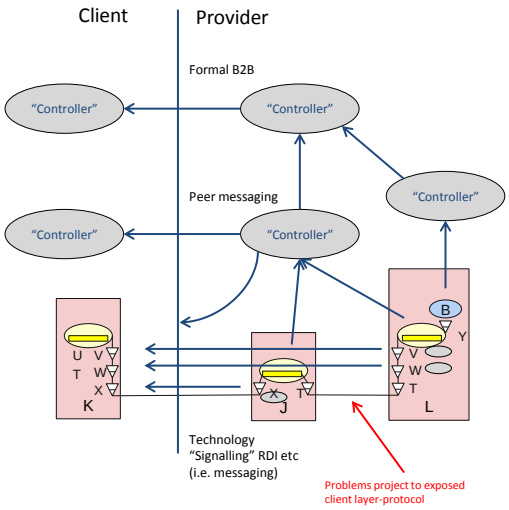


Figure 4-5 Client view of network and control

The diagram above highlights the following:

- Signalling is messaging
- Network device essentially has embedded controller
  - The embedded controller generates messaging at the "network technology level" (traditionally called signalling)
- Messaging at the network technology level is "immediate" but provides minimal information and hence causes somewhat "knee-jerk" actions
- Higher controller provides richer information but with reduced immediacy
- Higher controller may drive network technology level messaging (signalling)
- In the longer term embedded controller become part of the continuum
- Approach to messaging source depends upon trust and information usage

The figure below shows a simplified picture of the client view of an actual service (capability) and view of control of that capability. The figure uses the symbol set highlighted earlier in this section from [TMF IG1118]

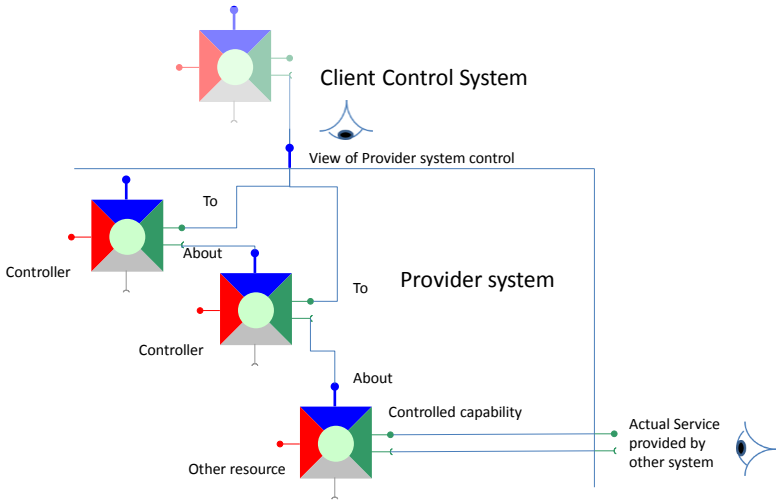


Figure 4-6 Simplified view showing exposure of controllable capability to a client

4.7 Dismantling the NE – Some rationale

The Network Element (NE) concept has been around for a long time.

- A Network Element is defined in US law[9] as "Network element is defined as a facility or equipment used to provide a telecommunications service. Such term also includes features, functions, and capabilities that are provided by means of such facility or equipment, including subscriber numbers, databases, signaling systems, and information sufficient for billing and collection, or used in the transmission, routing, or other provision of a telecommunications"
- [ITU-T Q.1741.9] defines NetworkElement as "A discrete telecommunications entity, which can be managed over a specific interface, e.g., the RNC."
- [ITU-T G.780] defines "network element (NE)" as "A stand-alone physical entity that supports at least network element functions (NEFs) …"

The NE is a somewhat messy thing. One of the issues we have is that existing representations make a number of assumptions that aren't true in many cases. To avoid confusion by redefining the existing concepts, new terms are required to clearly define what it is and isn't.

[9] https://definitions.uslegal.com/n/network-element/

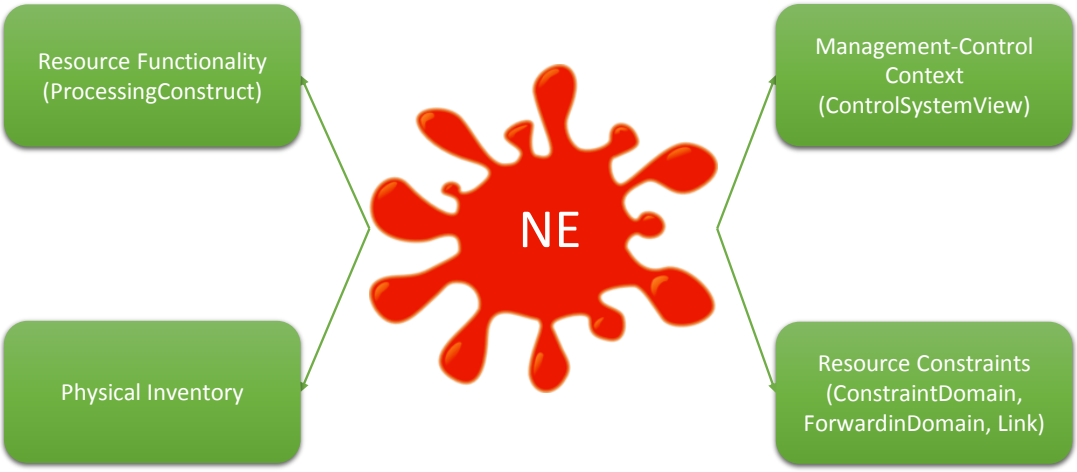


Figure 4-7 The "NE"

A much cleaner, recursive and consistent model has been formulated that takes advantage of the Control-View model discussed above.

The following section discusses the rational for dismantling of the NE.

4.7.1 The analysis

Looking broadly at the drivers from earlier sections:

- The Management-Control Continuum, as identified by TM Forum, extends down through the SDN Controller into the NE such that an aspect of the NE is a controller
    - The SDN Controller looks like any other manager/controller
    - The NE looks, in part, like any other manager/controller
- A generalized model of control, access to control and control scope will provide a basis for a coherent reworking of both the NE and SDN controller representation
- The SDN Controller, like the NE, needs to present a representation of the functionality it is controlling as well as to present itself as a set of control functions
- There appears to be a need for a generalized representation (pattern) of a coherent unit of functionality
    - To cover both control functions and controlled functions
- Just as for the NE, there needs to be a representation of the relationship between the function (of control and being controller) and their physical realization
    - The representation of physical realization using the Equipment model will bring geographical positioning information
        - The control/communications channels for both the NE and the SDN Controller look like any other communications
            - The representation of communication channels using FC/LTP will link with the remainder of the communications network

The Network Element (NE), as concept, is a somewhat incoherent hybrid of various concerns where the hybrid is not viable for many cases. One aspect of the NE is control and this should be represented and considered in the same way as any other controller. The control aspect is the primary focus of a Managed Element (ME) but this also suffers from the same lack of coherence.

Clarity is brought by considering the separable concerns:

- Physical thing (solid i.e. a thing that can be measured with a ruler and has weight) and Physical space (i.e. with volume but no relevant weight)
    - A coherent physical thing that in context is not relevantly decomposable (component, atomic)
    - A coherent assembly of physical things (system/assembly, composite)
    - Similarly physical space
- Positioning of the physical thing in geographical space
    - Essentially a point in space (very small geographical area)
    - A large geographical area
- Virtual[10] function emergent from a physical thing where the virtual function has capability and is potentially active
    - A coherent virtual thing that is in context not relevantly decomposable (component, atomic)
    - A coherent assembly of virtual things (system/assembly, composite)
    - Only realisable via supporting physical things (see TR-512.6 for details of the relationship between the models of physical and functional things).
- Management-Control function, Management-Control scope and access to Manage-Control where that Management-Control function
    - The functions that fulfil and assure the intent and that provide access (can be talked to) to a view of things (that can be talked about)
    - Is itself a virtual function
    - Can view and manipulate virtual functions
    - Can provide a view of Physical things through virtual functions
- Port through which to access management-control information
    - Will necessarily be a partial view of information of each thing that can be viewed
    - May overlap with the view provided via another management access (such that some things are seen partly through one port, partly through another and partly through both)
    - May allow access to information on geographically distributes things
    - May allow access to information representing fragments of functionality some of which may be completely disjoint from others
- A named hybrid assembly of virtual and physical things spread over an arbitrary geographical area
- The assembly of information that can be accessed through a management port

The NE is a mix of the above (as is the SDN Controller, the EMS etc.). The challenges with the above conglomeration approach:

- Inconsistent boundaries

[10] Also called Logical Function.

    - The boundary of a coherent physical thing is highly unlikely to be coincident with a coherent virtual thing
    - The boundary of the visibility via the management access is likely to cut across the boundary of physical and virtual things
    - Some disjoint things are accessible via the same management access
- Geographical spread
    - The management access may be to things that are spread across geography and hence:
        - Themselves do not have shared fate
        - Have shared fate with things accessible via other management accesses
- Identity and name challenge
    - Each instance of the concept has identity and some form of identifier in a context that allows identification and potentially allows location via some form of address
    - The identifier for the management access may differ from the identifier for the various virtual things and for the various physical things accessible
    - The same thing may be accessed via management accesses of several different controllers
- Lifecycle fragmentation
    - A virtual thing visible via the management access may persist beyond the life of the management access etc.
- The assembly of information that can be accessed through a management port
    - For a geographically distributed "ME/NE" it is potentially necessary to open up the "ME/NE" to understand its cabling etc. and fate share with other systems
    - An "ME/NE" may group multiple "subnetworks" and have internal interconnecting "links"
- A composite "ME/NE" may provide access to disjoint functions that have independent network purpose
    - For example, an FRU that only draws power and perhaps receives basic control and that has no functions relevant to the rest the FRUs in a shelf that forms part of an NE
- Some things may be accessible as if in two different "MEs"/NEs"

Considering the current model clearly physical and functional things can be represented. Hence the focus of the model to replace the NE is the control view and the control entities themselves (the control entities are controllable).

- The control entities can be considered as Components.
- In a controller view, there is potentially a view of the view provided by the subordinate controller (and so on)
    - The critical consideration is what needs to be exposed. The "NE" exposes a view. The controller of the NE "may choose" to expose a view which may include the NE view or an abstraction of it (which the controller may claim is the NE view
- A view is accessible through a port and a port is an LTP (which is a component-system)
    - There is an address of the port at which the information the controller expose is available

All systems involved in Control (e.g. NE, EMS, NMS, SDN Controller, Orchestrators) can be treated in the same way.

- The views are aggregation. The provider of the view can be removed without the system ceasing to function
    - The lifecycle of the presentation is independent of the lifecycle of the presenter
    - A view may be provided through several accesses. An LTP could be visible through multiple views
    - There could be fragments of entities provided in a view where the whole entity is made by assembling information from several views
    - It is the ControlEntity that is requested to perform actions on the things presented through the view
- NE cases illustrating points on the broad spectrum
    - A simple regenerator which is a single piece of hardware with one function and two… this is clearly representable as a traditional NE (single Geographical place etc.)
    - The DSL case with a direct access to the remote and a head end that consolidates the remote. If I consider monolithic NEs then there is a problem, if I consider views then there is no problem.
- Control of a "white box" NE will benefit from this approach
    - The views are decoupled from the physical platform and from the ControlEntity. They can move. The location of the producer of the view is determined via the relationships to the equipment model.
        - Equipment gives rise to function gives rise to complex function gives rise to LTP
- There is no need to create a virtual NE or virtual hardware.
    - Simple view based or domain based groupings of functionality covers all cases

The following figure shows an NE that happens to be significantly geographically distributed.

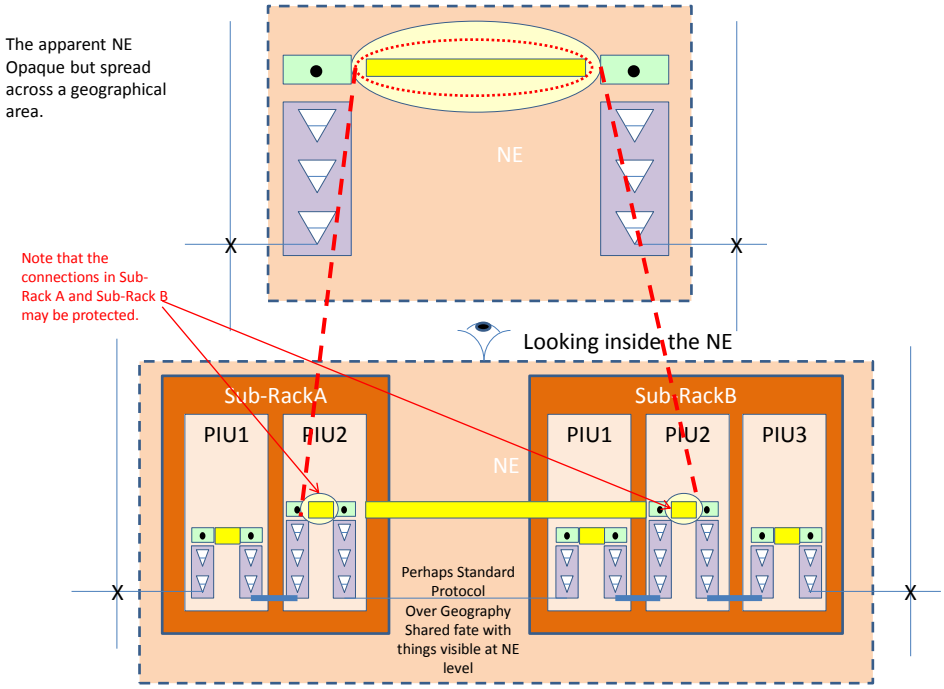


Figure 4-8 Geographically distributed NE

In the figure above:

- A subset of functions form a coherent unit of stand-alone network function
- There is significant geographical distance between two functions accessible through the control interface

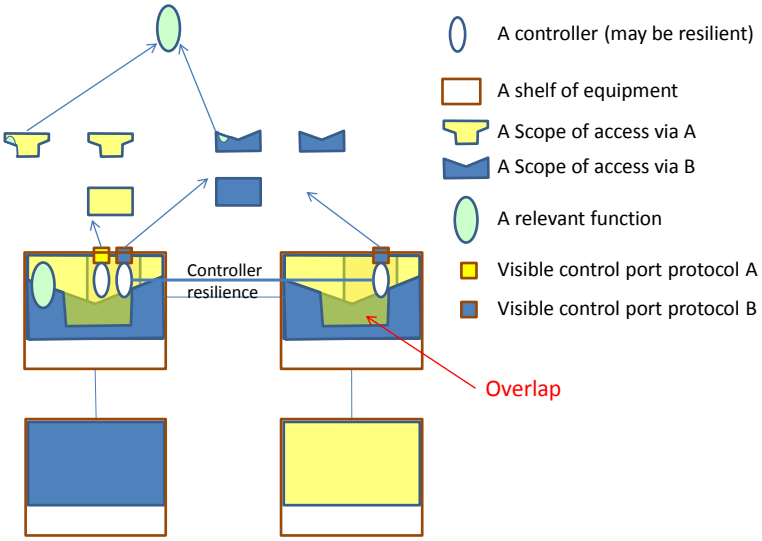


Figure 4-9 An NE with two control access ports each providing a partial view

In the figure above, an assembly of equipment forms a traditional NE that happens to have two control access ports, each providing a partial view. Part of a relevant function (e.g. an LTP) is accessible through one control interface and part through another.

4.8 The control model applied to the "Controller"

The control model discussed here can be applied to any manager/orchestrator/controller. The ControlComponent can be used to represent any control functions. If a more detailed functional model of the Controller is required, the model described in this document can be supplemented with the ProcessingConstruct/ConstraintDomain (see TR-512.11). The Controller model is not fully developed in this release.

4.9 The configurationAndSwitchController (C&SC)

The C&SC is is described in TR-512.5. It is a specialized ControlComponent used for control of forwarding resilience. It is expected that the C&SC, the Control model described here and the PC/CD model will be further refined in subsequent releases.

End of document