Network Markup Language Base Schema

Jeroen van der Ham Martin Swany Freek Dijkstra
Lars Fischer

April 26, 2010

Abstract

A recommendation document describing a normative schema which allow the description of a basic network topology. This schema does not include any layer or technology specific information.

1 Introduction

This document describes the object model of the Network Markup Language. This is essentially an abstract schema. These basic objects will be extended, or sub-classed, to represent concrete instances and to articulate their particular attributes. These basic objects and extended objects will also be representable in multiple syntaxes, including at least XML and RDF.

2 NML Topology Schema

The NML Topology schema describes an information model with elements and their relations that describe computer networks. This schema is kept intentionally general, with provisions to extend the base schema to describe layer-specific information.

TODO: Determine URI.

2.1 Network

This document explicitly does not try to provide a definition of the term 'Network'. The working group has discussed the possible meanings of this term and in the end we were forced to conclude that is not possible to provide a workable definition for the term *Network*. The term *Network* has become so widely used for so many diverse meanings that it is impossible to create a definition that everyone can agree on, while still expressing something useful.

2.2 Network Object

The basic abstract element of the schema is the *Network Object*. Other basic elements inherit from it. The *Network Object* can have a *Location*, can be related to other instances via a *Relation* and can be described by a *Lifetime*. Every Network Object MUST have an *id* attribute, which MUST be a unique URI. These characteristics are inherited by the subclasses of the *Network Object* class.

The base *Network Object* has three elements that describe it and its relationships:

- Location
- Lifetime
- Relation

The location of an object in the physical world can be described using the *Location* object. The actual location is then described using properties of the *Location* object.

All network objects can potentially have a *Lifetime*, that consists of vector of *time* elements, which contain a start time and an end time.

The Relations between different network objects are represented using relation objects. These are discussed in more detail in section 2.10.

The base *Network Object* is subclassed into the top-level topology components, that are sufficient to cover the description of networks. The top-level network objects in this schema are:

- Node
- Port
- Link
- Service
- Group

These objects are described in more detail below.

We also have a separate *Layer* object to describe layers. This is described in more detail in section 2.8.

2.3 Node

A *Node* is generally a device connected to, or part of, the network. A Node does not necessarily correspond to a physical machine. It may be a virtual device or a group of devices.

In this case, the *implemented by* relation can be used to describe this.

A Node is connected to the network by its *Ports*. A Node provides *Services*.

The Relations of Node:

- A *Node* MAY share a *has port* relation with one or more *Ports*.
- A *Node* MAY share a *located at* relation with one *Location*
- A *Node* MAY share an *implemented by* relation with one or more *Nodes*.

2.4 Port

A *Port*, or interface, connects a *Node* or *Group* to the rest of the network.

A *Port* is related to zero or one *Node* or *Group*, and also has a relation with zero, one or two (uni-directional) *Links*.

To adapt traffic to and from different layers, an *adaptation* relation is used. An adaptation consists of two unidirectional components, an *adaptation sink* to go from a server layer port to a client layer port, and an *adaptation source* for the other direction.

A Port can have up to two adaptation sink relations, one for a server layer, and one for a client layer. The same applies for adaptation source relations.

Relations of Port:

- A *Port* MAY have a up to two *adaptation sink* relations.
- A *Port* MAY have a up to two *adaptation source* relations.
- A *Port* MAY have a *source* relation with up to two *Unidirectional Links.*
- A *Port* MAY have a *sink* relation with up to two *Unidirectional Links.*

2.5 Link

A *Link* object describes that there is a unidirectional connection from one *Port* to another. These ports are identified using the *source* and *sink* relationships.

A *Link* MUST have an attribute *type* which is either *Link* or *Crossconnect.* When the type is Crossconnect, the source and sink MUST be part of the same node.

A *Link* should have a *capacity* attribute which describes the capacity of the link in bytes per second. This value should correspond to the actual amount of data that can be transported over the link, excluding overhead.

Relations of Unidirectional Link:

- A *Link* MAY have a *source* relation with one *Port.*
- A *Link* MAY have a *sink* relation with one *Port.*
- A *Link* SHOULD have a *capacity* attribute which describes the capacity of the link in *bytes per second.*

2.6 Service

A *Service* object describes a certain capability being offered by a Network Object. The key idea that this is a generic container representing any service that a network-centric user or agent might want to discover and use. Below we describe some example Services, but others are also possible.

SwitchingMatrix describes the ability of a network object to create cross connects between its different ports.

Configured cross-connects should be described using the *switched to* relation.

Adaptation describes that ports on different layers within a node can possibly be connected together to form a connection or cross connect on a different layer. The Adaptation service must define both its client and server layer.

Once this is implemented it is described using the *adaptation source* and *adaptation sink* relations between ports.

Measurement Point services are an essential component of network measurement services like perfSONAR.

2.7 Group

To describe collections of network objects, there is a group element. Any element defined above can be part of a group, including another group.

We also define a set of special groups:

- Bidirectional Link
- Topology
- Path
- NetworkDomain

2.7.1 Bidirectional Link

A *Bidirectional Link* is a special group of two (unidirectional) *Links* together forming a bidirectional link between two ports.

2.7.2 Topology

A *Topology* is a set of Network Objects and the links connecting them.

2.7.3 Path

A *Path* is an ordered collection of Network Objects.

A Path describes a path taken through the network from the source, the first element, to the destination, the last element. This can be used to represent a routed path in an IP network, as well as the path that a virtual circuit takes. These can also be partially specified in a request.

2.7.4 NetworkDomain

A *NetworkDomain* is an unordered collection of Network Objects managed under the same shared mechanism.

2.7.5 PolicyDomain

A *PolicyDomain* is an unordered collection of Network Objects managed under the same policy umbrella.

2.8 Layer

A *Layer* is a collection of Ports with common Characteristic Information. The term Characteristic Information is defined in ITU-T Recommendation G.800[**?**]: TODO: Summarize this.

> Transport information entities, being themselves instances of information, exist separately as three forms of information:
>
> a) Client information: This is the communication that the client requires to be transported transparently and accurately.
>
> b) Adapted information: Adapted information is the information that is transported transparently across a server layer network. Adapted information is the client information encoded in such a manner that it is transportable across the layer network. This encoding can include labelling of the client information in order to distinguish the client information within the context of a single instance of adapted information. Adapted information is the construct that allows independence between client and server.

c) Characteristic information: This is the combination of the adapted information with additional information (layer information) that is transported across the network. Some of the layer information can remain unchanged across the network, though it can be read within the network, while other layer information may be altered within the network.

The only information which can be read (and by implication understood) within the layer network is the layer information which is added to the adapted information to form the characteristic information and is added by a termination function. This is irrespective of whether or not the symbols encoding the information are changed within the layer network. For example, if the layer network needs to read a field for its operation, this field is not transparent to the layer network and cannot be part of the adapted information. This field must be treated as belonging to the termination function.

The *Layer* object describes a specific network technology over which data can be transported. Ports are members of the Layer group at which they operate. Layer objects should reference identifiers as specified in Deliverable 3.

The possibility of moving data from one layer is described by the Adaptation service. This service specifies its server and client layer to describe between which two layers the data can be transported.

Configured adaptations from one layer to the other are described using the *adaptation sink* and *adaptation source* objects. The first is a relation between two ports at different layers to describe that data can flow from a server layer port to a client layer port.

2.9 Adaptations

Adaptations are defined in three different ways:

AdaptationType for the generic type of Adaptations

AdaptationService describes a capability of performing an Adaptation (instance of a Service)

Adaptation instance of an abstract AdaptationType to describe the adaptation being performed in a node.

Adaptation is 'Actual data transport function where data of one port is embedded in the data of another port.'

2.10 Relation

Relations describe how different network objects can be combined to form a network topology description. The relations have been described above, but for ease of reference we also give a full list and definition here (in alphabetical order):

adaptation is the bi-directional equivalent of the source and sink adaptation combination.

adaptation sink goes from a server layer port to a client layer port, describing the way that data is passed between these two layers.

adaptation source is the reverse of the sink adaptation, i.e. it goes from a client layer port to a server layer port.

at layer is a relation between a port and a layer to describe the layer at which the port operates.

has port describes the relation between a node or group and a port.

has service describes the relation between a node or group and a service that it provides.

implemented by is a relation from a node to a node, describing that the source node is a virtualized node on the sink node of the relation.

located at is a relation between a network object and a location object.

sink is the connection between the end of the link and the sink port.

source describes the connection of a port to the source port of the link.

switched to defines a relation between two ports, meaning that traffic from the source port is automatically forwarded to the destination port.

2.11 Summary

Figure 1 shows an overview of all the objects in the NML schema in a UML class diagram. The figure also shows the relations between the objects, and their cardinalities.

Note: this schema diagram is still under discussion, any comments are greatly appreciated.

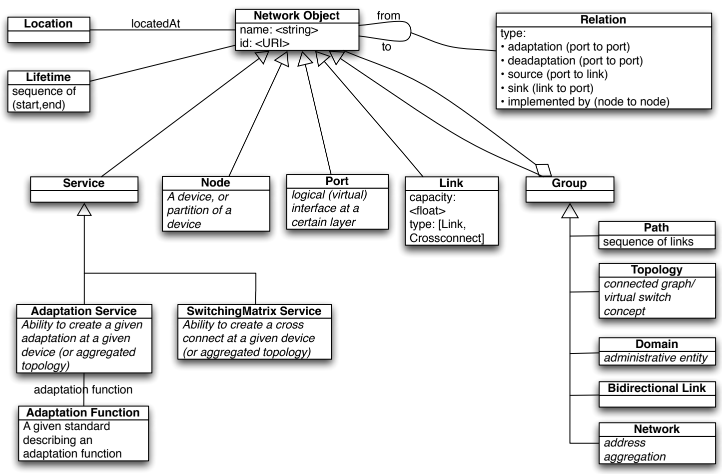

Figure 1: A UML class diagram of the objects in the NML schema and their relations

3 Identifiers

4 Examples