Network Markup Language Base Schema

Jeroen van der Ham Freek Dijkstra Jason Zurawski
Martin Swany

February 16, 2012

Abstract

A recommendation document describing a normative schema which allows 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. These basic objects may be extended, or sub-classed, to represent technology specific classes. 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.

The URI of the class-objects will become `http://http://schemas.ogf.org/nml/base/yyyy/mm/` where `yyyy/mm` is the four digit year and two digit month of the publication of the schema document.

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 precise 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 related objects that describe the *Network Object* 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. The Location and a Network object are related to each other using the *locatedAt* relationship.

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 2.9.

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.

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 the virtualization.

A Node is connected to the network by its *Ports*. A Node can provide *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* object is unidirectional.

A *Port* is related to zero or one *Node* or *Group*, and also has a relation with zero, or one (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 one *adaptation sink* relations.
- A *Port* MAY have one *adaptation source* relations.
- A *Port* MAY have one *source* or *sink* relation with a *Unidirectional Link*.

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 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
- Bidirectional Port
- Topology
- Domain
- Network

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 Bidirectional Port

A *Bidirectional Port* is a special group of two (unidirectional) *Ports* together forming a bidirectional representation of a physical or virtual port.

2.7.3 Topology

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

2.7.4 Domain

A *Domain* represents an adminstrative domain as a collection of resources part of that Domain.

2.7.5 Network

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

2.8 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.'

2.9 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.

2.10 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.

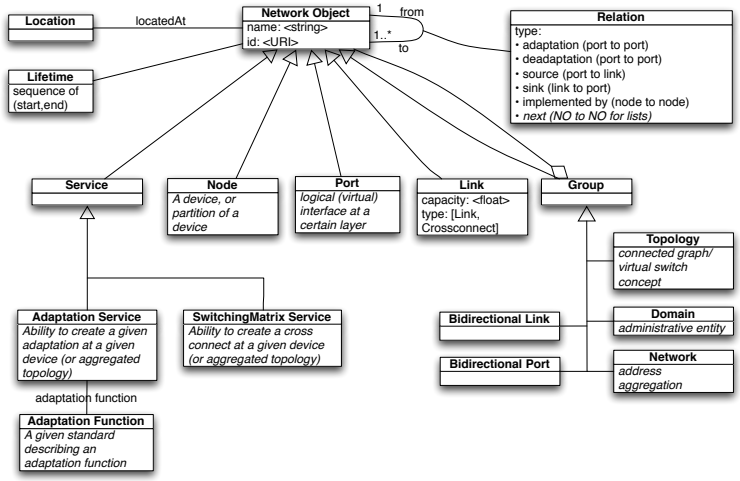

Figure 1: A UML class diagram of the objects in the NML schema and their relations

3 Identifiers

4 Object Identifiers

The namespace for the class objects defined in this document is `http://schemas.ogf.org/nml/base/20`
TODO: change to correct year and month of the schema.

All objects and attributes defined in this document reside in this namespace. For example, the link object is identified by `http://schemas.ogf.org/nml/2013/10/base/link`

5 Instance Identifiers

Section 2.2 requires that instances of Network Objects MUST have an *id* attribute, which MUST be a unique URI.

Implementations that receive a network topology description MUST be prepared to accept any valid URI as an identifier.

Implementations that publish a network topology description instance identifiers MAY adhere to the syntax of Global Network Identifiers as defined in **??**, which ensures global uniqueness and that easy recognition of Network Object instances.

Two different Network Objects instance MUST have two different identifiers.

Once an identifier is assigned to a resource, it MUST NOT be re-assigned to another resource.

A URI MAY be interpreted as an International Resource Identifier (IRI) for display purposes, but URIs from external source domains MUST NOT be IRI-normalised before transmitting to others.

5.1 Lexical Equivalence

Two identifier are lexical equivalent if they are binary equivalent after case-normalisation.

No interpretation of percent-encoding or PUNYCODE decoding should take place.

For the purpose of equivalence comparison, any possible fragment part or query part of the URI is considered part of the URI.

For example the following identifiers are equivalent:

```
1 - urn:ogf:network:example.net:2012:local_string_1234
2 - URN:OGF:network:EXAMPLE.NET:2012:Local_String_1234
```

while the following identifiers are not equivalent (in this case, the percentage encoding even make URI #3 an invalid Global Network Identifier.):

```
1 - urn:ogf:network:example.net:2012:local_string_1234
3 - urn:ogf:network:example.net:2012:local%5Fstring%5F1234
```

5.2 Further Restrictions

An assigning organisation MUST NOT assign Network Object Identifier longer than 255 characters in length.

Parsers MUST be prepared to accept identifiers of up to 255 characters in length.

A Parser SHOULD verify if an identifier adheres to the general URI syntax rules, as specified in RFC 3986.

Parsers SHOULD reject identifiers which do not adhere to the specified rules. A parser encountering an invalid identifier SHOULD reply with an error code that includes the malformed identifier, but MAY accept the rest of the message, after purging all references to the Network Object with the malformed identifier.

5.3 Interpreting Identifiers

A Network Object identifier MUST be treated as a opaque string, only used to uniquely identify a Network Object. The local-part of a Global Network Identifier MAY have certain meaning to it's assigning organisation, but MUST NOT be interpreted by any other organisation.

5.4 Network Object Attribute Change

A Network Object may change during its lifetime. If these changes are so drastic that the assigning organisation considers it a completely new Network Object, the assigning organisation should be assigned a new identifier. In this case, other organisations MUST treat this object as completly new Network Resource.

If the assigning organisation considers the changes are small, it MUST retain the same identifier for the Network Object, and use some mechanism to signal it's peers of the changes in the attributes of the Network Object.

6 Examples