Network Markup Language Base Schema

Jeroen van der Ham Martin Swany Freek Dijkstra
Lars Fischer

August 21, 2009

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

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 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 elements 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 2.7.

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 elements in this schema are:

- Node
- Port
- Link
- Service
- Group

2.2 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*.

2.3 Port

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

A *Port* is related to zero or one *Nodes*, 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.

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*.

2.4 Link

A *Link* is a unidirectional link, it provides connectivity 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.

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*.

2.5 Service

A *Service* object describes the capabilities of node in terms of switching and adaptation between different technology layers. There are two different kinds:

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

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

2.6 Group

To describe collections of network elements, 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
- Path
- Topology
- Domain

2.6.1 Bidirectional Link

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

2.6.2 Topology

A *Topology* is a group of network elements with the restriction that this group must be connected.

2.6.3 Path

A *Path* is an ordered collection of network elements.

A Path describes a path taken through the network from the source, the first element, to the destination, the last element.

2.6.4 Domain

A *Domain* is an unordered collection of network elements. The *type* attribute can be used to define what kind of Domain is meant.

The value of the type attribute is unrestricted, but several predefined values and their meaning are given below:

- *User*
- *Linking*
- *...*

2.7 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.

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

has service describes the relation between a node 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.

2.8 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.

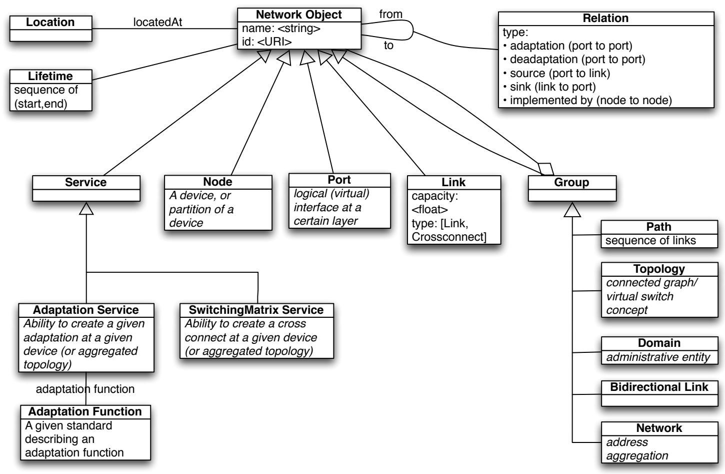
Location
locatedAt
Network Object
name: <string>
id: <URI>
from
to
Relation
type:
• adaptation (port to port)
• deadaptation (port to port)
• source (port to link)
• sink (link to port)
• implemented by (node to node)
Lifetime
sequence of (start,end)
Service
Node
A device, or partition of a device
Port
logical (virtual) interface at a certain layer
Link
capacity: <float>
type: [Link, Crossconnect]
Group
Path
sequence of links
Topology
connected graph/ virtual switch concept
Domain
administrative entity
Bidirectional Link
Network
address aggregation
Adaptation Service
Ability to create a given adaptation at a given device (or aggregated topology)
SwitchingMatrix Service
Ability to create a cross connect at a given device (or aggregated topology)
adaptation function
Adaptation Function
A given standard describing an adaptation function

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

3 Identifiers

4 Examples