Framework for path finding in multi-layer transport networks

Dijkstra, F.

Publication date
2009
Document Version
Final published version

Chapter 6

Multi-Layer Network Description Language

6.1 Goal

In section 3.3.3 of chapter 3 we asked ourselves if it is possible to create a technology-independent model and syntax to describe current multi-layer transport networks?

In the previous chapter, we introduced the Network Description Language (NDL) to describe transport networks. However, the schemata presented there did not provide tools to describe the multi-layer properties of these networks. In this chapter we will extend the syntax by providing the multi-layer and technology schemata. So far, we only applied the model to a simplified model of STS, which did not contain inverse multiplexing. In this chapter, we also examine the extend to which we can apply the same model to other technologies. Our explicit goal is to do so without sacrificing the technology independence of the model.

We do so by mapping the functional elements defined in chapter 4 into a syntax.

6.1.1 Scope

In this chapter, we will test the applicability of our model and apply it to as many technologies as possible, including at least all technologies that are in use in the GLIF community:

- Ethernet VLANs
- SONET and SDH
- WDM with wavelength selective switches

- Photonic cross connects

Other technologies we examine are: MPLS, ATM, PPP, Ethernet Q-in-Q [s4], IP, VPN tunnel, wireless, and fibre bundles in a trunk.

6.1.2 Technology Independence

In section 2.3 we have seen that technologies (and thus incompatibilities) change over time. Thus **what needs to be described changes over time**.

Any good path finding algorithm is network independent; it should not need to be modified to take the specifics of a particular network into account. Similarly, **a multi-layer path finding algorithms should be layer independent**; it should not need to be modified to take the specifics of a particular technology into account.

If a path finding algorithm is technology-specific, that means it needs to be updated as new network technologies come along. In chapter 3 we claimed that technologies and incompatibilities change over time. We have also seen that multi-layer incompatibilities can not be resolved locally, but that it crosses

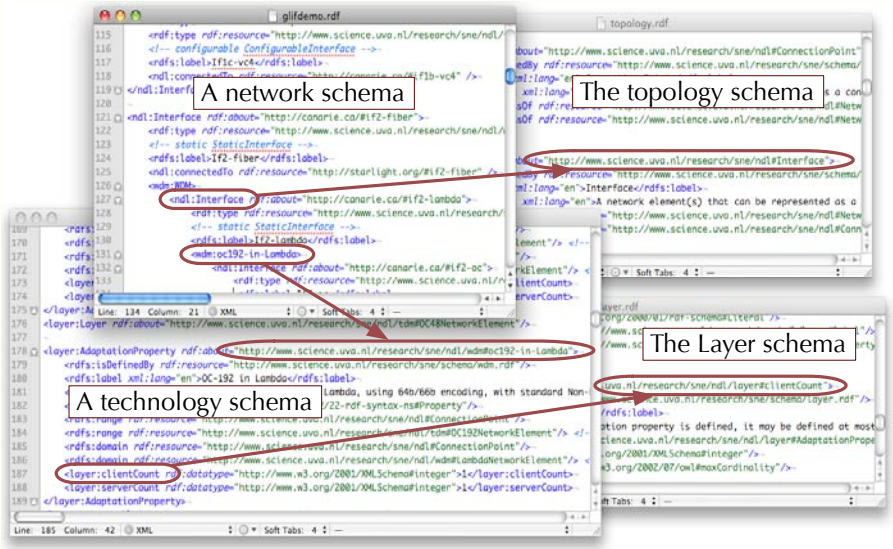


Figure 6.1: *A network description relies on the topology and specific technology schemata. The technology schemata rely on the layer schema.*

multiple domains. This infers that if a new technology comes along, all domains need to update their path finding algorithms. This is not realistic.

Graphs and the Network Description Language are network-independent. They provide simple building blocks to describe networks. Similarly, we want our multi-layer network description to be able to be layer independent by providing building blocks to describe technologies.

This is a clear de-coupling of topology and technology information.

Figure 6.1 shows our implementation of the decoupling between topology and technologies. A network description creates instances of classes defined in the topology schema and in one or more technologies schemata (such as Ethernet, WDM or TDM). The technology schemata are defined as subclasses or instances of classes defined in the layer schema. A path finding algorithm should only have knowledge of the topology schema and layer schema, and learn about a specific network or about specific technologies by reading specific descriptions based on these schemata. With that information, it can find a path.

6.2 NDL Schemata

The implementation of the model is done in RDF. This is a natural extension of the Network Description Language (NDL) presented in chapter 5, which also uses RDF.

The new NDL classes and properties are organised in five modular schemata:

Topology schema that describes the concept of devices, interfaces and connections between them on a single layer;

Layer schema that describes the concept of network layers, and the relation between network layers;

Capability schema that describes device capabilities, rather than just the current state of devices;

Domain schema that describes grouping of network elements in operational domains, describes services, and allows an abstracted view of the network in a domain;

Physical schema that describe locations and physical properties of network elements.

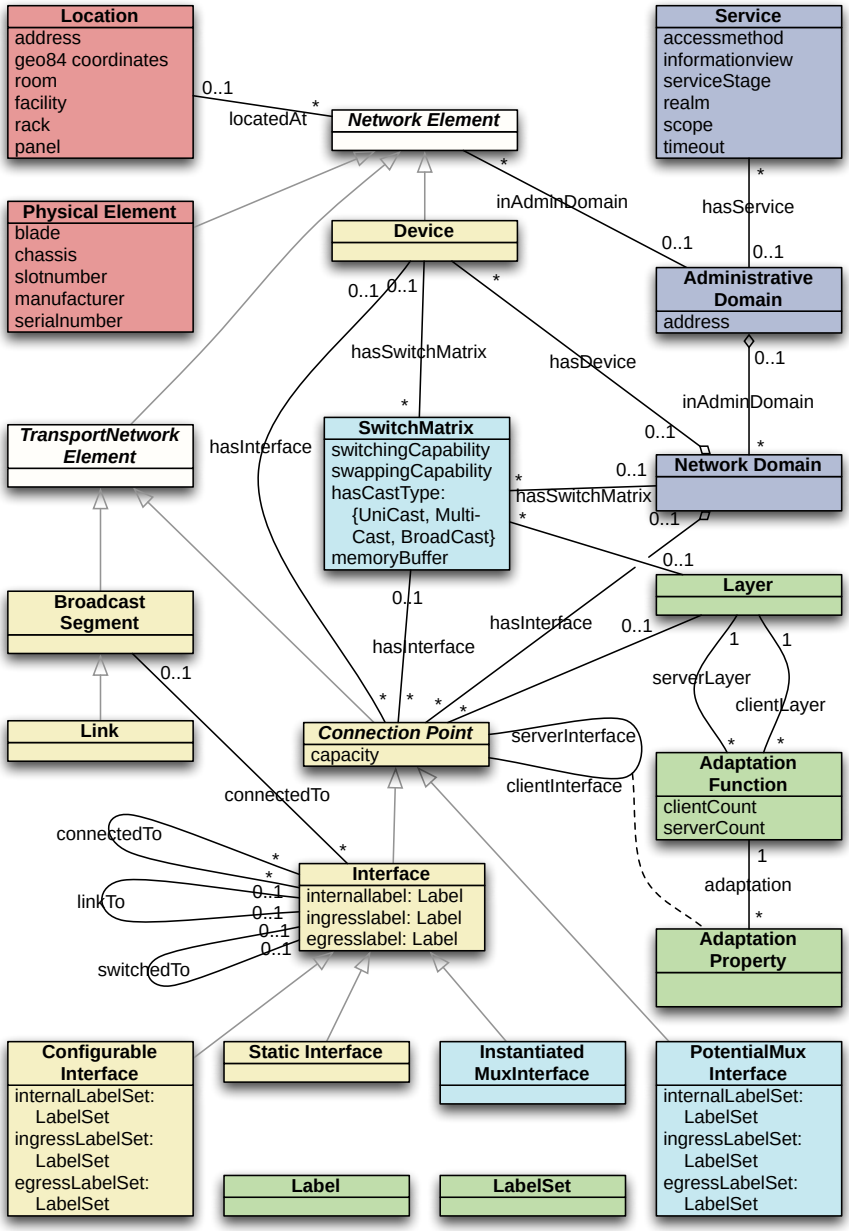


Figure 6.2: *UML representation of the NDL schemas*

The precise schemata are defined in an RDF schema, published on their designated URI [u3]. Figure 6.2 shows a representation of the most important classes and properties of the schemata in the Unified Modelling Language (UML).

In this figure, yellow classes appear in the topology schema, green classes appear in the layer schema, purple classes appear in the domain schema, cyan classes appear in the capability schema and red classes appear in the physical schema[1].

The physical schema is not relevant for path finding, and we skip its details.

6.2.1 NDL Topology and Domain Schema

We already discussed the topology and domain schemata in section 5.3 of the previous chapter. It suffices to mention that we can formally describe the `linkTo` and `connectedTo` properties in ITU-T G.805 terms as unidirectional link connection and unidirectional tandem connection.

Each `Interface` as defined in the topology schema is a connection point; a *logical interface*, not a physical interface. Each channel on each layer is –by default– represented as a single logical interface.

6.2.2 NDL Layer Schema

The layer schema defines the concepts of `Layer`, `Adaptation`, `Label` and `LabelSet`. Our current implementation is to model Layers as a generic class. For example, the Ethernet Layer is modelled as an RDF class *EthernetNetworkElement*. which is of type *Class*, as well as a subclass of *Layer*. The alternative would be to make layers a class instance, and define an RDF predicate *onLayer*. The advantage of our approach is that we can use the domain and range of a predicate to define the client and server layers of adaptations (see below).

Figure 6.3 shows the classes and properties in this schema. The layer schema does not define actual adaptation functions, but instead provides a common vocabulary to describe technologies, layers and the relation between layers.

A *Layer* is a specific encoding, or set of compatible encodings in network connection.

Most *Layers* have an associated *Label Set* are used to distinguish between different channels of a multiplexing adaptation function. For example, the label on the Ethernet layer is a VLAN, and the labelset is the set on integers $0 \ldots 4095$.

[1] For historic reasons, the Location class appears in the topology schema, but semantically, it belongs to the physical schema, and we classify it as such in this thesis.

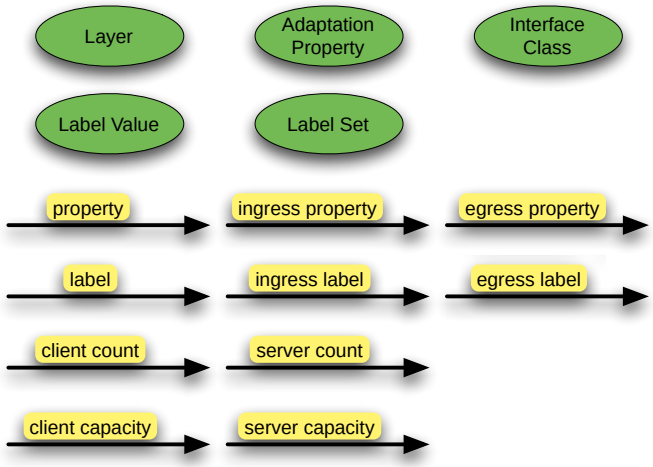


Figure 6.3: *Classes and predicates in the NDL layer schema.*

Each *Adaptation* describes a specific adaptation function, and has four properties, besides its identifier: the client layer, the server layer, the client count and the server count. The client (layer) and server (layer) refer to the *Layers* before and after the *Adaptation*. The client count represents the maximum number of client layer interfaces. The server count represents the number of required server layer interfaces. For 1:1 adaptations, the client count and server count are 1. For multiplexing adaptations, the client count is greater than 1, and the server count is 1. For inverse multiplexing adaptations, the client count is 1, and the server count is greater than 1.

The client count of a multiplexing adaptation is usually equal to the size of the label set for the client layer of the adaptation.

6.2.3 NDL Capability Schema

The NDL capability schema defines the concepts of Switch Matrix and Potential Interface.

Figure 6.4 shows the classes and properties in this schema. The capability schema allows a descriptions of the capabilities, rather than the current state of a network device or network domain.

A Switch Matrix represents a subnetwork in G.805 terminology. It represents the switching capability of a device or domain *at a single layer*. If a

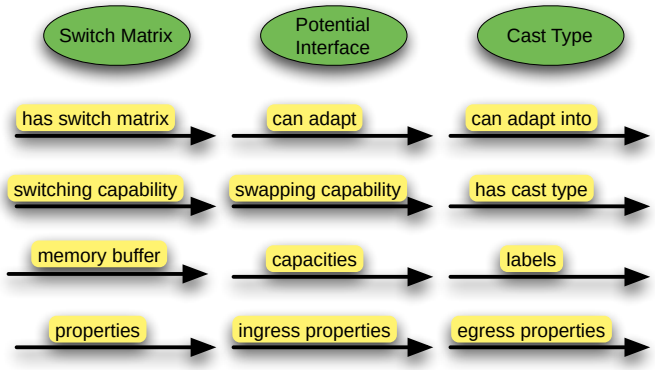


Figure 6.4: *Classes and predicates in the NDL capability schema.*

domain or switch can switch with multiple granularity, it may have multiple switching matrices: one for each layer. In general devices have exactly one switch matrix.

By default, each switch matrix can be configured to forward data from one logical interface to another logical interface. These configurations are represented by a switchTo property in NDL.

The switching capability of a switch matrix is defined by two orthogonal properties:

Switching / swapping capability Some devices are not able to convert between labels. For example, most WDM devices can not convert the wavelength, and most Ethernet switches can not change the VLAN tag. The *switching capability* represents the ability of a device to *forward data from one interface to another interface with the same label*. Two interfaces without a label are considered to have equal labels – both the "empty" label, ϵ. The *swapping capability* represents the ability of a device to *forward data from one interface to another interface with a different label*.

Cast type Most switch matrices can only unicast, meaning that it is possible to make a cross connect from one to another unused interface. Multicast switch matrixes can also make a cross connect from A to B, even if there is already another cross connect with source A. Broadcast switch matrices are entirely different: if two interfaces have the same label, then they must exchange data. Most switch matrices can not only multicast,

but also merge data: there can be multiple source interfaces with the same destination.

The Potential Interface, also defined in the NDL capability schema is an optimisation. NDL defines four different interfaces in total:

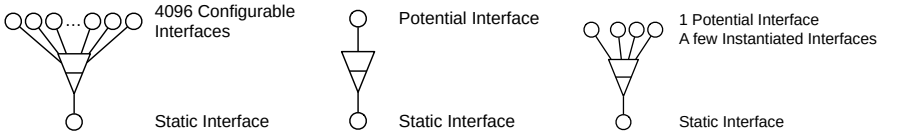
Static Interface is a non-configurable interface. E.g. a laser at a 1310 nm;

Configurable Interface is has not only an actual value, but also a range of possible values. E.g. a tuneable laser;

Potential Mux Interface is an abstract interface, and is semantically equivalent to a set of multiple optional configurable interfaces. For example the set op potential tagged VLAN channels in Ethernet; non-configurable interface. E.g. a laser at a 1310 nm.

Instantiated Mux Interface is an instantiation of a potential interface.

The static and configurable interface are defined in the topology schema. The potential mux and instantiated mux interfaces in the capability schema.



(a) *Ethernet VLANs as 4096 configurable interfaces.* **(b)** *Ethernet VLANs as 1 potential interface.* **(c)** *3 configured VLANs and 4093 potential others.*

Figure 6.5: *Use of a potential interface to describe Ethernet VLANs.*

The potential interface is used to describe that a device can create one or more logical interfaces on the fly. Typically, it is used as a shortcut for multiplexing adaptation functions. Take for example Ethernet VLANs. That would be described as 4096 logical channels in a single fibre. This could be described as 4096 *configurable interfaces* over one *static interface*, as seen in figure 6.5a. While this is fine for a model, it is not efficient for a syntax (a model can be verbose, a syntax should be compact, see section 4.4.3). Instead of saying 4096 times that a channel can be created, it is more efficient to say once that a channel can be created 4096 times. This is described using a *potential interface*, as seen in figure 6.5b. The count of 4096 can be deduced by the number of available labels for the potential interface, as well as the client

count in the adaptation function. Only if certain VLANs are actually in use, then it is required to distinguish between the different configured VLANs. We use an *instantiated interface* instead of a configurable interface to signify that the interface is dynamically created and not permanent.

6.3 Technology Schemata

We proceeded to create a technology schema for each of these technologies: IP, Ethernet, ATM, PPP, MPLS, VPNs, copper, WDM, TDM (SDH/SONET), fibre, fibre bundle, wireless.

All these schemata are *technology specific*. The success of our model is determined by the ability to describe each of these schemata using *only* our technology independent model. If we are successful in doing so, that means that all technologies can be described using a technology independent model, and we can say that our *model* and path finding algorithm are still technology independent.

6.3.1 Encodings

As indicated in section 4.6.5 in the previous chapter, the choice of layers and sublayers for a technology is to some extent arbitrary. Many technologies define more than one possible encoding. An encoding defines the format of data on a link (e.g. the header and the payload, or the framing).

For all practical purposes, we define two different encodings as incompatible, and two equal encodings to be compatible.

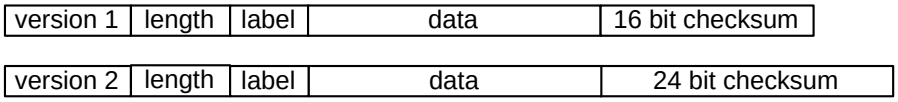


Figure 6.6: *Sample of two encodings for the same layer.*

Figure 6.6 shows a (fictitious) example of two encodings in a single technology: one with a 16-bit checksum, and one with a 24-bit checksum.

Table 6.2 lists a few more examples of layers and encoding types.

We have four options to describe different encodings, and thus to model the possible incompatibilities between network elements:

- Each encoding is modelled as a different Layer;

Layer	Incompatible Encodings
Ethernet	untagged, tagged or Q-in-Q tags
Ethernet	different maximum packet size (MTU)
DWDM spacing	100, 50 or 25 GHz spacing between wavelengths
Ethernet in UTP	10, 100, 1000 or 10000 Mbit/s

Table 6.1: *Examples of encoding types for different layers.*

- Each encoding is modelled as a different adaptation;
- Each encoding is modelled as different labels;
- Each encoding is modelled as a layer property.

Each method has its advantages and disadvantages.

The advantage of different layers, different adaptations or different labels is that the path finding algorithm remains technology-independent. Layer properties are technology specific, since clear rules about compatibilities and incompatibilities need to be defined

The advantage of the different adaptations, labels or the layer property is that it allows the definition of an interface that is not aware of the difference in encoding. For example, while most Ethernet interfaces can extract the packets and the VLAN information, some can not. In chapter 3, we saw an Ethernet in a SONET device, which has no knowledge of the difference between tagged an untagged Ethernet; it simple adapt all data in STS channels, without knowing the exact contents. If we would model tagged and untagged Ethernet as two layers, it would not be possible to properly describe such an interface.

Finally, the advantage of the different layers, different labels and the layer property is that they allow to describe incompatibilities at a single layer. For example, if we want to describe the maximum packet size (MTU) of Ethernet using different adaptations, it would not be possible to describe that (in)compatibility without also describing the layer above Ethernet.

When deciding how to model a certain encoding, we used the following approach:

1. If the two encodings can never appear in conjunction on the same link, model it as two different layers. For example, it is unlikely that there will be a SONET interface which can automatically switch from OC-48 to OC-192, so we modelled OC-48 and OC-192 as two distinct layers.

2. Else, if the compatibility appears in different labels, model it as a label. For example the difference between tagged and untagged Ethernet is the presence or absence of a label.

3. Else, if the incompatibility has many distinct options (such as a MTU from 1518 to 16114 bytes), then we model it as a layer property to avoid an explosion of possible adaptations or possible layers.

4. If two encodings only occur in combination with one or a few other layers, we model it as a different adaptation. For example, we model the different spacings of WDM systems (CWDM, DWDM with 25, 50 or 100 GHz spacing) as distinct adaptations.

5. If all alternatives are exhausted, model it as a (technology-specific) layer property.

In addition to the above, we define layer properties that are not directly associated with different encodings. For example, we define the power level for fibres. While this information is ignored by a technology-independent path finding, it can be used by a technology-specific path finding or fault isolation software. The advantage of RDF is that it can easily be extended with these kind of properties.

6.3.2 Layers and Labels

Table 6.2 lists the layers and label types of the technologies we modelled. The schemata are available from the NDL website [u3]. Note that there is no hierarchy in this list: in our model, all layers are treated equally.

Most of the choices are straightforward, given the method described in the previous section. In particular, MPLS, ATM, and SONET are all straightforward to model.

For SONET and SDH, we relied on the sublayering chosen by GMPLS, as defined in RFC 4606 [s26]. ATM (Asynchronous Transfer Mode) is modelled as four layers, with two VPI (Virtual Path Identifier) layers, since there are two possible VPI labels (8-bit and 12-bits long), and they are never mixed on the same link as far as we are aware.

The next few subsections discusses the technologies where the mapping to our model is not straightforward.

Technology	**(Sub)Layer**	**Label type**
IP	IP	IPv4 address
IP	IP	IPv6 address
MPLS	MPLS	MPLS label
Ethernet	MAC	MAC address
Ethernet	Ethernet	802.1q (VLAN) tags
ATM	AAL0 layer	VCI
ATM	VPI (UNI)	12-bit VPI
ATM	VPI (NNI)	8-bit VPI
ATM	ATM cell layer	*none*
SONET/SDH	VT1.5 / VC-11 layer	M label
SONET/SDH	VT2 / VC-12 layer	M label
SONET	VT3 layer	M label
SONET/SDH	VT6 / VC-2 layer	*none*
SONET/SDH	VTG / TUG-2 layer	L label
SONET/SDH	STS-1 SPE / VC-3 layer	U label
SDH	TUG-3 layer	K label
SONET/SDH	STS-3c SPE / VC-4 layer	*none*
SONET/SDH	STS-3 / AUG-1 layer	stm
SONET/SDH	OC-1 layer	*none*
SONET/SDH	OC-3 layer	*none*
SONET/SDH	OC-12 layer	*none*
SONET/SDH	OC-48 layer	*none*
SONET/SDH	OC-192 layer	*none*
SONET/SDH	OC-768 layer	*none*
SONET/SDH	OC-3092 layer	*none*
WDM	lambda layer	wavelength
WDM	fibre layer	*strand identifier*
UTP/STP	copper layer	*strand identifier*
PPP	PPP layer	*none*
L2TP	L2TP layer	*none*
802.11	802.11 layer	SSID
Fibre bundle	bundle layer	*none*

Table 6.2: *Examples of label types for different layers.*

6.3.3 Wavelength Division Multiplexing

Wavelength Division Multiplexing (WDM) technology is relatively straightforward to model. Using the switching and swapping capability, it is possible to distinguish between a wavelength selective switch (WSS) that can switch individual wavelengths. With the swapping capability it is also possible to describe a device that is able to convert between wavelengths, something which is currently not possible in commercially available devices.

The only difficulties in WDM technologies comes from the continuous (non-discrete) nature of wavelengths, the different spacings between wavelengths and differences in switching granularity. In general, the following WDM systems exist:

- A single wavelength per fibre;
- One wavelength in each optical window: 850 nm, 1310 nm, and 1550 nm;
- One wavelength in each of the O, E, S, C, L and U bands;
- Coarse Wavelength Division Multiplexing (CWDM) as defined in ITU-T G.694.2, with a spacing of 20 nm [s38];
- Dense Wavelength Division Multiplexing (DWDM) as defined in ITU-T G.694.1, with a central frequency of 193.1 THz and a spacing of 100 GHz between different frequencies [s37];
- DWDM as defined in ITU-T G.694.1, with a spacing of 50 GHz between different frequencies;
- DWDM as defined in ITU-T G.694.1, with a spacing of 25 GHz between different frequencies;
- DWDM as defined in ITU-T G.694.1, with a spacing of 12.5 GHz between different frequencies.

Since wavelengths are intrinsic properties of the wavelength layer, they are modelled as different labels. The different spacings on the other hand can either be implied by the different wavelengths (which means that $1552.52 \pm 0.41\ nm$ and $1552.52 \pm 0.21\ nm$ are different labels), by modelling it as layer properties, or as different adaptation functions of a wavelength over a fibre.

Our first attempt modelled this as a layer property, but we later realised we could model it as different adaptations, which would allow us to model different spacings in the technology independent model.

In this model, the adaptation using one spacing and de-adapt in an interface with a different spacing would be considered impossible, while in practice it may work. Also, with only two layers (wavelength and fibre layer), it is not possible to describe a switch that switches with the granularity of a group of wavelengths. Both are reasonable restrictions, and it is possible to alleviate these restrictions by another choice of layers and adaptations, without modifying the underlying model.

A more fundamental problem is the description of available wavelengths. For most multiplexing adaptation functions, the client count is equal to the size of the label set for the client layer of the adaptation. That is not true for WDM, since the label set is continuous (a float representing the wavelength) rather than discrete (such as an integer for VCI, VPI, VLANs or MPLS labels).

One way to solve this is to require the label to be an integer, defined in the context of the adaptation function, rather than the wavelength itself. For DWDM, this would be the integer n in the equation for the wavelength $\lambda = \frac{c}{f_0 + n \cdot f_s}$ with c the speed of light (299792458 m/s), f_0 the central frequency (193.1 THz), and f_s the spacing (12.5, 25, 50 or 100 GHz).

We choose to use the wavelength in nanometre as the label, in correspondence with the proposed choice in GMPLS [s35]. This allowed us to compare wavelengths regardless of their adaptation. The consequence is that our implementation always says there are available wavelengths -floats are continuous- as long as the client count of the adaptation is not set.

Finally, the switching granularity of WDM may not be static. While most devices can switch individual wavelengths, some may switch waveband: a continuous group of wavelength in a frequency range. This can be solved by modelling an additional layer. However, since this is an uncommon technology, we decided to ignore wavebands.

6.3.4 Signal Degeneration

Layer properties such as power levels have been defined in our schema, and are used for fault isolation, the localization of network configuration error [p33]. However, these properties are technology-specific. A technology-independent path finding algorithm can not use them. Our model does not define a logic for signal degeneration, and this is thus not taken into account in our path finding algorithm.

6.3.5 Shared Risk Link Groups

The generic approach of our model allow the description of shared risk links. A shared risk link is a set of fibres that use the same duct. Backup connections should not use two fibres in the same risk group because a digging machine may break both fibres in the same accident. We modelled this as a multiplexing function of multiple fibres adapted in one duct. We use the optional 'strand' label to distinguish between channels (fibres) in the same duct.

6.3.6 Packet Layers

All technologies we discussed so far create bidirectional connections, with the same label in both directions. The SONET timeslot is the same for both directions. This is generally not true for packet switched layers that use lookup tables for routing. For example, the MAC and IP layers use the destination MAC address and destination IP address to find the egress interface in a switch. In our model, we distinguish between the ingress and egress labels (see figure 4.11 in the previous chapter).

The modelling of MAC and IP requires routing tables, and many logical interfaces to describe all connections in terms of circuits. While we are technically able to distinguish between the ingress and egress labels, we found that the description of MAC and IP layers as circuits yields so many logical interfaces that we feel our model is not suitable. While it is certainly possible to extend the RDF schema to describe these kind of switches, it would also require a new logic, which we have not defined yet.

The reason for this limitation lays in the routing table nature of MAC and IP layers, not in their packet switched nature. For example, Ethernet VLANs does not suffer from this scaling issue.

6.3.7 Ethernet

It turns out that it is non-trivial to model Ethernet according to our model for three reasons: (1) Ethernet can be tagged or untagged, (2) Ethernet has different labels (VLAN and I-SID) and (3) Ethernet is a broadcast technology as opposed to a unicast or multicast technology.

As we mentioned in section 6.3.1 above, we had to model Ethernet as a single layer; otherwise we would not be able to describe an Ethernet interface in a STS device. Another advantage of this approach is that we could model tagged Ethernet as 4096 Ethernet channels in Ethernet, and similarly Q-in-Q is modelled as Ethernet in Ethernet in Ethernet.

While the client layers in the Ethernet in Ethernet adaptation has labels, the server layer Ethernet generally does not. There is a distinction that Ethernet makes between labels which is uncommon for other technologies:

Internal labels are used to decide if a cross connect can be made. In case of a switch matrix with switching, but no swapping capability, both ends of the cross connect (the subnetwork connection) must have equal labels. This is the VLAN for both tagged and untagged Ethernet.

External labels are used to distinguish between channels in a multiplexing adaptation function. This is the 802.1Q tag in tagged Ethernet, but is not defined for untagged Ethernet.

So in order to support Ethernet, a distinction must be made between internal and external labels, and untagged Ethernet interfaces must be marked as not using an external label. Alternatively, we can add the logic that a link connection over a 1:1 (non multiplexing) adaptation function does not carry a label, and thus can covert label on the wire. We prefer the explicit notation.

A second complexity is that the Ethernet switch matrix is a broadcast switch matrix. While a regular unicast switch matrix (without swapping capability) **may** make a cross connect between interfaces with same label, a Ethernet switch matrix **must** have a cross connect between interfaces with the same label.

With the addition of this logic, we were able to implement Ethernet VLANs.

6.4 Conclusion

We successfully applied the model of the previous chapter to MPLS, ATM, SONET, SDH, UTP, PPP, WDM and fibre trunk technologies. In addition, with some additional logical on broadcast matrices it was possible to model Ethernet VLANs technologies.

Since the technologies in use within the GLIF community are Ethernet VLANs, SONET, SDH and WDM, we conclude that the model we presented in chapter 4 provides a technology-independent way to describe multi-layer networks.