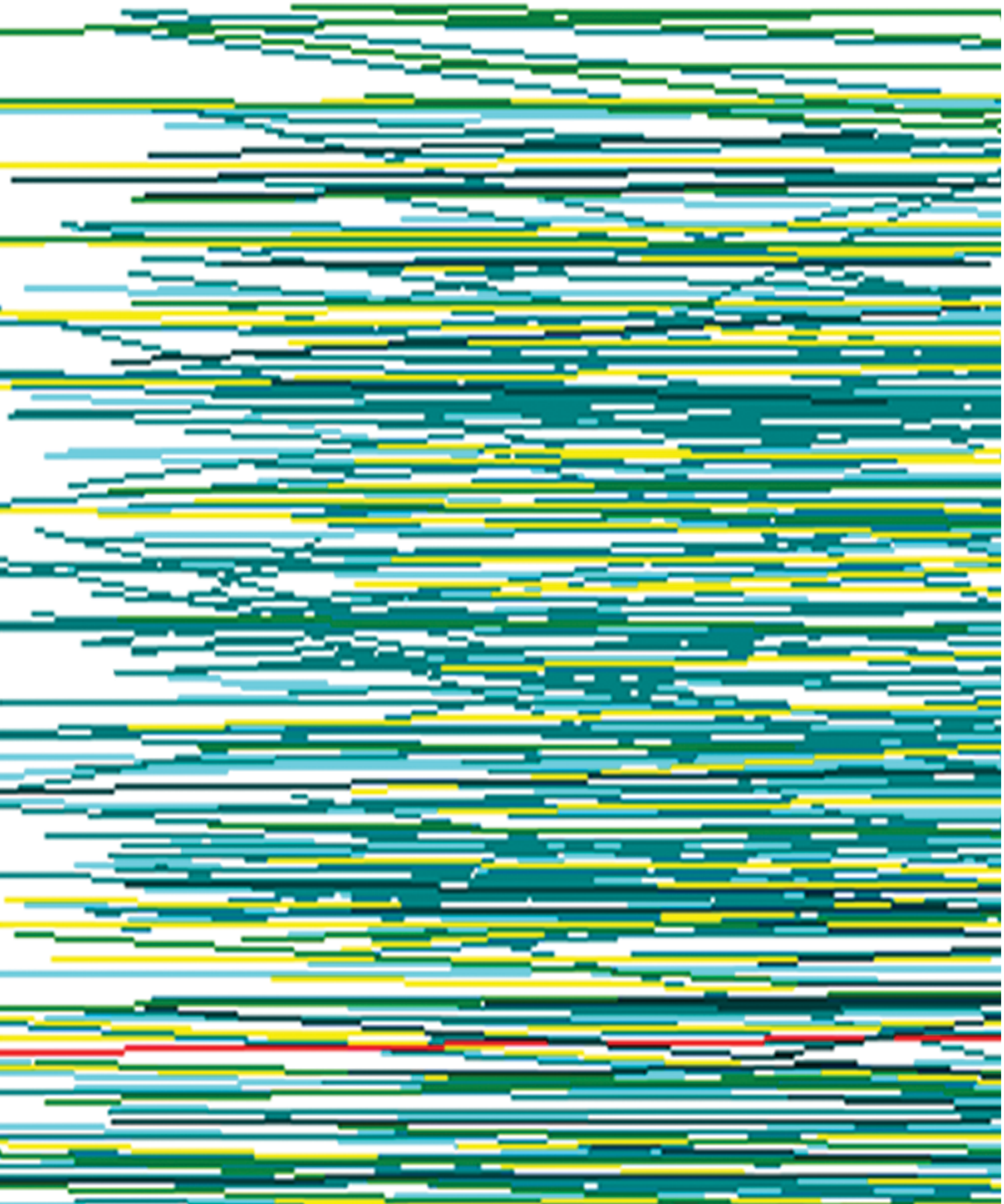
A Semantic Model for Complex Computer Networks
The Network Description Language



Jeroen van der Ham

A Semantic Model for Complex Computer Networks

The Network Description Language

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. D.C. van den Boom
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in de Agnietenkapel
op vrijdag 23 april 2010, te 14:00 uur

door

Jeroen Johannes van der Ham

geboren te Breda.

Promotor:	Prof. dr. P.M.A. Sloot
Co-promotor:	Prof. dr. ir. C.T.A.M. de Laat
Overige Leden:	Prof. dr. P.W. Adriaans
	Prof. dr. ir. H. Bal
	Dr. P. Grosso
	Dr. H. Keus
	Prof. dr. R.J. Meijer
	Prof. dr. D. Simeonidou

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

This work was carried out in the ASCI graduate school.
ASCI dissertation series number 194.
This research was supported by TNO, the Interactive Collaborative Information Systems (ICIS) project, and the GigaPort project led by SURFnet.
The ICIS and GigaPort projects are funded by the Dutch Ministry of Economic Affairs under grant numbers BSIK03024 and BSIK03020 respectively.

Copyright © 2010 by Jeroen van der Ham
This work is licensed under the Creative Commons Attribution 3.0 NL licence.
Cover design by Anneke Joosten © 2010
Typeset by LaTeX. Printed and bound by your printer.
ISBN: 978-90-814160-2-3

Motto

‘Come, we shall have some fun now!’ thought Alice. ‘I’m glad they’ve begun asking riddles. – I believe I can guess that,’ she added aloud.

‘Do you mean that you think you can find out the answer to it?’ said the March Hare.

‘Exactly so,’ said Alice.

‘Then you should say what you mean,’ the March Hare went on.

‘I do’ Alice hastily replied; ‘at least – at least I mean what I say – that’s the same thing you know.’

‘Not the same thing a bit!’ said the Hatter. ‘You might just as well say that “I see what I eat” is the same thing as “I eat what I see”!’

‘You might just as well say,’ added the March Hare, ‘that “I like what I get” is the same thing as “I get what I like”!’

‘You might just as well say,’ added the Dormouse, who seemed to be talking in his sleep, ‘that “I breathe when I sleep” is the same thing as “I sleep when I breathe”!’

‘It *is* the same thing with you,’ said the Hatter, and here the conversation dropped, and the party sat silent for a minute [. . .].

From *Alice’s adventures in Wonderland* by Lewis Carroll.

Contents

Chapter 1

Introduction

1.1 Computer Networks

Communication over computer networks is a very important part of our society today: we make phone calls, send emails, and surf the web. All these processes are enabled by the physical infrastructure of wires, and fibers in the ground, combined with networking devices that communicate electronically over these cables.

There are two different types of network services: packet and circuit switched network services. To describe the difference we can make an analogy to traffic. Packet switched networks are like regular highways that everyone can use, and may encounter traffic jams. Circuit switched networks on the other hand are more like dedicated highways between certain origins and destinations. Cars on these dedicated highways may not drive faster, but it is guaranteed that they do not encounter congestion.

The public switched telephone service is an example of a circuit based switching technology. The Internet on the other hand is largely based on packet switched technology. Both technologies have their merits. Packet switched networks are very robust against failures, but can not guarantee a constant quality of service.

More and more scientific research applications require better quality of services than the regular packet-switched Internet can offer. Such applications may produce so much traffic that if they use the regular Internet, they cause congestion, fail to run smoothly and disrupt other Internet traffic. These applications require dedicated network connections, such as the circuits in the above analogy.

Creating connections in a circuit-switched networks, and especially optical networks, is a complex task. A complete end to end circuit must be created before the connection can be used. Provisioning such a circuit requires knowledge of the network topology, capacities and capabilities.

This thesis describes the Network Description Language (NDL), an ontology for describing complex network topologies and technologies. The language defines a clear terminology for describing network topologies that can be linked to other descriptions, to other kinds of resources, but also to other networks, creating a distributed interoperable description of the global topology.

Network operators tend to be protective of detailed topology information, because of scalability, security, or policy reasons. It is also possible to share an aggregated view of the topology, so that only the most important details are published. In part two of this thesis we examine what kind of effect aggregation has on inter-domain pathfinding.

1.2 e-Science Applications

Scientific research has grown in step with faster computers and more ubiquitous computer networks. In eScience it is now common to see experiments with very large data-sets or requiring high speed or large amounts of bandwidth[8]. As stated above, if these transmissions use the regular Internet, they can disrupt other traffic. Some experiments do not require large bandwidth, but require very controlled jitter and delay on the data transmissions. Below we provide some examples where scientists use dedicated connections in optical networks, *lightpaths*, for their experiments.

For example lightpaths are used by scientists to perform large-scale screening for lung cancer[9]. Large, high resolution radiological images are transmitted to a central repository. An expert can then access the repository and quickly review the new and archived images to come to a diagnosis.

Besides sharing data sets, scientists can also use the network to share specialized equipment such as electron microscopes[10]. With high-speed optical networks such a microscope can be connected to the network, and scientists can remotely control the microscope and view the results real-time.

An extreme example of a specialized instrument is the Large Hadron Collider (LHC) at CERN. Once operating, the experiments in this particle accelerator will generate over 300 GByte/s of data, which is filtered locally to about

300 MByte/s. This data is then immediately globally distributed to 10 Tier 1 institutes through dedicated optical connections. These institutes distribute the data further to Tier 2 institutes. Together these Tier 1 and 2 institutes form the Worldwide LHC Computing Grid[11, 12] where the data of the experiments is stored and analysed.

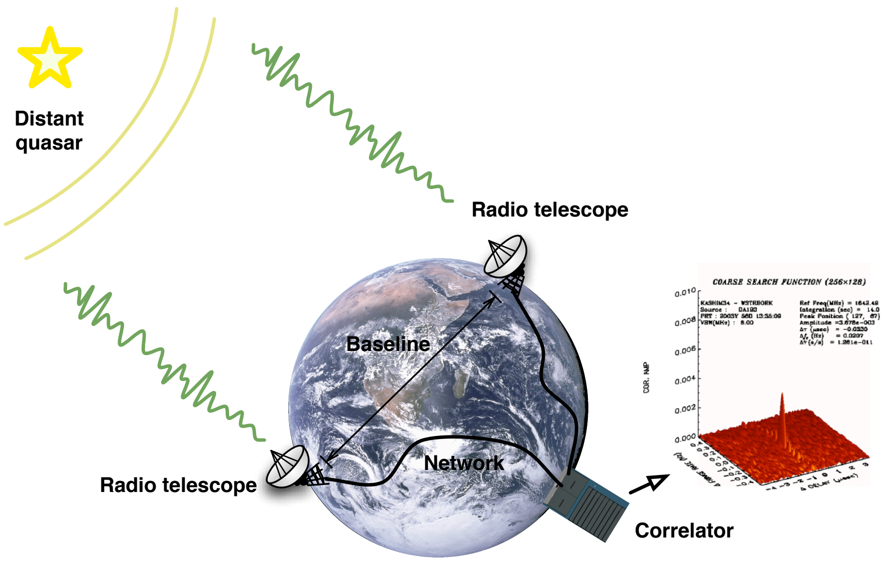


Figure 1.1: *Schematic diagram of e-VLBI (Image courtesy of F. Dijkstra[13])*

Another e-science application that requires dedicated network connections is Very Long Baseline Interferometry (e-VLBI). Two or more radio telescopes, that are far apart, pick up signals from the sky, as shown in picture 1.1. The received data is directly sent to a correlator for processing. The resolution of the correlated signal improves with the distance between the telescopes. Ideally, the telescopes are located on different continents.

Historically, data is shipped on tape from the telescopes to the correlator. Experiments in 2004 have shown that the data can be transmitted over networks [14, 15, 16]. Transmitting this data in real time requires a bandwidth of 1 to 10 Gbit/s. Since the raw measured signal is nearly white noise, it can not be compressed. Typical observation times for telescopes are in the order of several hours. Depending on what astronomical source is being observed, e-VLBI observations use different sets of telescopes.

The StarPlane project[17] provides researchers a dynamic network topology for computing. The StarPlane network connects together the five different clusters of the DAS-3[18] distributed supercomputer. This network is a dedicated part of the SURFnet6 network. The topology interconnecting the clusters can be dynamically changed, optimising the configuration for the demands of the application being run on the distributed supercomputer.

Dynamic high-speed connections also provide a way to quickly migrate virtual machines to other locations[19]. At SuperComputing 2007 it has been shown that virtual machines can quickly be migrated between different sites using lightpaths with minimal downtime of the virtual machine: in the order of 1–2 seconds. This migration can for example be used to migrate the computation to the data set instead of vice versa, or to conserve energy or carbon footprint.

In reality creating such a dedicated network connection is a long process with multiple parties involved. The scientist, his provider network, possible intermediate networks and the destination network all have to agree on the characteristics and details of the connection. The connection is then manually configured by experienced operators in each of the networks. The characteristics and progress on the configuration are then communicated through phone and email. We examine these steps more closely in section 1.6.

The process of creating a network connection can be improved by having a clear and well-defined description of the network. This allows all the parties involved to clearly express their requests and intentions. Having such a language that can also be processed by applications is a first step towards automating this procedure.

1.3 Hybrid Networking

The idea of providing e-science applications with deterministic point-to-point connections was fostered by a community of research networks, later organised in the Global Lambda Integrated Facility (GLIF)[20]. This community provides a global network to support data-intensive scientific research, and also supports middleware development for optical networking. The ideas in this community led to the concept of *hybrid networking*, the offering of packet switched (IP) services and circuit switched connections over the same physical network infrastructure[21].

Since most e-science applications operate in a large scale environment, with collaborators at different universities, the networks required for these applica-

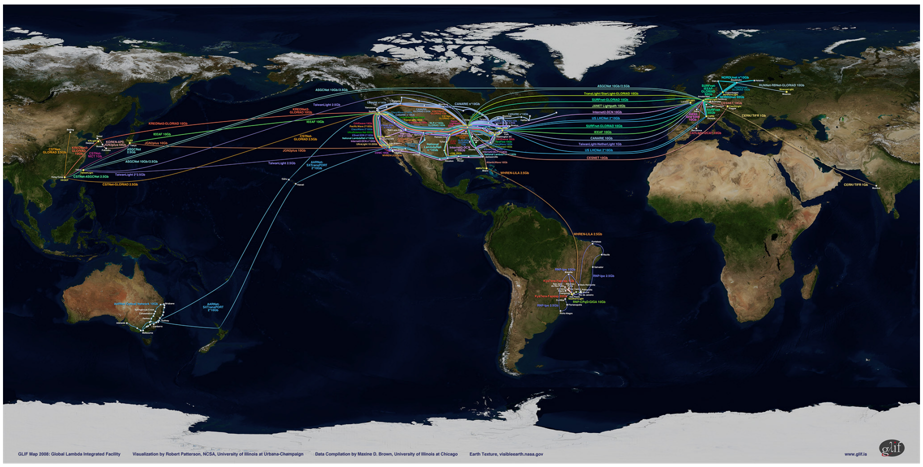

GLIF Map 2008: Global Lambda Integrated Facility
Visualization by Robert Patterson, NCSA, University of Illinois at Urbana-Champaign
Data Compilation by Maxine D. Brown, University of Illinois at Chicago
Earth Texture, visibleearth.nasa.gov
www.glif.is
glif

Figure 1.2: *GLIF world map of May 2008, with all network connections offered by its participants. Source: Patterson, Brown [22].*

tions are nearly always multi-domain networks. De Laat estimated in 2000 that a typical network connection for a physics experiment crosses seven domains [23]. To achieve inter-domain operation, the different networks have to collaborate. For dedicated network connections, this collaboration is done in the GLIF community. In few years time a number of international network connections have been established to provide the inter-domain connectivity. Figure 1.2 shows a collection of the interconnections provided by partners in the GLIF community as of May 2008.

The GLIF community is working hard at improving the lightpath provisioning process by exchanging experiences, documenting processes and developing middleware.

1.4 Military Networks

A large part of the research described in this thesis was performed at TNO, the Dutch Research Laboratory[24], where we also observed a similar complexity in the management of military networks. The military is moving towards more and heavier use of computer networks through network enabled capabilities (NEC)[25]. This way of operating intends to enhance military effect through

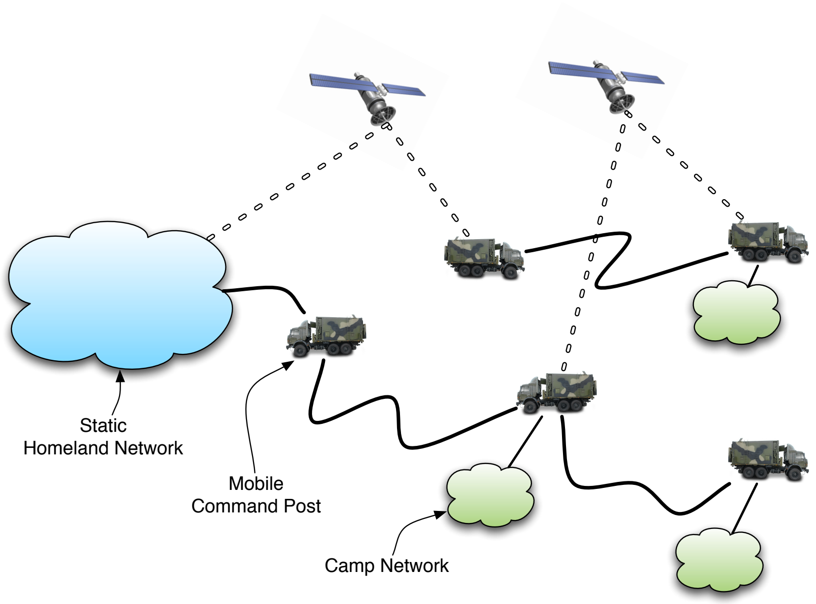


Figure 1.3: *The abstract architecture of the TITAAN network*

better use of information systems. The data from these information systems must be distributed in order to create a shared awareness among the participants in the operations. The goal is to have the right information at the right place at the right time.

The communication network is a key component in achieving the goal of true network enabled capabilities. It is important that all the actors involved have a clear understanding of the capabilities and state of the network. The network can then be used as optimally as possible, given the current situation and possibilities.

An example of such a military network is TITAAN[26], the Theatre Independent Tactical Army & Airforce Network. It is a robust network, configured in such a way that it can be packed up and deployed quickly, anywhere in the world. The network can then be used to exchange data, email, telephone and video-conferencing.

Figure 1.3 shows a high-level abstract view of the TITAAN network. On the left side is the fixed tactical network in the Netherlands. In the middle and right there are mobile command-posts, or bases. Each has a local network. These networks are connected either through radio, satellite, or fixed links.

Currently the TITAAN network operates independently, but the global trend is that military operations are multi-national operations. Combined with the trend towards more network enabled capabilities, we see that nations are trying to create tighter couplings between their networks. In order to make full use of these networks specific information about the network topologies and its properties must be exchanged.

1.5 Management of Computer Networks

Before we present the research overview, we first describe the current architecture for management of computer networks. Figure 1.4 shows a schematic overview of this architecture.

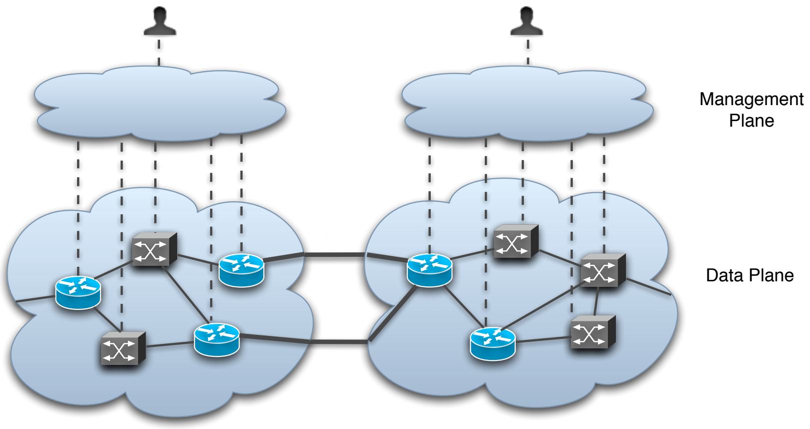


Figure 1.4: *The management plane (top) and the data plane (bottom).*

At the bottom is the *data plane*. This is the physical network over which the signals are sent using data communication protocols, electric or optical pulses. The data plane takes care of moving the data from source to destination. Examples of data plane implementations are Ethernet, TCP/IP, or fiber infrastructure.

The *control plane* is used by the network to manage the topology of the data plane. The routers and switches in a network communicate over the control plane with routing and switching protocols in order to get an overview of the data plane topology. The control plane is not shown in the figure, as it can either be implemented as a small dedicated part of the data plane (in-band), or as an independent network (out-of-band). Examples of control plane protocols are the Spanning Tree protocol (STP) and the Open Shortest Path First (OSPF) protocol, which we will discuss in more detail later.

Finally there is the *management plane*, which is used by the network engineers and operators to manage and monitor the network. The management plane can use the control plane network, or a separate network. Examples of management plane protocols are Simple Network Management Protocol (SNMP) and NetConf. Vendors of networking equipment often also have separate management software which is used to monitor and manage the network. Examples of these applications are HP Openview[27] or Nortels DRAC[28].

Note that the management planes of different networks are shown separated in the figure above. Detailed topology or management data of different networks is not shared between them. Some limited information exchange between networks happen so far as it is required for the operation of the network, for example the announcement of prefixes and connectivity to other Autonomous Systems in case of the Border Gateway Protocol (BGP)[29].

1.6 Research Overview

The goal of our research is to provide a first step towards automatic pathfinding and provisioning of inter-domain lightpaths. Figure 1.5 shows the steps that currently need to be taken to establish a network connection for any e-science application, in this example between a radio telescope and a correlator. If we examine this procedure in more detail, we see that it is broken up in the following underlying steps:

1. The user formulates the requirements, including the end points and the network characteristics like bandwidth, latency, jitter, minimum packet size (if applicable), reliability, etc.

2. These requirements must be communicated to their upstream network provider, in our case the national research and education network (NREN).

   The network provider must gather information about available resources, including the resources in other networks, as the two end-points are typically in different networks.

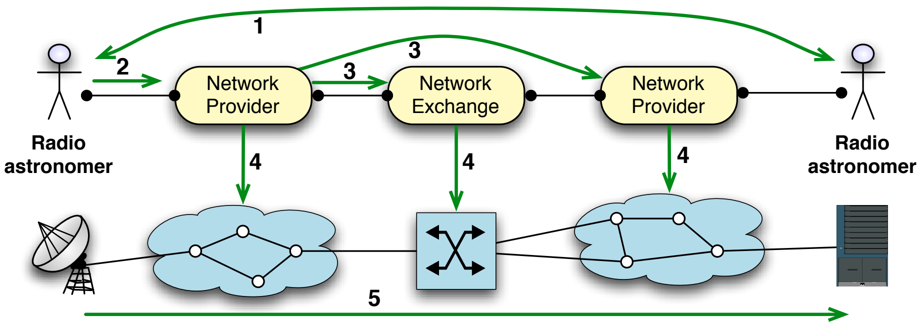


Figure 1.5: *Steps required to set up a network connection between a radio-telescope and correlator.*

3. The network provider must, in collaboration with the other network providers, determine a valid path that uses available resources, and is within the specs of the user.

   The resources needed for the path must be reserved in all networks involved.

4. Once the reservations are all confirmed, the reserved resources must be configured in the networks.

   The end-to-end path must be tested, and in case of faults the faults must be examined and resolved.

   The network provider informs the user, and the user must configure the end nodes (e.g. configure the IP addresses and set the routing table).

5. The user runs the applications.

Currently, this whole process of acquiring a (working) lightpath across multiple domains can take several weeks, a lot of emails and phone calls and extensive testing. It is clear that the whole process needs to be improved and automated in order to scale.

The first step is where the user formulates his requirements. Users can have very different use-cases for lightpaths. For example, a user who wants to do live video streaming has very different requirements than a user who wants to transfer a very large data set. Often users do not know which basic settings they need, or are not able to communicate these clearly to the operator.

Sobieski and Lehman have proposed 'Common Service Definitions' [30], a set of common services with associated values for common parameters for lightpath performance. These definitions make it easier for users to pick the right values. The requirements are also clear to the network operators, allowing them to provide a measurable and reproducible service.

Fault detection and isolation are processes that often occur when operators are trying to bring up a new lightpath. Pure optical sections are hard for fault isolation, it is easy to detect that no light is coming through, but very hard to isolate the section where the light is stopped. Of course faults can also occur during normal operation. Good communication between domains is required for being able to detect and isolate problems. The GLIF community is currently working on this problem[31] by introducing inter-domain lightpath identifiers to facilitate communication between domains[32].

One of our students has studied the problem of fault-detection and isolation in optical networks[33]. This study provides an analyses of the problem and proposes an expert system that can help users and administrators to detect and isolate problems.

If we examine the intermediate steps taken by the network operators, 2–4 in Figure 1.5, then it becomes clear that this involves a lot of communication between the operators. In order to determine a path, they have to exchange topology and capability information. Once a path has been determined an operator must communicate the specifics to the other operators involved. This process is hampered by the lack of an interoperable way of describing and exchanging network topologies.

The first research question of this thesis is the following: *Is it possible to create a distributed information model for the description of topologies and technologies for inter-domain pathfinding?*

When answering the above research question, I assume complete openness about network topologies. The goal is to provide an information model that is as complete as possible.

However, network operators do not always want to provide a full description of their topology, for example because of scalability, or security reasons. A solution for this is to apply topology aggregation, publishing only an aggregated view of the network topology of a domain. However, leaving out information means that pathfinding will not always find the optimal path.

It should be noted that in some cases aggregation of graphs can be reversed, by using directed queries. The original graph can then be inferred from the aggregated topology and the responses. However this is hard to do because the original graph changes due to provisioned connections, and the management system of the network could also detect this behavior and either stop answering or provide misleading answers.

When requesting a path based on an aggregated topology, it is also possible that a path is found in the aggregated graph, which turns out not to be available in the actual network. With that in mind, in the second part of this thesis, I answer a second research question: *What impact does topology aggregation have on inter-domain pathfinding?*

1.6.1 Thesis Outline

The rest of this thesis is structured as follows.

Chapter 2 reviews the basic ideas of network information and data models. We examine the current state of the art, and discuss related research.

Part I describes the Network Description Language:

- **Chapter 3** presents our information model, the Network Description Language, and the reasoning behind the design decisions that we have made to come to this model.
- **Chapter 4** validates our model by implementing components in the context of real life networks. It further discusses the strengths and limitations of our approach.

Part II examines the aggregation of network topology, and the impact of aggregation on pathfinding:

- **Chapter 5** presents different ways of aggregating network topologies, and discusses related work.
- **Chapter 6** describes the experiments that we performed to determine the impact of aggregation on pathfinding.

Chapter 7 summarizes the overall research work presented in this thesis, and answers to the research questions.

Part I

The Network Description Language

Chapter 2

Describing Computer Networks

2.1 Introduction

In the previous chapter we have shown that a main problem for requesting lightpaths or managing complex networks is the lack of topology descriptions. In this chapter we provide an overview of the current approaches to describing computer networks. It is important to note that any description requires two kinds of models[34]: information models that describe resources at a conceptual layer, and data models that describe protocol and implementation details. Obviously there is a relation between data models and information models, as a data model implements an information model. A relation can also exist the other way; if a certain data model is preferred, then this can impose limitations on how the information model is expressed.

The rest of this chapter is organised as follows: in section 2.2 we first look at the requirements of network descriptions in light of our research question. With these requirements, we review existing information models for computer networks in section 2.3. We show that none of the existing information models provide a good solution to our problem. In section 2.4 we examine other approaches that are used in routing protocols.

Before moving onto chapter 3 where we introduce our own information model, we look at possible data models, and provide a brief introduction to Resource Description Framework and the Semantic Web in section 2.5. We summarize and conclude our overview of information and data models in section 2.6.

2.2 Requirements for a Network Model

In the previous chapter we have shown that there are several steps involved in getting a lightpath. The whole provisioning process goes from submitting a clear request to a network provider, distributing the request to other networks, to testing and delivering the lightpath. Currently automating the process of requesting and creating a lightpath is hampered by the lack of network topology descriptions.

A model for network topology descriptions for a global optical network must fulfil the following requirements:

Concise The network model must provide a clear description of the network topology, and all the nodes involved. The definitions of elements must be clear to all the actors involved.

Interoperable There are different vendors for control plane software, which are mostly incompatible. The network descriptions must be in such a form that they can easily be parsed by applications.

Distributed The global network consists of a large number of domains, each managing its resources. Each domain must be able to maintain their own descriptions, and information should stay with its owner. The descriptions should somehow link to each other to form a description of the global network.

Portable Network descriptions will be exchanged between different domains, which means that the model must make use of portable identifiers, which are suitable for use in a global context.

Extensible The model must allow for easy extension or combination with descriptions of other resources. The networks are only vehicles for transporting data. The true value of the network is through the services that are provided at the edges, such as compute services, instruments, or data storage.

Human Readable Currently the process of provisioning lightpaths is done mostly manually. Because of the many parties involved, there will be a transition period where some domains are still provisioned manually, while

Name	Data Model	Expressivity	Portability	Activity	Human Readable	Standards Body
SNMP	ASN.1	-	-	dormant	-	IETF
NetConf	XML	+	+	active	+	IETF
CIM	XML	++	+	active	+	DMTF
GMPLS	OSPF LSAs	+	-	active	-	IETF
NM-WG	XML	+	+	active	+	OGF
cNIS	SQL	+	-	active	+	–
G.805	-	++	-	low	+	ITU-T

Table 2.1: *Feature comparison of information models, for details and abbreviations see section 2.3.*

others are automated. The descriptions should be human readable or tools should be provided to easily visualize them.

2.3 Information Models

In this section we provide an overview of current information models that are in use for describing network topologies. Table 2.1 shows different information models, along with their data models. The table also scores different features of the information and data models. However, not all requirements given in the previous section are easily quantifiable, so we discuss each of the information models in more detail below.

SNMP The Simple Network Management Protocol[1][35] is a set of standards describing a protocol, a database schema, and data objects. The whole suite was originally created as a way of both monitoring and managing network resources. In current networks it is mostly used only for monitoring purposes. Diagnostic, performance and configuration information of network devices can be retrieved from the Management Information Base (MIB) of devices. The MIB is a tree

[1]Technically, the information model is formed by the MIBs, Management Information Bases, and SNMP denotes the whole set: protocol, information and data model.

of name, value pairs, which can be requested and changed. A large part of the MIB is standardised, but vendors also have their own part of the tree. This vendor space is used to store most configuration and performance data of their devices in a proprietary format. Almost all networking devices support SNMP, with different levels of detail in their MIB.

The network description provided by SNMP is distributed over the devices. Depending on the layer the device is operating on, it may have a pointer (address or identifier) to its neighbours on that layer. A view of the whole topology can be created by combining the information gathered from all the devices.

NetConf The Internet Engineering Task Force (IETF) is currently working to replace SNMP with a new standard, NetConf[36]. While SNMP uses its own protocol and only allows for three data-types (integer, string, sequence) , NetConf uses XML, allowing for any data type. Netconf defines a way of transporting monitoring data and change requests over standard protocols, SSH, BEEP or SOAP. NetConf is aimed at distributing diagnostic, performance and configuration information, but also managing devices. NetConf is currently being introduced in networking devices.

As NetConf follows similar principles as SNMP, the network description provided by NetConf is similarly distributed over the managed devices. Each device will have information about the neighbour it connects to on the layer it operates on. The network topology can be created by combining the information of the devices in the network.

CIM Another network device information model is the Common Information Model, CIM[37]. It is being developed by the Distributed Management Task Force (DMTF)[38] and it is an object-oriented information model described using the Unified Modelling Language. The model captures information regarding computer systems, operating systems, networks and other related diagnostic information. CIM is a very broad and complex model, the current UML schemata of the network model is almost 40 pages, the total model is over 200 pages.

DMTF has defined a mapping from CIM to XML, which is mainly used in Web-Based Enterprise Management (WBEM). WBEM is mainly implemented in enterprise-oriented computing equipment, and operating systems such as Windows and Solaris. The high expressiveness in CIM combined with the very active development is to us a negative property. There have been many significant changes in the infrastructure part of CIM over the past two years.

GMPLS GMPLS, Generalized Multi-Protocol Label Switching[39], is a protocol suite developed by the IETF for the provisioning and management of label-switched paths through multi-technology networks. It provides a unified control and management plane for the management of multi-layer networks. Networking devices use the Open Shortest Path First Traffic Engineering (OSPF-TE) protocol to exchange topology data with their neighbours. Devices broadcast the received topology data to their other neighbours, so that in the end all the devices in the domain have the same view of the network topology. The IETF is currently trying to extend GMPLS towards the inter-domain environment.

The topology data in OSPF-TE is exchanged in Link State Announcements (LSAs) packets. The topology data contained therein is encoded in a compact byte format, using specifically defined header fields and Type-Length-Value (TLV) containers. This format is aimed at being easy to process and store for participating network devices, but it is hard to export to external applications.

NM-WG The Network Measurements Working Group (NM-WG) of the OGF has also developed a way of representing network topologies[40, 41]. The NM-WG schema can describe network topologies as used in network measurements. They have created an XML schema, used in several network measurement and assessment tools. Descriptions in this format are for example used by the performance measurement tool PerfSONAR[42]. This tool can collect information and parse output from other tools, and expresses the resulting information in XML. This allows for easy exchange of topology data with other tools.

cNIS cNIS[43] has been developed in the GÉANT2[44] project as a way to store topology information of a single domain. This information is collected and stored by different components such as the perfSONAR measurement tool, the AutoBAHN[45] provisioning tool and others. The model is defined in UML and the tables of the database. The cNIS server can provide the data in different formats towards the different tools. cNIS has a very detailed information model allowing it to describe single domain topologies on different layers. This model has also been brought into the NML-WG. The current plans are that once the NML-WG produces a standard model cNIS will move towards a unified interface for all network tools using that model.

G.805 G.805 is an information model defined by the ITU-T to describe end-to-end connections through a multi-technology network. It is a model to describe the theoretical foundation of network technologies and relations between them.

As such, G.805 only has a graphical data model. We will come back to the G.805 model in section 3.5.

NML-WG In 2007 the Network Markup Language Working Group (NML-WG)[46] was formed in the OGF. This working group has taken the topology schema of NM-WG, the Network Description Language and other schemata as input to define a new schema specifically for describing multi-layer network topologies in a broader application area.

2.3.1 Comparing Information Models

If we examine the requirements that we stated in section 2.2, and compare this with the available information models, then we have to conclude that none of the information models fulfil all requirements. The first three information models summarised above are aimed at describing diagnostics information. NetConf and CIM are also aimed at the configuration management. These three information models are therefore aimed to inform the direct operators of those machines, and the topology part of the models are aimed at that context. The models only describe neighbour information, which makes it hard to describe a complete network topology with it.

The GMPLS information model is aimed at a very different public, namely the switches and routers themselves. The data model is aimed at compactness and is therefore not easy for other applications to understand, nor is it human readable. Unlike the first three information models, GMPLS is aimed at purely describing network topologies, so we take the information model of GMPLS into account, see also section 3.4.

The NM-WG and cNIS information models are aimed at both human and computer consumption. The goal of the models is to publish and share network measurement data along with topology data of those measurements. These measurements are stored in a database so that historical performance data is preserved. The main aim of the NM-WG data model is to relate it to the performance data. So it is not directly suited to create distributed domain descriptions. Nonetheless, the NM-WG model is a very good fit to our problem and is one of the important base models in the new NML-WG.

2.4 Topology Descriptions in Routing Protocols

In the previous section we have examined protocols and information models for describing computer networks. There are also other approaches to routing traffic and circuits in networks which do not require a complete overview of the network topology. For example routing in the backbone of the Internet is managed using the Border Gateway Protocol (BGP)[29, 47], and call routing in circuit switched telephone networks is handled with SS7 (Signalling System #7)[47].

BGP is the routing protocol used in the core routers of the Internet. Networks attached to these routers are identified using Autonomous System (AS) numbers. The exchanged topology information is defined with AS-paths for IP network prefixes. Reachability information that is published in BGP often does not travel through the whole network. A specific path can be aggregated together with other prefixes, saving space in the routing table. Another option is that prefix announcements are filtered out, when the same prefix can be reached through another way, or because of policy reasons. The limited topology information is very suitable for packet-switched networks, but not directly applicable to circuit-switched networks.

There has been an effort to attempt to combine optical networking with BGP called Optical BGP (OBGP)[48]. It uses virtual routers to exchange reachability information on behalf of optical cross connects (OXC). There are some difficulties with these approaches, most users of current lightpaths do not use public IP space, they are often on separate networks, which are temporarily connected to other resources. This makes it hard to announce the connectivity with prefixes. Another difficulty is that the technology used in the optical connectivity is relevant for the whole path construction, and it is not possible to encode this information in the OBGP announcements.

A completely different approach can be seen in telephony networks. Historically these circuit-switched networks have worked without or with very limited topology information. SS7 is the routing protocol currently used in telephony networks. Routing is performed in a distributed hierarchical fashion, based on dynamic lookups and default routes. This approach is feasible in telephone networks, because the network is provisioned and configured in such a way that blocking within the network is not very likely to occur. Such an approach is not applicable to optical networks, where blocking can easily occur.

2.5 Data Models

In section 2.2 we have shown that key requirements are interoperability and exchanging data between different parties. This makes it very important that we have a common, interoperable data model. All of the existing models we described in section 2.3, except the NM-WG model, are not suitable for distribution outside domains.

The information models used in routing protocols as described in section 2.4 are suitable for inter-domain distribution. However, these models have been specifically designed for the networks they are used in, and can not be applied without major modifications to optical networks.

The interoperability and suitability for exchange are mostly properties of the data model. Currently the lightpath provisioning is still done mostly by hand, so we also prefer a data model which is human readable or can be easily visualised, to help the transition towards full automation. There are many data models available, ranging from just plain text to complex binary serialized objects.

Most data models currently in use for computer networks and network management are aimed at information on a single layer. They are typically transported either by a dedicated protocol, or adapted to fit into header space of data packets. Needless to say, these data models are not very portable outside their subnets or protocols.

A standard data model that supports human-readability and interoperability is XML. Since a few years however, there is another data model gaining popularity: the Resource Description Framework (RDF). RDF has been developed by the W3C to support the idea of the Semantic Web.

We assume that the reader is familiar with XML[49, 50]. Below we provide a short introduction to the Semantic Web and RDF, before proceeding to compare RDF with XML.

2.5.1 Introduction to the Semantic Web

The World Wide Web has allowed us to publish and share documents and information with other people in the world. However, because the web is so popular and widespread, it has almost become the victim of its own success. Because of the large scale and the abundant availability of data, it becomes very hard to find what we want. Search-engines, such as Google or Yahoo, have come to the rescue and have indexed the data. However, computers still have no common sense, so the search capabilities of the search machines are rather limited. Consider for example the following two sentences:

- A is connected to B.
- There is a connection between A and B.

Even humans can differ in opinion whether these two sentences have the same meaning. So there is no way that a computer without common sense will understand that these two lines mean the same thing. This is where the Semantic Web comes to the aid of computers (and people). The following is an excerpt of the activity statement of the Semantic Web initiative [51]:

> *The goal of the Semantic Web initiative is to create a universal medium for the exchange of data where data can be shared and processed by automated tools as well as by people. For the Web to scale, tomorrow's programs must be able to share and process data even when these programs have been designed totally independently.*

In 2000 the Semantic Web initiative was started by the World Wide Web Consortium (W3C). Since then they have been working on several specifications to publish and share (meta)data, including the Resource Description Framework (RDF) [52] and SPARQL [53]. In the following section we provide a brief introduction to RDF.

2.5.2 Resource Description Framework

In order for two computer programs to communicate there must be a common understanding about the vocabulary being used. Currently most communication by computer programs is defined by protocols. The form of the interaction is fixed, but the meaning of the data being exchanged is not.

Take a web-browser for example: when a user types in a URL, the web-browser starts communicating with the designated server, asking for the resource identified by the URL ('GET'). The server then answers the browser with the designated file. This interaction is strictly defined in RFC 2616[54]. But neither the web-browser nor the server know what kind of data is being exchanged, it could be about the weather, traffic information, etc.

Because in this example the applications do not grasp the meaning of the data being presented, it limits the possibilities to mere presentation. The Semantic Web idea originated as a solution to this problem; it tries to make it easier for computers to understand the meaning of the content they present, so that they can navigate autonomously through this information to find what they are looking for.

The Resource Description Framework has been created to describe things in a meaningful way for computers. It provides a common framework for expressing metadata so that it can be exchanged between applications without loss of meaning.

Information in RDF is expressed as statements. Each statement is a triplet, with the following elements:

Subject The resource being described

Predicate The property of the subject that is described

Object The value of the property for the subject

A set of triplets is called a graph. An object can also be the subject of another triplet, so complex graphs can be created. An example of such a graph is shown in figure 2.1. An example triplet in the graph is '`Thesis creator Jeroen`', with the subject, predicate and object respectively. The graph also contains other triplets providing more information about the object `Jeroen`, such as his name, email address and homepage.

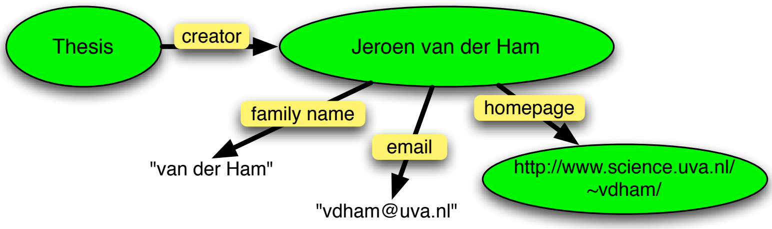


Figure 2.1: *A simple RDF graph*

The graph shown in figure 2.1 still has a problem; we have provided an abstract way of defining relations, but we still use plain English as labels for identifying these relations. Consider for example the `author` relationship, we could also have expressed this as `creator` without much loss of meaning to human readers. RDF solves this terminology problem by using Uniform Resource Identifiers (URIs)[2]. Related terms are usually defined using the same

[2]A URL is one kind of URI. See also [55].

URI-prefix, taking the form of XML namespaces. See for example the Dublin Core Metadata Initiative[56].

There are several ways of expressing RDF graphs, one is the graphical form as in figure 2.1. The most common textual form is *RDF/XML* [57], where the graph is encoded in an XML format. Throughout this thesis we use the RDF/XML notation, which allows us to leverage tools for XML as well as RDF.

2.5.3 RDF Schemata

An example of using semantics with data is the Friend of a Friend project[58]. Participants of this project describe themselves, giving their name, homepage, place of work, etc. The properties are predefined to make sure there is no conflict with e.g. using *'last name'* versus *'surname'*. But definitions of terms is not enough for computers: is *'surname'* the same as *'Surname'*? An example of a FOAF description is given below.

```
<foaf:Person rdf:nodeID="#me">
   <dc:creator rdf:resource="http://www.science.uva.nl/~vdham/thesis/">
   <foaf:family_name>van der Ham</foaf:family_name>
   <foaf:mbox>vdham@uva.nl</foaf:mbox>
   <foaf:homepage rdf:resource="http://www.science.uva.nl/~vdham/"/>
</foaf:Person>
```

Listing 2.1: *The RDF/XML representation of the semantic graph in figure 2.1*

Listing 2.1 describes the semantic graph of figure 2.1 in RDF/XML format. The properties in the example are defined using XML namespaces for readability, they actually point to specific URIs with a well-defined meaning. For example the `creator` property is defined by the URI `http://purl.org/dc/elements/1.1/creator`, which is defined by the the Dublin Core Initiative [59]. When defining RDF properties, it is possible to define what kind of types are valid as subject and object. The set of valid subjects is called the *domain*, and the set of valid objects is called the *range* of that property.

The other terms are either from the `rdf` namespace to describe standard RDF types and objects, or the `foaf` namespace, which provides definitions for the `Person` class, and basic properties of that class. Note that the `homepage` relationship points to a URL, but in this case it is also treated as an RDF object. This homepage object can then have other properties, such as a `creator` property.

An XML namespace with definitions of related terms is called an RDF schema. RDF schemata define the URIs and properties of RDF classes and RDF predicates. RDF classes define the types of subjects and objects.

```
1 <foaf:Person rdf:nodeID="#me">
2    <foaf:knows>
3       <foaf:Person>
4          <foaf:name>Freek Dijkstra</foaf:name>
5          <rdfs:seeAlso rdf:resource="http://staff.science.uva.nl/~fdijkstr/foaf.
               rdf"/>
6       </foaf:Person>
7    </foaf:knows>
8 </foaf:Person>
```

Listing 2.2: *Example of linking descriptions.*

2.5.4 Distributed Repositories

So far we have described how to create local descriptions. However, the Friend of a Friend project as described above is aimed at creating a distributed network of people descriptions. Each person publishes his own description, and links that with descriptions of other people. In RDF the `seeAlso` statement is used to create these pointers. See for example listing 2.2, which extends the earlier example.

Line 1 is the same as in listing 2.1, defining that the description of this person is about me. Line 2 defines that I have a `knows` relation with the `Person` object given in lines 3 to 6. In this case we do not provide an identifier of that object, but rather give a description of it. Line 5 points to another description file which contains more information about this object.

2.5.5 Comparing XML and RDF

Now that we have introduced RDF, we examine some differences between RDF and XML. The most important features in a multi-domain distributed descriptions context are the flexibility of the syntax, how object identifiers are created, how distributed descriptions can be handled, and extensibility.

XML stands for extensible markup language, and ordinary XML syntax is completely extensible, anything is allowed, as long as it is well-formed. This changes dramatically once schemata are defined for documents. These schemata are by definition restrictive.The schema must explicitly define places where extensibility is allowed. If external schemata are used for extensibility, then often an explicit version of that schema is chosen. XML schemata in itself do not contain versioning information. Currently, it is common practice to embed the version number in the name or URL of the schema. If an external schema is used to define some limited extensibility, then to use a new version, the original schema must also be updated.

RDF schemata take a different approach than XML schemata: rather than restrictively defining the structure of a (sub)document, RDF schemata define object classes and properties, and how they can be combined. This means that one document can use many schemata, and also that a single object can have properties defined in different schemata without depending on each other. This also means that when a new version of a certain schema is released, then only the documents need to be updated, since schemata tend not to point directly to other schemata.

The difference in the way schemata are defined also shows when we examine what kind of structures can be described using XML or RDF. In RDF an object is identified using its identifier, which makes it possible to define a flexible graph containing loops. XML on the other hand defines a tree structure, and uses that tree to define relations between objects. This means that it is not straightforward to describe a graph containing loops.

The identification of objects is also a strong feature of RDF. Objects must be identified using a URI, and it is common practice to use the URL of the document as part of that URI. This makes it very easy for authors to ensure global uniqueness of identifiers, which is an important aspect in a multi-domain environment. In XML there is no restriction on what kind of identifiers can be used, and it is not straightforward to create globally unique identifiers. It is also not possible for XML schemata to define restrictions on identifiers, so this must be defined externally.

Creating distributed descriptions is possible with both XML and RDF. In XML it is possible to link documents using XLink[60], or by defining links yourself. On the other hand in RDF the linking between descriptions is a built-in feature using `seeAlso` statements as described earlier. Unfortunately, there is little library support for either XLink or seeAlso, so parsers will have to implement this.

In terms of verbosity, XML has an advantage over the XML syntax of RDF. In our experience, RDF/XML is about twice as verbose as XML. It should be noted that RDF models can also be described using other more compact syntaxes, such as N3. These syntaxes are also supported by almost all RDF tools and libraries.

Another important difference is that RDF imposes some restrictions on the information model. When defining a schema for RDF, that is a mapping of the information model to the data model, a complete *ontology* has to be defined. This means that every element must be given a well-defined meaning. This is both an advantage to the schema author, who is forced to clearly define context and meaning for every single element, as well as for users, who may use the meaning to leverage the information on the semantic web.

2.6 Conclusion

In this chapter we have given an overview of the features of existing, well-known information models for describing computer networks. The first three information models we discussed, SNMP, NetConf and CIM, are aimed at diagnostics and configuration management. The information models of NM-WG, and G.805 are more aimed at topology description, however these models are not intended to publish topology information to other domains. The model of GMPLS is designed for use on the control plane between networking devices, and is not aimed at publishing to other domains, and is also not easily suitable for human consumption. We have to conclude that all these models are unsuitable to publish topology descriptions for inter-domain pathfinding in hybrid networks.

The most common data model is XML, which is suitable for distribution and human-readable. The other data models, for SNMP and OSPF, are specifically designed for use in their respective protocols, and these are not directly suitable for exchange outside of their protocols.

We have also examined another possible data model from the Semantic Web research, RDF. Data model may not be the proper label for RDF since it is more like a layer between the information model and data model. Data is described using triplets forming a graph, and these triplets can be encoded in several different data models, including XML.

In the previous section we have explained several advantages that RDF has over plain XML. We believe these features are important when developing a network schema:

- The use of URIs as generic identifiers is an advantage in multi-domain environments, since it makes it easy to express a request like 'a path from A to B' with A and B clearly and uniquely defined.
- We want to describe the interrelation of different (administrative) network domains. Each domain must be able to publish its own network information and point to other network domains. The seeAlso property makes it possible to easily link between different domain descriptions.
- We want to allow for easy extensibility of the network schema. That is, allow the users to not only publish network information they care about,

but also allow them to mix this with other schemata, both current (e.g. geographic information or organizational information in `geo` and `vcard`), and future schemata, either direct extensions of NDL or non-directly related schemata.

Because RDF/XML is an XML syntax, one could argue that XML can always be used to achieve all of the above strengths by defining an XML syntax that mimics the RDF syntax. However, in that case it is better to use an existing standard, than to develop a custom solution, both for compatibility reasons, as well as being able to leverage existing tools.

Chapter 3

The Network Description Language

This chapter is based on *Using RDF to Describe Networks* by J.J. van der Ham, F. Dijkstra, F. Travostino, H.M.A. Andree and C.T.A.M. de Laat [1] and *Semantics for Hybrid Networks Using the Network Description Language* by J.J. van der Ham, P. Grosso and C.T.A.M. de Laat [2].

3.1 Introduction

In the previous chapter we have examined existing models for describing computer networks. We found that there currently is no suitable model for distribution and pathfinding in a multi-domain scenario. While current information models often use Extensible Markup Language (XML) as portable interchange syntax, we have argued that Resource Description Framework (RDF) is more appropriate to the task.

In this chapter we introduce our information model for topology and network state the Network Description Language. It builds upon RDF and its linking capabilities to produce a distributed view of the global inter-domain network. NDL provides a way to implement the idea of the Topology Knowledge Base as proposed by Travostino in 2005 [61]. To the best of our knowledge NDL is the first ontology in RDF to describe computer networks.

It is worth noting that the proposed network description language is only a method to *describe* topology information. It does not eliminate the need for a control plane for signalling and provisioning.

In section 3.2 we take a closer look at the problem area of describing networks and in section 3.3 we show the initial version of NDL. Then in section 3.4 we improve the model with simple layering information. Finally in section 3.5 we examine the problem of layer descriptions in more detail, and we further improve the model with a technology independent solution for describing multi-layer topologies.

3.2 Terminology for Computer Networks

Terminology plays a very important part in in scientific discussions and research. It must identify the important concepts and objects in the problem space. The terminology to describe computer networks and their operations has been the subject of long debates. In 1978 Shoch[62] attempted to define a terminology in order to help discussions on routing in computer networks:

> "The "name" of a resource indicates *what* we seek, an "address" indicates *where* it is, and a "route" tells us *how to get there*."

Unfortunately his paper was little noticed, and it took until 1982 to get more recognition when Saltzer[63] observed that the discussion on computer networking was still confusing because of lack of strict definitions. He took the terminology of Shoch and provided more strict definitions of these terms, and also introduced the term *network attachment point*, quoting from [63]:

> **Service and Users** These are the functions that one uses, and the clients that use them. Examples of services are one that tells the time of day, one that performs accounting, or one that forwards packets. An example of a client is a particular desktop computer.
>
> **Nodes** These are computers that can run services or user programs. Some nodes are clients of the network, while others help implement the network by running forwarding services. (We will not need to distinguish between these two kinds of nodes.)
>
> **Network attachment points** These are the ports of a network, the places where a node is attached. In many discussions about

data communication networks, the term “address” is an identifier of a network attachment point.

Paths These run between network attachment points, traversing forwarding nodes and communication links.

Unfortunately, the confusion in discussions on computer networks continued to exist as observed in 1999 by Chiappa[64]. He has examined the use of the term ‘address’, and identifies several different meanings and uses of the term. Chiappa concludes that the overloading of the term ‘address’ has lead to confusion and introduces a more strict definition:

> “The name of a network connection entity to which the system of routers will deliver a packet.”

He also introduces the abstract concept of an ‘endpoint’:

> “An ‘endpoint’ is thus defined as one participant of an end-end communication; i.e. the fundamental agent of end-end communication. It is the entity which is performing a reliable communication on an end-end basis.”

The main point Chiappa makes is that the address and name for an endpoint should not be the same. When they are different, it becomes possible to have mobile connections, mobile applications, and on the whole makes for a cleaner solution:

> “Put in more concrete terms, this argument for explicit recognition of endpoints, and naming of them, says that doing so will result in substantial improvements in overall utility, directness, simplicity, robustness, flexibility, etc; these are all properties which are treasured highly in designs that have to have a long life-time.”

The terminology in the above three papers is mostly aimed at describing routing functionality in computer networks. However, the difficulties encountered there show that describing computer networks is not a trivial problem. Even though computer networks are entirely the product of human engineering, its working has become complex and has grown far beyond easy comprehension.

3.3 The Network Description Language

With the discussions and papers as described in the previous section in mind, we have set out to create an ontology for describing topologies of computer networks. Our contribution is to use the outcomes of the papers and discussions not only to write down clear definitions, but to also make this terminology easily available to applications that want to deal with the network.

Our initial goal has been to describe all the necessary elements for doing pathfinding on a single layer in optical computer networks. The most important elements are then devices, interfaces and how these are connected. A path from one device to another goes through interfaces, connections between interfaces, and in the end to another device.

A second use-case is to provide a good overview of resources. For example, our experimentation network is distributed over two sites with a number of connections between them. The network can be seen as a single network, yet to create an accurate description of both our sites, we use location objects.

Our first schema for the Network Description Language is shown in figure 3.1. An ontology in RDF consists of classes and properties, as we discussed in section 2.5.3. Below we describe the classes and properties of NDL in more detail.

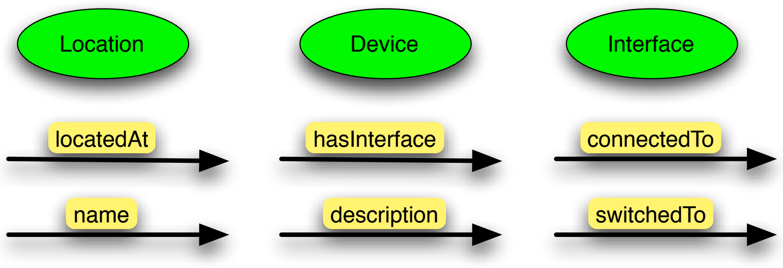


Figure 3.1: *The classes and properties of the Network Description Language (version 1)*

NDL version 1 has three classes, shown at the top of the figure, which define the kind of resources.

`Location` This class describes a place where resources are located. Often requests for lightpaths are from location to location to connect two computing clusters, or between other sets of resources that are specific to that location. A Location class is also helpful when drawing network maps, see also section 4.2.

`Device` The physical nodes in the network, this can be any kind of device such as a computing node, a switch, or a router.

`Interface` The interfaces with which devices are connected to a network.

The figure also shows six properties at the bottom, to define the relations between instances of the NDL classes, other classes, or static values.

`locatedAt` A relation between resources and their location,

`hasInterface` A relation between devices and interfaces,

`connectedTo` A relation between two interfaces, describing that they are directly externally connected,

`description` A relation that can be used to include a (human-readable) description of a resource,

`name` To define the name of a resource,

`switchedTo` A relation between two interfaces describing that they are *internally* connected, for example in optical cross-connects.

The choice for this limited set of classes and properties has been governed by the desire to keep things simple, yet powerful enough to provide accurate descriptions. The language aims to describe the network topology just above the physical layer, so we ignore static physical elements such as filters, or amplifiers. Yet the descriptions provide enough information for our two use-cases, pathfinding and network visualization.

One explicit simplification of the topology has been to describe every element of the network as a generic device. We have also considered differentiating devices by the layer that they operate on, however the actual functionality of networking devices can be very complex. For example, consider current ethernet router/switches. They are capable of both switching and routing, which makes it very hard to explicitly describe their functionality. In the end we concluded that it is important to accurately describe the physical topology. Through the extensibility of RDF, the capability descriptions can always be added later, while retaining full backward compatibility.

A network topology described using these classes and properties contains all elements for pathfinding through the network, if we assume a single technology layer. Even though in optical networks different technologies are used, descriptions in this schema already provide enough information to create network maps, such as the map of the GLIF network shown previously in figure 1.2. These maps give a global overview of the network, and engineers can then get a sense of which domains to contact for connection requests.

An example of a network description is shown in listing 3.1, which describes the network shown in figure 3.2.

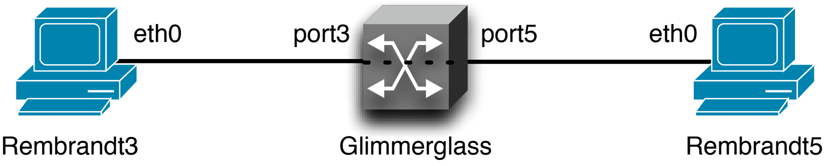


Figure 3.2: *A simple network.*

The example in listing 3.1 starts with the standard header of an XML file using an UTF-8 encoding. Lines 2 and 3 opens the RDF description, and defines namespaces: `rdf` is the standard RDF namespace, and `ndl` is the NDL namespace. The URIs for namespaces in RDF are just identifiers, but often the URI is also used as a URL to publish the schema. In our case we also use `http://www.science.uva.nl/research/sne/ndl#` to publish the NDL schema.

Line 4 opens the definition of the `Lighthouse` location. The `#`-prefix states that the device is defined in the local namespace. Line 5 provides the human-readable name for the location, and line 6 closes the location definition. Lines 7 to 11 define the device `Rembrandt3`. Line 8 provides a human readable name and line 9 states that this device is located in the location `Lighthouse`. Finally, line 10 defines that `Rembrandt3` has an interface, `Rembrandt3:eth0`. This interface is defined on lines 12 to 15. The connection to another interface is defined using the `connectedTo` property on line 14, in this case it is defined to be connected to `Glimmerglass:port3`. The `Glimmerglass` device is defined similarly on lines 16–31, and the `Rembrandt5` device on lines 32–40. The RDF description is then closed on line 41.

The connection between the `Rembrandt3` and the `Glimmerglass` is defined in both directions. This is used to denote a duplex connection and further ensures the consistency of the description.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:ndl="http://www.science.uva.nl/research/sne/ndl#">
   <ndl:Location rdf:about="#Lighthouse">
      <ndl:name>Lighthouse</ndl:name>
   </ndl:Location>
   <ndl:Device rdf:about="#Rembrandt3">
      <ndl:name>Rembrandt3</ndl:name>
      <ndl:locatedAt rdf:resource="#Lighthouse"/>
      <ndl:hasInterface rdf:resource="#Rembrandt3:eth0"/>
   </ndl:Device>
   <ndl:Interface rdf:about="#Rembrandt3:eth0">
      <ndl:name>eth0</ndl:name>
      <ndl:connectedTo rdf:resource="#Glimmerglass:port3"/>
   </ndl:Interface>
   <ndl:Device rdf:about="#Glimmerglass">
      <ndl:name>Glimmerglass</ndl:name>
      <ndl:locatedAt rdf:resource="#Lighthouse"/>
      <ndl:hasInterface rdf:resource="#Glimmerglass:port3"/>
      <ndl:hasInterface rdf:resource="#Glimmerglass:port5"/>
   </ndl:Device>
   <ndl:Interface rdf:about="#Glimmerglass:port3">
      <ndl:name>port3</ndl:name>
      <ndl:connectedTo rdf:resource="#Rembrandt3:eth0"/>
      <ndl:switchedTo rdf:resource="#Glimmerglass:port5"/>
   </ndl:Interface>
   <ndl:Interface rdf:about="#Glimmerglass:port5">
      <ndl:name>port5</ndl:name>
      <ndl:connectedTo rdf:resource="#Rembrandt5:eth0"/>
      <ndl:switchedTo rdf:resource="#Glimmerglass:port3"/>
   </ndl:Interface>
   <ndl:Device rdf:about="#Rembrandt5">
      <ndl:name>Rembrandt5</ndl:name>
      <ndl:locatedAt rdf:resource="#Lighthouse"/>
      <ndl:hasInterface rdf:resource="#Rembrandt5:eth0"/>
   </ndl:Device>
   <ndl:Interface rdf:about="#Rembrandt5:eth0">
      <ndl:name>eth0</ndl:name>
      <ndl:connectedTo rdf:resource="#Glimmerglass:port5"/>
   </ndl:Interface>
</rdf:RDF>
```

Listing 3.1: *The NDL description of the network in figure 3.2.*

Besides a topology description, the file also describes the current configuration of the Glimmerglass device. The `switchedTo` statement in line 25 states that the `Glimmerglass:port3` has an *internal* connection to `Glimmerglass:port5`. Just like the `connectedTo` property, the `switchedTo` property must be defined in both directions. The inverse `switchedTo` property from `Glimmerglass:port5` to `Glimmerglass:port3` is given on line 30. With the `connectedTo` and `switchedTo` statements as given above, we have defined a path from the device `Rembrandt3` to `Rembrandt5`.

3.4 Extending the Network Description Language

The first version of NDL is very basic and only allows to describe the physical topology of the network. While this is useful in itself, we also want to describe more properties of the network, so that we can do more accurate pathfinding in more complex networks. For that reason we have extended the first version of NDL with extra classes and properties [3].

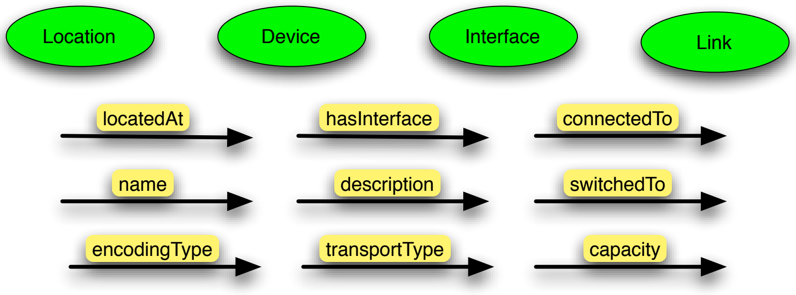


Figure 3.3: *The classes and properties of the Network Description Language (version 2)*

The new schema is shown in figure 3.3. We introduce one new class: `Link`, this class is used to describe connections through a network that operators do not know the exact details of. This is useful to describe trans-oceanic connections; the actual connection is often a leased line provided by a carrier who does not provide a description. There are many trans-oceanic links, sometimes even multiple between the same locations, so it is important to be able to distinguish between them.

On the other hand a `Link` object can also be used to provide a more detailed description of an internal connection, because it allows operators to break up the connection in more detailed parts. An example of this is shown later.

We also add two new properties to allow for the description of layer information to provide support for multi-layer pathfinding: `encodingType` and `transportType`. These two properties can be associated with objects of the Link or Interface class to describe layering information similar to GMPLS[65].

The choice for using the layer descriptions of GMPLS has been governed by the fact that GMPLS provides a very detailed model for describing the encodings and capabilities of devices in multi-layer networks. GMPLS is also the de facto standard for intra-domain circuit provisioning.

Type	GMPLS Value
Packet	1
Ethernet	2
ANSI/ETSI PDH	3
SONET/SDH	5
Digital Wrapper	7
Lambda (photonic)	8
Fiber	9
FiberChannel	11

Table 3.1: *Description of values for the* `encodingType` *property*

Table 3.1 shows the possible values for the `encodingType` property as defined by the Internet Engineering Task Force (IETF). NDL uses the same values as defined for GMPLS. The *lambda* encoding type refers to interfaces that use whole wavelengths, such as wavelength selective switched. The *fiber* encoding refers to interfaces that encode the whole fiber, such as in an optical cross connect.

The values of the `transportType` property define the technology used to map the data on to the encoding. For example, value 18 denotes 'Byte synchronous mapping of DS1/T1', which can be used on `encodingType` 5 (SDH). The values of the `transportType` property are given in table 3.2.

Furthermore we extend NDL with another property for defining the capacity of a `Link` or `Interface` using the `capacity` property. There we also follow the definition of GMPLS and use *bytes per second* as the unit of the capacity, where the values are given using the IEEE floating point format[66].

We follow the definition of capacity as used in GMPLS[65]. That is, the

Value	Type	Technology
0	Unknown	All
5	Asynchronous mapping of E4	SDH
6	Asynchronous mapping of DS3/T3	SDH
7	Asynchronous mapping of E3	SDH
8	Bit synchronous mapping of E3	SDH
9	Byte synchronous mapping of E3	SDH
10	Asynchronous mapping of DS2/T2	SDH
11	Bit synchronous mapping of DS2/T2	SDH
13	Asynchronous mapping of E1	SDH
14	Byte synchronous mapping of E1	SDH
15	Byte synchronous mapping of 31 * DS0	SDH
16	Asynchronous mapping of DS1/T1	SDH
17	Bit synchronous mapping of DS1/T1	SDH
18	Byte synchronous mapping of DS1/T1	SDH
19	VC-11 in VC-12	SDH
22	DS1 SF Asynchronous	SONET
23	DS1 ESF Asynchronous	SONET
24	DS3 M23 Asynchronous	SONET
25	DS3 C-Bit Parity Asynchronous	SONET
26	VT/LOVC	SDH
27	STS SPE/HOVC	SDH
28	POS - No Scrambling, 16 bit CRC	SDH
29	POS - No Scrambling, 32 bit CRC	SDH
30	POS - Scrambling, 16 bit CRC	SDH
31	POS - Scrambling, 32 bit CRC	SDH
32	ATM mapping	SDH
33	Ethernet	SDH, Lambda, Fiber
34	SONET/SDH	Lambda, Fiber
35	Reserved (SONET deprecated)	Lambda, Fiber
36	Digital Wrapper	Lambda, Fiber
37	Lambda	Fiber
38	ANSI/ETSI PDH	SDH
40	Link Access Protocol SDH	SDH
41	FDDI	SDH, Lambda, Fiber
42	DQDB (ETSI ETS 300 216)	SDH
43	FiberChannel-3 (Services)	FiberChannel
44	HDLC	SDH
45	Ethernet V2/DIX (only)	SDH, Lambda, Fiber
46	Ethernet 802.3 (only)	SDH, Lambda, Fiber

Table 3.2: *The possible values of the* `transportType` *property (from [65])*

capacity includes the header space and is in bytes per second. For example, the capacity of a 10 Gigabit Ethernet Link is given as `0x4E9502F9` in IEEE floating point. This translates to $1.25 \cdot 10^9$ bytes per second, which is equal to 10^{10} bits per second.

3.5 The Multi-Layer Network Description Language

In the previous section we have shown a first attempt at describing multiple layers in NDL. The definitions given there are not yet complete, it is only possible to describe the transport layer, and not the switching layer. The definitions used in GMPLS to describe this are quite complex, and even require splitting the description of a device in certain cases[1]. Another disadvantage of using the GMPLS method is that it requires the definition of the technologies in the schema. A new technology means that it cannot be described until there is an updated schema.

While GMPLS is the de facto standard in practice, the ITU-T G.805[68] provides a strong theoretical foundation for describing network technologies and the relations between them. Below we provide a brief introduction to the theoretical foundation of G.805 and show how we have mapped this to NDL. We also extend NDL further with more classes and properties to be able to describe more details of networks. We have split up multi-layer NDL in several schemata, in the following subsections we first explain the new topology schema, then the layer schema, the capability schema and finally the domain schema.

3.5.1 NDL Topology Schema

The multi-layer topology schema is an updated version of the previous NDL schema. We have added several new classes:

`Virtual Device` The resources of physical devices are often split up using virtualization. One physical device can host several virtual devices.

`Static Interface` An interface which is fixed and not configurable.

`Configurable Interface` An interface whose label can be dynamically configured. For example a tuneable laser.

[1]See for example section 4.2.1 in Request For Comments (RFC) 5212[67] which shows how to describe a node that is capable of doing both packet-switching and time-division multiplexing.

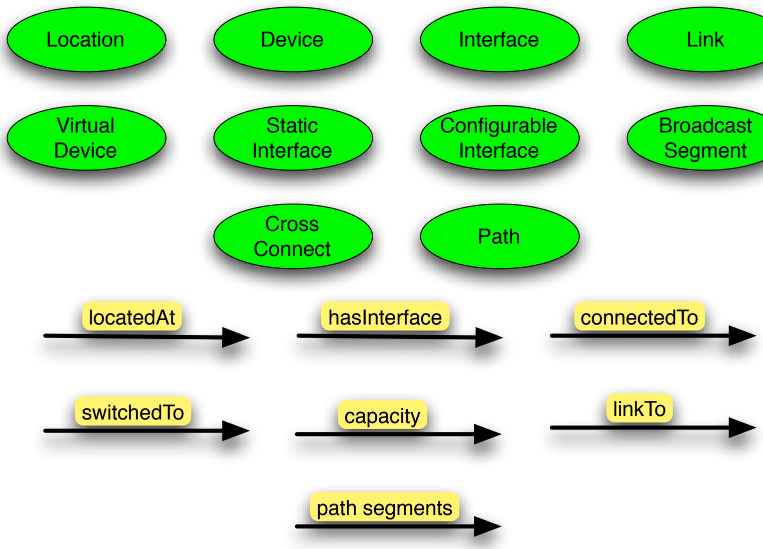


Figure 3.4: *Classes and predicates in the NDL topology schema.*

`Interface` The Interface class is now used to describe a configured interface, i.e. a particular configuration of a `Configurable Interface`.

`Broadcast Segment` A generic case of a Link. A Broadcast Segment is a direct (not concatenated) connection between multiple `Interfaces`. A Link can only connect two Interfaces, while a Broadcast Segment connects multiple interfaces.

`Cross Connect` A collection of network elements that can be represented as a subnetwork connection (ITU-T G.805 terminology). A Cross Connect is an internal data transport within a Device, unlike Links which transport data between two Devices.

`Path` A collection of network elements that can be represented as a tandem connection (ITU-T G.805 terminology) or as a path in a graph (in graph theory). A path is always a connection at a single layer.

We have also introduced three new properties and changed definitions of several others. In the new schema we have introduced the Path concept to describe Paths. Along with this we have also changed the way of describing connections between Interfaces.

A direct connection between two Interfaces (on the same layer) is described either by `linkTo` statements between two Interfaces, or to an intermediate `Link` object. The `linkTo` property is taken to be a uni-directional connection. Similarly, the internal links are described using the `switchedTo` property, either directly or by using an intermediate `Cross Connect` object.

The `connectedTo` property is used to define an external connection between two Interfaces when the details are not all known. Connections can be defined directly or using an intermediate `Path` object.

The `path segments` property is used to describe a Path. The Path object is a container object with a sequence of paths and links.

In this new schema we have removed the `encodingType` and `transportType` properties. We have deprecated these in favour of the ITU-T G.805 approach described with the Layer schema in the next section.

3.5.2 NDL Layer Schema

The ITU-T G.805 document provides an extensive definition of details of adaptations between different technologies. Clients of the network only require knowledge about a subset of these definitions, so below we provide a simplified version. For a more extensive introduction to G.805 and its relation to NDL see [4], [13].

At the core of the definition is an *adaptation function*, this function defines how data is taken from a higher layer, the *client* layer, and adapted into a lower layer, the *server* layer. An adaptation function is always bi-directional, so it also defines how data is *de-adapted* from the server layer to the client layer. Finally, the function defines the adaptation of a specific client layer to a specific server layer, and it is possible to have multiple adaptation functions for the same layer tuple.

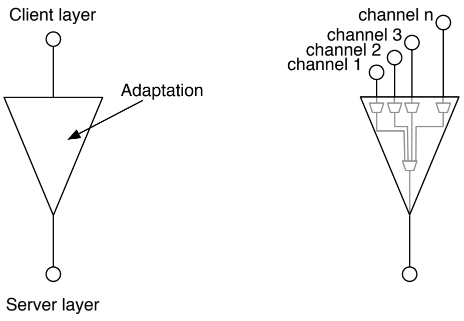


Figure 3.5: *A simplified graphical representation of an adaptation function (left) and a multiplexing function (right)*

Figure 3.5 shows the graphical notation. On the left we show a simple adaptation function. On the right is an example of multiplexing: multiple client layers, *channels*, are adapted together into a single server layer. In order to demultiplex to the original channels *labels* are required, for example VLAN numbers or wavelengths in a WDM signal. The reverse is also possible, i.e. one client layer into multiple server layers, this is called *inverse multiplexing*.

Layers are described in NDL using logical interfaces. This means that a single physical interface is described by multiple `Interface` objects, each on a different layer, depending on the properties of that interface. For example a physical interface in an Ethernet switch will have a logical interface on the physical layer, and on the Ethernet layer, with an adaptation between them. A more extensive example is described below in listing 3.2.

In figure 3.6 we show the NDL layer schema based on the G.805 model. The layer schema does not define actual adaptation functions, but instead provides

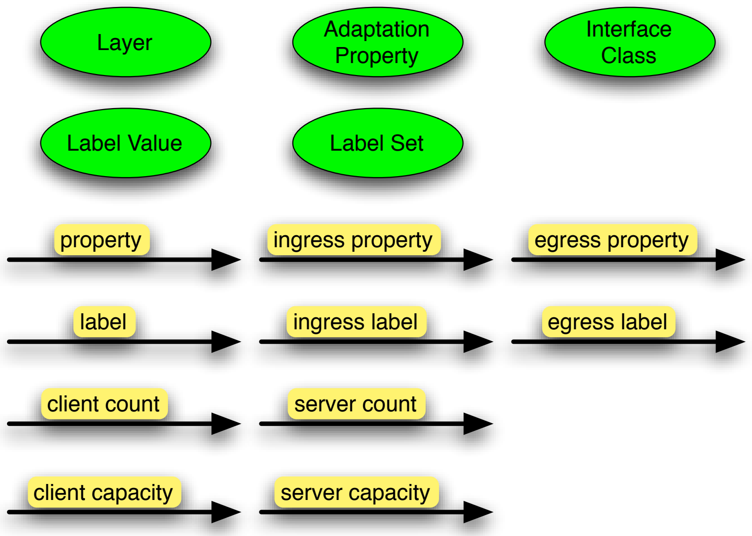


Figure 3.6: *Classes and predicates in the NDL layer schema.*

a common vocabulary to describe technologies, layers and the relation between layers.

Adaptation functions are defined using the class `Adaptation Property`. The definition of that function is given using four properties: the client layer, the server layer, the `client count` and the `server count`.

The client and server layer properties are not explicitly defined as such, instead they are given as the `rdfs:domain` and `rdfs:range` of that specific `AdaptationProperty` instance.

The `client count` represents the maximum number of client layer interfaces. The `server count` represents the number of required server layer interfaces. For one-to-one adaptations, the client count and server count are both one. For multiplexing adaptations, the server count is set to one, and the client count is greater than one, see the example in listing 3.2. For inverse multiplexing adaptations there is a single client layer, transported over multiple server layer connections, for example a 1 Gigabit Ethernet connection that is transported using 21 STS channels. For such an adaptation the client count is 1, and the server count is 21.

A `Layer` is a specific encoding, or a set of compatible encodings. Associated with a layer is a `Label Set`, the set of labels allowed for that layer. For example the label used for the Ethernet Layer is the VLAN, which must come from the set of integers $\{0, 1, 2, \ldots, 4095\}$.

The set of labels that are allowed on an interface are described using the `ingress label set` and `egress label set` properties. The property `label set` is shorthand for setting both ingress and egress to the same value. The actual labels configured on an interface are described using the `ingress label` and `egress label` properties, with a similar `label` shorthand.

Finally we also define `ingress property` and `egress property` for layer specific properties of an Interface. These properties are important to describe, as they may cause incompatibilities between Interfaces on the same layer. For example the MTU size of an Ethernet Interface: normally this is 1500 bytes, but in some cases it is configured to a higher setting.

An example multi-layer description is given in listing 3.2. Lines 1 to 6 start the XML RDF description and define namespaces. Besides the `rdf` and `ndl` namespaces, we also define the RDF Schema (`rdfs`) namespace to use some extra RDF properties, and we include the `layer` and `wdm` schemata. Lines 7 to 12 show an excerpt of that `wdm` schema with the definition of the `WDM` adaptation property. Line 7 defines that it is an `AdaptationProperty` and line 8 defines that it is also a regular RDF property. Lines 9 and 10 define the domain and range of the adaptation property, in this case `FiberNetworkElement` and

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:ndl="http://www.science.uva.nl/research/sne/ndl#"
      xmlns:wdm="http://www.science.uva.nl/research/sne/ndl/wdm#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:layer="http://www.science.uva.nl/research/sne/ndl/layer#">
   <layer:AdaptationProperty rdf:about="http://www.science.uva.nl/research/sne/ndl
        /wdm#WDM">
      <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"
           />
      <rdfs:domain rdf:resource="http://www.science.uva.nl/research/sne/ndl/wdm#
           FiberNetworkElement"/>
      <rdfs:range rdf:resource="http://www.science.uva.nl/research/sne/ndl/wdm#
           LambdaNetworkElement"/>
      <layer:serverCount rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1
           </layer:serverCount>
   </layer:AdaptationProperty>
   <rdf:Property rdf:about="http://www.science.uva.nl/research/sne/ndl/wdm#
        wavelength">
      <rdfs:subPropertyOf rdf:resource="http://www.science.uva.nl/research/sne/ndl
           /layer#label"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
   </rdf:Property>
   <ndl:Interface rdf:about="#port3-l1310">
      <rdf:type rdf:resource="http://www.science.uva.nl/research/sne/ndl/wdm#
           LambdaNetworkElement"/>
      <wdm:wavelength rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1310.0
           </wdm:wavelength>
   </ndl:Interface>
   <ndl:Interface rdf:about="#port3">
      <rdf:type rdf:resource="http://www.science.uva.nl/research/sne/ndl/wdm#
           FiberNetworkElement"/>
      <wdm:WDM rdf:resource="#port3-l1310"/>
   </ndl:Interface>
</rdf:RDF>
```

Listing 3.2: *The NDL description of a WDM adaptation.*

`LambdaNetworkElement` respectively. For this adaptation property we define that the `serverCount` is 1 (an XML Schema integer) on line 11. We do not define the `clientCount`, because this is variable for WDM. WDM is a multiplexing adaptation that is always transported over a single fiber (the server layer). The client count is dependent on the specific implementation and configuration of WDM on a device.

A label property is defined on lines 13 to 16. It is a regular RDF property, as defined on line 13, but also a kind of NDL label, as defined on line 14. The range of the label is an XML Schema float number. This label is used in an interface, `port3-l1310`, that is defined in lines 17–20. Line 17 defines the NDL Interface, and line 18 defines that it is on the Lambda layer. The wavelength of the interface, 1310nm, is defined on line 19. Another interface, `port3`, is defined on lines 21–24, line 21 states that it is an NDL Interface, and line 22 defines that the interface is on the Fiber layer. Everything is tied together on line 23, there we define that the `port3-l1310` interface is adapted to the `port3` using the `WDM` adaptation.

3.5.3 NDL Capability Schema

In the previous section we have defined classes and properties to describe how data is encoded in the network. In this section we define how to describe the capabilities of networking devices, that is how they move data coming in from one interface out to another interface.

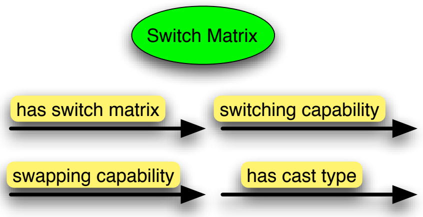


Figure 3.7: *The SwitchMatrix class and its related properties*

A `Switch Matrix` represents the switching capability of a device or domain *at a single layer*. If a domain or switch can operate on multiple layers it may have multiple switching matrices: one for each layer. A switch matrix can be statically configured to forward data from one logical interface to another logical interface. These configurations are represented by a `switchedTo` property in NDL.

The capabilities of a switch matrix are defined using three properties:

Switching capability The *switching capability* describes 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.

Swapping capability Some switch matrices are also able to convert between labels. For example some Ethernet switches can change the VLAN tag. The *swapping capability* represents the ability of a device to forward data from one interface to another interface with a *different* label.

Cast type The cast type defines how a switch matrix can switch different interfaces. Most switch matrices can only do *unicast*, i.e. only cross connect two unused interface.

Some switch matrices can do *multicast*: they can make a cross connect from A to B, even if there already is another cross connect with source A.

Broadcast switch matrices are entirely different: if two interfaces have the same label, then they must exchange data. An example of this is an Ethernet switch matrix that switches on VLAN labels.

3.5.4 Domain Schema

The topology schema allows the description of physical network topologies. The NDL domain schema allows to group these descriptions in networks.

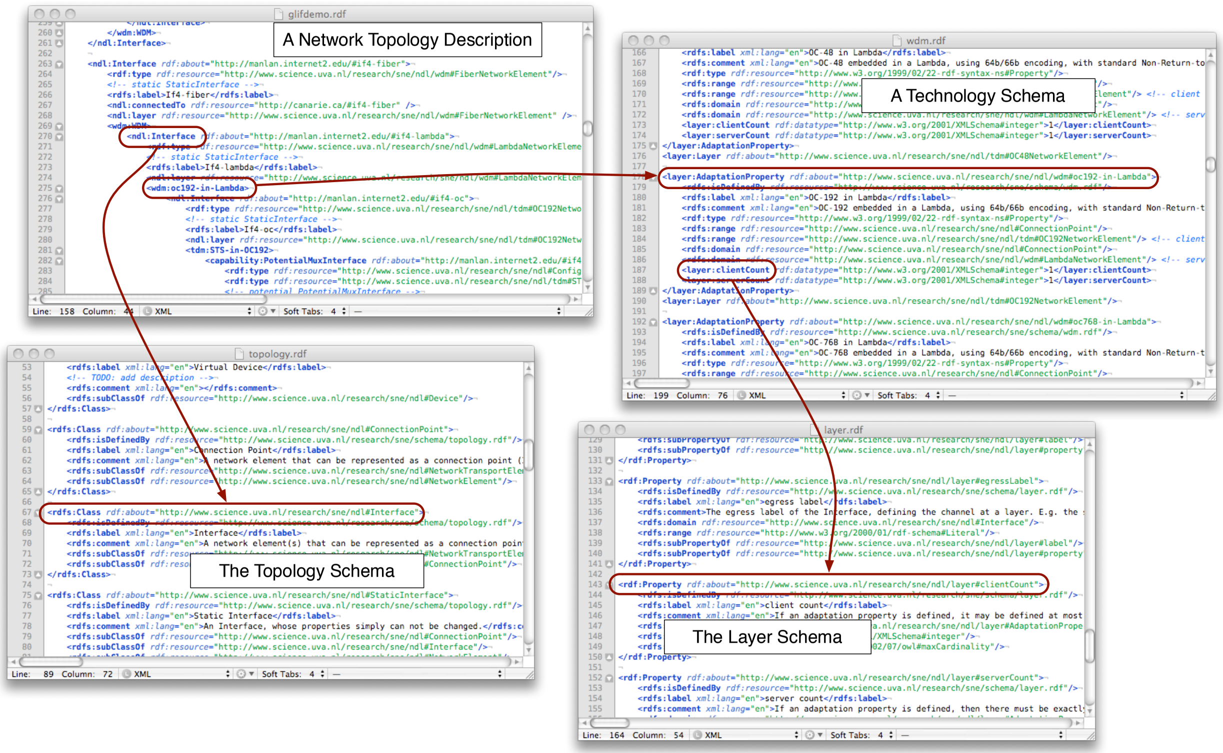
The two classes in the domain schema are:

NetworkDomain is a collection of network elements. It behaves very similar to a Device in the topology schema, but describes a domain rather than a physical device.

AdministrativeDomain is an organizational entity that is responsible for the operational control of resources (including network resources).

3.5.5 Technology Independence

Each network can independently describe their own network in NDL. Similarly, we want our multi-layer network description to be technology independent


glifdemo.rdf
A Network Topology Description
wdm.rdf
A Technology Schema
topology.rdf
The Topology Schema
layer.rdf
The Layer Schema
Line: 158 Column: 44 XML Soft Tabs: 4
Line: 199 Column: 76 XML Soft Tabs: 4
Line: 89 Column: 72 XML Soft Tabs: 4
Line: 164 Column: 54 XML Soft Tabs: 4

```
            </wdm:WDM>
        </ndl:Interface>

        <ndl:Interface rdf:about="http://manlan.internet2.edu/#if4-fiber">
            <rdf:type rdf:resource="http://www.science.uva.nl/research/sne/ndl/wdm#FiberNetworkElement"/>
            <!-- static StaticInterface -->
            <rdfs:label>If4-fiber</rdfs:label>
            <ndl:connectedTo rdf:resource="http://canarie.ca/#if4-fiber" />
            <ndl:layer rdf:resource="http://www.science.uva.nl/research/sne/ndl/wdm#FiberNetworkElement" />
            <wdm:WDM>
                <ndl:Interface rdf:about="http://manlan.internet2.edu/#if4-lambda">
                    <rdf:type rdf:resource="http://www.science.uva.nl/research/sne/ndl/wdm#LambdaNetworkEleme
                    <!-- static StaticInterface -->
                    <rdfs:label>If4-lambda</rdfs:label>
                    <ndl:layer rdf:resource="http://www.science.uva.nl/research/sne/ndl/wdm#LambdaNetworkElem
                    <wdm:oc192-in-Lambda>
                        <ndl:Interface rdf:about="http://manlan.internet2.edu/#if4-oc">
                            <rdf:type rdf:resource="http://www.science.uva.nl/research/sne/ndl/tdm#OC192Netwo
                            <!-- static StaticInterface -->
                            <rdfs:label>If4-oc</rdfs:label>
                            <ndl:layer rdf:resource="http://www.science.uva.nl/research/sne/ndl/tdm#OC192Netw
                            <tdm:STS-in-OC192>
                                <capability:PotentialMuxInterface rdf:about="http://manlan.internet2.edu/#if4
                                    <rdf:type rdf:resource="http://www.science.uva.nl/research/sne/ndl#Config
                                    <rdf:type rdf:resource="http://www.science.uva.nl/research/sne/ndl/tdm#ST
                                    <!-- potential PotentialMuxInterface -->
```

```
    <rdfs:label xml:lang="en">OC-48 in Lambda</rdfs:label>
    <rdfs:comment xml:lang="en">OC-48 embedded in a Lambda, using 64b/66b encoding, with standard Non-Return-to
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:range rdf:resource="http://www.                                        />
    <rdfs:range rdf:resource="http://www.                                    lement"/> <!-- client
    <rdfs:domain rdf:resource="http://www.                                   "/>
    <rdfs:domain rdf:resource="http://www.science.uva.nl/research/sne/ndl/wdm#LambdaNetworkElement"/> <!-- serv
    <layer:clientCount rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</layer:clientCount>
    <layer:serverCount rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</layer:serverCount>
</layer:AdaptationProperty>
<layer:Layer rdf:about="http://www.science.uva.nl/research/sne/ndl/tdm#OC48NetworkElement"/>

<layer:AdaptationProperty rdf:about="http://www.science.uva.nl/research/sne/ndl/wdm#oc192-in-Lambda">
    <rdfs:isDefinedBy rdf:resource="http://www.science.uva.nl/research/sne/schema/wdm.rdf"/>
    <rdfs:label xml:lang="en">OC-192 in Lambda</rdfs:label>
    <rdfs:comment xml:lang="en">OC-192 embedded in a Lambda, using 64b/66b encoding, with standard Non-Return-t
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:range rdf:resource="http://www.science.uva.nl/research/sne/ndl#ConnectionPoint"/>
    <rdfs:range rdf:resource="http://www.science.uva.nl/research/sne/ndl/tdm#OC192NetworkElement"/> <!-- client
    <rdfs:domain rdf:resource="http://www.science.uva.nl/research/sne/ndl#ConnectionPoint"/>
    <rdfs:domain rdf:resource="http://www.science.uva.nl/research/sne/ndl/wdm#LambdaNetworkElement"/> <!-- serv
    <layer:clientCount rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</layer:clientCount>
    <layer:serverCount rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">1</layer:serverCount>
</layer:AdaptationProperty>
<layer:Layer rdf:about="http://www.science.uva.nl/research/sne/ndl/tdm#OC192NetworkElement"/>

<layer:AdaptationProperty rdf:about="http://www.science.uva.nl/research/sne/ndl/wdm#oc768-in-Lambda">
    <rdfs:isDefinedBy rdf:resource="http://www.science.uva.nl/research/sne/schema/wdm.rdf"/>
    <rdfs:label xml:lang="en">OC-768 in Lambda</rdfs:label>
    <rdfs:comment xml:lang="en">OC-768 embedded in a Lambda, using 64b/66b encoding, with standard Non-Return-t
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:range rdf:resource="http://www.science.uva.nl/research/sne/ndl#ConnectionPoint"/>
```

```
    <rdfs:label xml:lang="en">Virtual Device</rdfs:label>
    <!-- TODO: add description -->
    <rdfs:comment xml:lang="en"></rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://www.science.uva.nl/research/sne/ndl#Device"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.science.uva.nl/research/sne/ndl#ConnectionPoint">
    <rdfs:isDefinedBy rdf:resource="http://www.science.uva.nl/research/sne/schema/topology.rdf"/>
    <rdfs:label xml:lang="en">Connection Point</rdfs:label>
    <rdfs:comment xml:lang="en">A network element that can be represented as a connection point (
    <rdfs:subClassOf rdf:resource="http://www.science.uva.nl/research/sne/ndl#NetworkTransportEle
    <rdfs:subClassOf rdf:resource="http://www.science.uva.nl/research/sne/ndl#NetworkElement"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.science.uva.nl/research/sne/ndl#Interface">
    <rdfs:isDefinedBy rdf:resource="http://www.science.uva.nl/research/sne/schema/topology.rdf"/>
    <rdfs:label xml:lang="en">Interface</rdfs:label>
    <rdfs:comment xml:lang="en">A network element(s) that can be represented as a connection poin
    <rdfs:subClassOf rdf:reso                                       #NetworkTransportEle
    <rdfs:subClassOf rdf:reso                                       #ConnectionPoint"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.science.uva.nl/research/sne/ndl#StaticInterface">
    <rdfs:isDefinedBy rdf:resource="http://www.science.uva.nl/research/sne/schema/topology.rdf"/>
    <rdfs:label xml:lang="en">Static Interface</rdfs:label>
    <rdfs:comment xml:lang="en">An Interface, whose properties simply can not be changed.</rdfs:c
    <rdfs:subClassOf rdf:resource="http://www.science.uva.nl/research/sne/ndl#ConnectionPoint"/>
    <rdfs:subClassOf rdf:resource="http://www.science.uva.nl/research/sne/ndl#Interface"/>
```

```
    <rdfs:subPropertyOf rdf:resource="http://www.science.uva.nl/research/sne/ndl/layer#property
</rdf:Property>

<rdf:Property rdf:about="http://www.science.uva.nl/research/sne/ndl/layer#egressLabel">
    <rdfs:isDefinedBy rdf:resource="http://www.science.uva.nl/research/sne/schema/layer.rdf"/>
    <rdfs:label xml:lang="en">egress label</rdfs:label>
    <rdfs:comment>The egress label of the Interface, defining the channel at a layer. E.g. the
    <rdfs:domain rdf:resource="http://www.science.uva.nl/research/sne/ndl#Interface"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
    <rdfs:subPropertyOf rdf:resource="http://www.science.uva.nl/research/sne/ndl/layer#label"/>
    <rdfs:subPropertyOf rdf:resource="http://www.science.uva.nl/research/sne/ndl/layer#property
</rdf:Property>

<rdf:Property rdf:about="http://www.science.uva.nl/research/sne/ndl/layer#clientCount">
                                                                         layer.rdf"/>
    <rdfs:label xml:lang="en">client count</rdfs:label>
    <rdfs:comment xml:lang="en">If an adaptation property is defined, it may be defined at most
    <rdfs                                     a.nl/research/sne/ndl/layer#AdaptationPrope
    <rdfs                                     /XMLSchema#integer"/>
    <rdfs                                     02/07/owl#maxCardinality"/>
</rdf:Property>

<rdf:Property rdf:about="http://www.science.uva.nl/research/sne/ndl/layer#serverCount">
    <rdfs:isDefinedBy rdf:resource="http://www.science.uva.nl/research/sne/schema/layer.rdf"/>
    <rdfs:label xml:lang="en">server count</rdfs:label>
    <rdfs:comment xml:lang="en">If an adaptation property is defined, then there must be exactl
```

Figure 3.8: *A network description relies on the topology and specific technology schemata. The technology schemata rely on the layer schema.*

by providing building blocks to describe technologies. In the multi-layer NDL schemata we have provided classes and properties to describe the details of layers and adaptations. This is a clear de-coupling of topology and technology information.

Figure 3.8 shows our implementation of the de-coupling between topology and technologies. It shows a network topology description at the top-left, which makes use of the topology schema (bottom-left), and a specific technology schema, in this case WDM (top-right). This technology schema is defined using the NDL Layer schema (bottom-right). These descriptions are all tied together using standard RDF definitions.

We have created examples of technology schemata for Ethernet, WDM or TDM and more[69]. 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. This approach allows for a pathfinding application that learns about layers and technologies from the NDL descriptions. This means that the application does not have to know a priori about all the technology details[70].

3.5.6 Comparing NDL and GMPLS

In our previous attempt at extending NDL to allow multi-layer descriptions in section 3.4 we have used concepts from GMPLS with the `encodingType` and `transportType` properties and values. However, we soon realised that this approach requires us to statically define the layers and technologies. With the ITU-T G.805 approach described above, we can describe layers and adaptations independently. The multi-layer topology descriptions can then link to the relevant layer and adaptation details. This allows for a more dynamic and future-proof approach.

However GMPLS is currently the de facto standard in the management of optical networks. Our multi-layer NDL should be able to describe anything that is possible to describe using GMPLS. In order to prove this we have attempted to map the information that can be carried in OSPF(-TE) LSAs to NDL. We have concluded that it is possible to translate all the relevant multi-layer topology information from OSPF and OSPF-TE to NDL. The details of these mappings can be found in appendices A and B respectively.

3.6 Conclusion

In this chapter we have introduced the way we apply RDF to describing networks: the Network Description Language. At first NDL only provided a way to describe the physical topology. This allowed us to create linked multi-domain topologies, which can be very useful for network diagrams, and as aid to engineers in manual pathfinding.

In the next step we have extended NDL towards multi-layer network descriptions. Building on the work of GMPLS, we extended NDL with properties to describe encoding and transport types, as described in section 3.4. However, the descriptions as inspired by GMPLS do not allow us to describe arbitrary technologies. When new technologies are introduced, the schema must be updated to include new values for these technologies.

Believing that there must be a more optimal solution, we turned to the ITU-T G.805 standard. ITU-T G.805 provides a way to generically describe layers and adaptations. Building on these ideas, we defined the NDL Layer schema as shown in figure 3.6. Taken together with the Capability schema, and the Domain schema, we now have a good solution for describing multi-layer multi-domain networks with at least the same expressivity as GMPLS.

A complete overview of the NDL schemata as a UML class diagram is shown in figure 3.9.

In section 2.2 we set requirements for a network model, it should be concise, interoperable, distributed, portable, and extensible. In this chapter we have shown the definitions of the NDL schemata, which provide concise definitions for how to use each class and property. By using RDF as the data model for NDL, it is interoperable, following an open standard. It is also distributed, with the powerful `seeAlso` property, it is possible to link together different, separately maintained network descriptions. RDF also makes it easy to create globally unique identifiers using the automatic prefixing with the `#` notation, this makes network descriptions easily portable. RDF also allows to easily extend descriptions with properties from other schemata. Finally, RDF can be expressed in XML or other human-readable syntaxes. For these reasons NDL satisfies all of our original requirements.

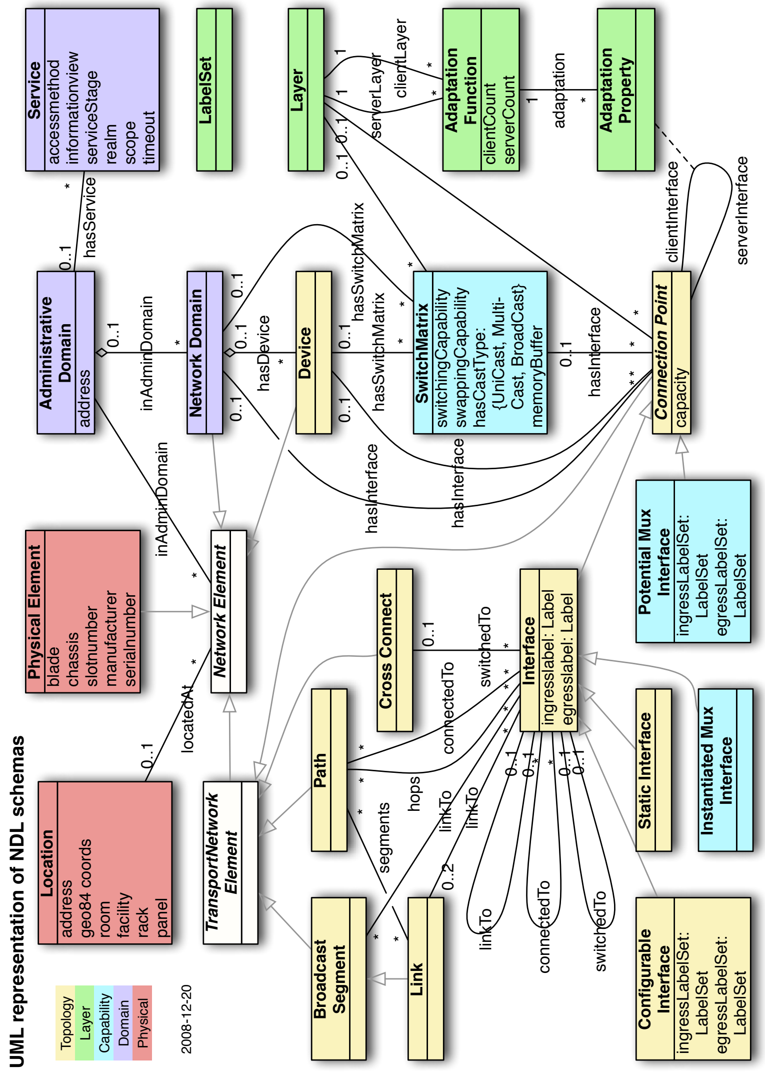


Figure 3.9: *UML representation of the complete NDL schema*

Chapter 4

NDL Applications

This chapter is based on *Semantics for Hybrid Networks Using the Network Description Language* by J.J. van der Ham, P. Grosso and C.T.A.M. de Laat[2], *Using the Network Description Language in Optical Networks* by J.J. van der Ham, P. Grosso, R. van der Pol, A. Toonk and C.T.A.M. de Laat[3] and *A Distributed Topology Information System for Optical Networks Based on the Semantic Web* by J.J. van der Ham, F. Dijkstra, P. Grosso, R. van der Pol, A. Toonk and C.T.A.M. de Laat[5].

4.1 Introduction

In the previous chapter we have introduced the Network Description Language. We have shown an example of how it can describe devices, and link between different domains.

One of the advantages of using NDL as the language for description of hybrid networks is the availability of semantic web tools for RDF that can parse and consume the information in each NDL file. This means that extracting the information needed for network management, and in our specific case lightpath provisioning, is straightforward and simple.

In this chapter we show several examples of how we have applied the language to solve many of the operational issues that operators and users face in hybrid optical networks, and other kinds of networks.

Network operators and users often use maps of the network to make sense of the topology, and to support them in diagnostics or manual pathfinding. Having up to date maps is very important, but creating and updating these maps is also very difficult and labour intensive. In section 4.2 we show our graph generation application.

Before we can start working on the operational issues, we need to gather data, preferably in an automatic way. In section 4.3 we present how we gather data from networks.

Besides extracting the visual aspect of the network description, we can also perform other queries on the data. Section 4.4 shows how we perform queries on the data, and our applications for lightpath planning both in a single network, as well as over multiple networks.

After developing several of these applications we have found that it is helpful to have a toolkit available to support the most common tasks performed with NDL files. In section 4.5 we discuss the Python NDL Toolkit (pynt).

Another application that we have developed is a network emulation toolkit called Virtual Network Experiments (VNE). We discuss this toolkit in section 4.6.

Finally in section 4.7 we summarize our work, and discuss our findings and results with the different applications.

4.2 Network Graph Generation

Our first application of the language has been the visualization of network topologies. Given that most lightpaths are still provisioned manually, at least when they involve crossing organization boundaries, maps become the visual aid used by network engineers to setup the circuits. The information about the connection between domains must be up to date, accurate and consistent, because mistakes in lightpath provisioning can have impact on other lightpaths and in the case of hybrid networks also on the regular traffic.

There are certainly many ways to create a graphical overview of a network, and when working in a single network domain plenty of tools to choose from. But in multi-domain environments, such as the cooperating universities and research institutes in GLIF, we need to take into account that the information for each domain is not centrally maintained and there is a big potential for inconsistencies. The manual creation of these large-scale topology overviews requires a tedious conversion of gathered data to a consistent representation, and the verification of consistent information at the boundaries. After these steps one can then feed the information to the graphing tools. Requesting this data from the different networks in GLIF and processing that is a process which can easily take months.

NDL provides a way for operators to interoperably and independently publish their network topology, including references to other networks. Starting from a network description in NDL format, we extract the connections between the devices and their names. Using a small script, this data is then converted to serve as input to GraphViz, an open source graph visualization tool[71]. An example of such a graph is shown in figure 4.1. This is a map of NetherLight[72], one of the network domains participating in GLIF.

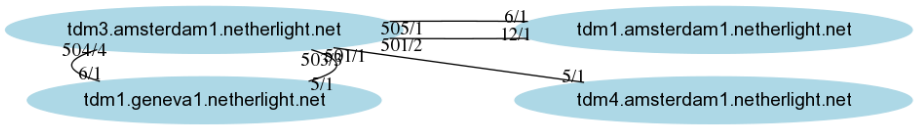

tdm3.amsterdam1.netherlight.net
tdm1.amsterdam1.netherlight.net
tdm1.geneva1.netherlight.net
tdm4.amsterdam1.netherlight.net
505/1
501/2
6/1
12/1
504/4
6/1
503/1
501/1
5/1
5/1

Figure 4.1: *A graph of NetherLight resources (generated from NDL file)*

The graphs generated using GraphViz are not always ideal, they are generated automatically, which means that a small change in the topology can radically change the generated image. For larger networks another option is to embed GPS coordinate information in the NDL documents using the standard `geo` RDF namespace. It then becomes possible to use Google Maps[73], for example, to display networks on the map.

Figure 4.2 shows a subset of the GLIF resources on a global Google Map plotted using NDL information. Each of the pointers in the figure is a separate NDL file. The links between these points are defined by both sides, including `seeAlso` pointers to the other file. The inter-domain topology is created by crawling the distributed descriptions.

4.3 Automatic Generation of Network Descriptions

In the early stages we created network descriptions by hand. This soon became very tedious, and error-prone. A first simple method of automating NDL generation is by using forms. We created a simple webform (see [74]), which allowed us to quickly create a description for a number of locations and devices with their interfaces, and connections between them.

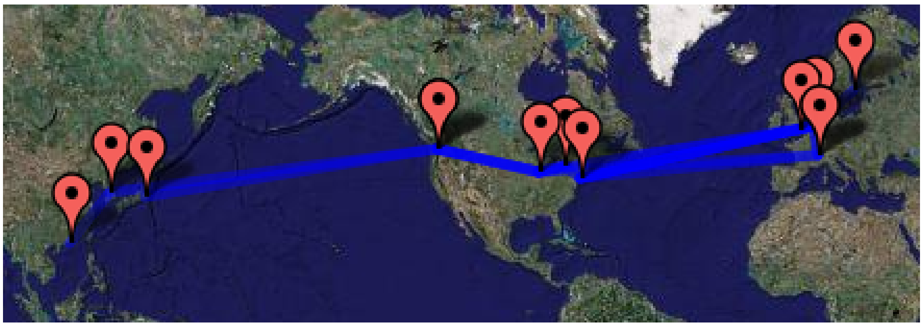


Figure 4.2: *A subset of the GLIF resources on a Google Map generated from NDL files as demonstrated at SC06*

Network descriptions based on real world data are harder to create. Unfortunately, as we have discussed in chapter 2 there is no standard way of representing management data in network hardware. Different vendors use different techniques for storing management information. Over the years we have created several different modules to cope with different sorts of hardware. Examples are hand-coded Python scripts for reading specific devices (Force10 E-Series, Glimmerglass), Python and Perl modules for input through TL1, and a Python module for input from OSPF and OSPF-TE traffic.

When reading information from separate devices, the information must be correlated using external information. That is, the information about the cables connecting devices must be provided in some way. Even in dynamic networks, these cables are often very static, most changes to the network topology are done by changing device configurations. For example in our test network, all machines are connected to an optical patch-panel, and several ports of this patch-panel are connected to a switch. This allows a very dynamic topology where connections between devices can either be made directly only passing through the patch-panel, or through the switch, possibly using VLANs.

In larger networks where OSPF is used to manage routing, it is possible to automatically gather topology data. The routers in such a network use a link-state algorithm. This means that each router monitors the links to its neighbours, and distributes this information to other routers in the network. The distribution is done in such a way that each router has full knowledge of the whole topology in its link state database. We wrote a program which requests a copy of this database, and translates the relevant information to NDL, resulting in an automatically generated description of the network. Unfortunately, OSPF only exchanges information about connectivity at the IP layer, so the topology description will be limited to that layer.

An extension to OSPF, called OSPF Traffic Engineering (OSPF-TE), is used in the GMPLS protocol suite. OSPF-TE defines several new types of messages. These can contain information about layers and different kinds of labelling, so that different transport techniques can be combined. The new messages also allow the topology information to be exchanged out-of-band, i.e. the messages about the network can be exchanged on a separate network. This allows it to be used in optical networks where in-band management is not possible.

4.3.1 Topology Generation for TITAAN

The TITAAN network is military network of the Royal Netherlands Army (see section 1.4), using commercial off-the-shelf hardware. The network nodes have been configured such that it allows for quick plug-and-play operations, requiring minimal configuration in the field. All this has been done (just) within the boundaries of normal networking standards and capabilities.

While the configurations have allowed for quick setup in the field, some of the complexity moved to managing and monitoring the network at the network operations centre. The engineers have had trouble getting a good overview of the network, because all available network management tools can not make sense of the configuration.

We have developed a proof-of-concept application for the TITAAN network which extracts the topology data from the OSPF network. The topology is then exported to NDL, so that it can be used in other applications, such as showing which path will be used between two nodes.

The application has provided the engineers with a valuable source of information with which they can easily see which route traffic will take through the network. The topology data can also easily be used as input information for other applications, so that for example they can gauge the current state of the network and adapt their behaviour to it.

4.3.2 Topology Generation from OSPF-TE

We have also extended our work on topology extraction from OSPF to OSPF-TE. Messages in OSPF-TE are exchanged out-of-band, the control-plane network runs regular OSPF, which is then supplemented with Opaque LSAs. These are a special kind of LSAs that can carry different kinds of contents describing the topology and details of devices and interfaces on the data plane.

Unfortunately, there are not many production networks using GMPLS, but we have successfully tested this with some experimental setups using DRAGON. DRAGON, Dynamic Resource Allocation over GMPLS Optical Networks, is an open-source implementation of OSPF-TE and other GMPLS protocols[75]. We have also successfully used the exporting from OSPF-TE in network emulations, which we discuss in section 4.6.

Technical details about OSPF and OSPF-TE and how the information in LSAs maps to NDL descriptions can be found in Appendices A and B.

4.4 Extracting Data from Network Descriptions

As we described in the previous chapter NDL is based on RDF. This has several advantages, one of which is that we can make use of generic RDF tools and standards. An important tool for RDF is the SPARQL Protocol and Query Language for RDF (SPARQL)[53], which is an SQL-like query language for RDF. It uses a simple syntax to specify variables and triplet templates for retrieving information from a repository. An example is shown in Listing 4.1.

```
PREFIX ndl: <http://www.science.uva.nl/research/sne/ndl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?hostname ?locationname
WHERE { ?host rdf:type ndl:Device .
      ?host ndl:locatedAt ?location .
      ?host rdf:label ?hostname .
      ?location rdf:label ?locationname .
     }
```

Listing 4.1: *Example of a SPARQL query.*

The example shows a query to select hostnames and their location names, the variables `?hostname` and `?locationname`. The values must satisfy the constraints in the `WHERE` clause. These constraints are expressed using two other variables, `?host` and `?location`. The `?host` must be of type `Device`, and must be `locatedAt` a `?location`. Then with the `label` property the names of both objects are found. In summary this query will return all host and location pairs, if the host has a location defined.

4.4.1 Lightpath Planning in SURFnet6

SPARQL allows for much more complex queries, and we have used it in a lightpath planning application for SURFnet6. SURFnet6 is the Dutch national research and education network. SURFnet6 is a hybrid network, offering both IP services and lightpath services. Toonk and Van der Pol of the Dutch national super-computing centre SARA have written a tool for planning new lightpaths based on NDL and SPARQL [76, 77] .

Part of the SURFnet6 hybrid network is formed by a collection of Nortel OME6500 Time-Division Multiplexing (TDM) nodes. We obtain the topology information by using the neighbour knowledge from each of these devices. We gather the data by periodically sending and receiving discovery messages on the control plane. We use NDL to describe the topology of the TDM layer in files.

```
 1 my $query = new RDF::Query ( <<"END", undef, undef, 'sparql' );
 2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
 3 PREFIX ndl: <http://www.science.uva.nl/research/air/ndl#>
 4 SELECT ?device1 ?device2 ?if1 ?if2
 5 WHERE {
 6       ?d1 ndl:hasInterface ?x .
 7       ?d1 ndl:name ?device1 .
 8       ?x ndl:connectedTo ?y .
 9       ?x ndl:name ?if1 .
10       ?d2 ndl:hasInterface ?y .
11       ?d2 ndl:name ?device2 .
12       ?y ndl:name ?if2
13 }
14 END
15
16 $ne1 = $result->[0]->getValue;
17 $ne2 = $result->[1]->getValue;
18 $if1 = $result->[2]->getValue;
19 $if2 = $result->[3]->getValue;
20
21 $g->add_edge("$ne1-$if1", "$ne2-$if2");
22
23 my @V = $g->ShortestPath("$v1", "$v2");
```

Listing 4.2: *A SPARQL Query implemented in Perl*

Listing 4.2 shows the SPARQL query that is used in the planning application. Line 6 gets all interfaces of a device and line 7 gets the name of the device. Line 8 gets the neighbour of an interface and finally line 10-11 get the name of the device to which the neighbour interface belongs.

Line 16-19 store the device and interface names of both ends of a link (connectedTo). These variables are used to build a graph. Line 21 builds the edges of the graph. This implicitly adds the vertices `$v1` and `$v2` to the graph. Finally, in line 23 a shortest path is computed by the method `ShortestPath`, which is a standard Dijkstra Shortest Path implementation.

Additionally, a network state database holds the cross-connect information for each network element in the network, that is, information about currently provisioned lightpaths. This enables the application to determine the amount of time-slots still available on each interface. Combining the NDL topology information and the database time-slot information we can find a shortest path through the network that has enough free time-slots to accommodate a new user request. To find this path we use a constraint based shortest path algorithm.

The result of the path calculations are then implemented by human operators who provision the lightpaths through SURFnet6. This application is a first step towards completely automatic lightpath provisioning. Before creating this application, the engineers had to piece together the time-slot information themselves, making the provisioning process a very time-consuming procedure.

4.4.2 Lightpath Planning in GLIF

We have also extended the lightpath planning application to GLIF, which constitutes an ideal environment to see NDL at work in a multi-domain and multi-administrator setup. The most important domains of the GLIF network are the lightpath exchanges, also called GLIF Open Lightpath Exchanges (GOLEs).

As a proof of concept, we created an abstraction of several GOLEs of the GLIF network, where each GOLE is described as a virtual device with several interfaces. These interfaces connect the GOLE to other GOLEs. To correlate the abstracted descriptions of each individual GOLE with each other we use the `seeAlso` property of RDF. Using these links, a linked web of descriptions is formed. This provides a global view of the network, where each domain maintains the description for its own GOLE. This network is also shown in figure 4.2.

```
<ndl:Device rdf:about="#netherlight">
   <rdf:label>Netherlight</rdf:label>
   <ndl:locatedAt rdf:resource="#NetherLight"/>
   <ndl:hasInterface rdf:resource="#netherlight:if1"/>
   <ndl:hasInterface rdf:resource="#netherlight:if5"/>
   <ndl:hasInterface rdf:resource="#netherlight:if6"/>
   <ndl:hasInterface rdf:resource="#netherlight:if10"/>
</ndl:Device>

```

```
10 <ndl:Interface rdf:about="#netherlight:if1">
11    <rdf:label>if1</rdf:label>
12    <ndl:connectedTo rdf:resource="http://networks.internet2.edu/manlan/manlan.rdf#
          manlan:if1"/>
13    <ndl:capacity rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.2E+9</
          ndl:capacity>
14 </ndl:Interface>
15
16 <ndl:Interface rdf:about="http://networks.internet2.edu/manlan/manlan.rdf#
       manlan:if1">
17    <rdfs:seeAlso rdf:resource="http://networks.internet2.edu/manlan/manlan.rdf"/>
18 </ndl:Interface>
```

Listing 4.3: *Part of the abstracted Netherlight GOLE description*

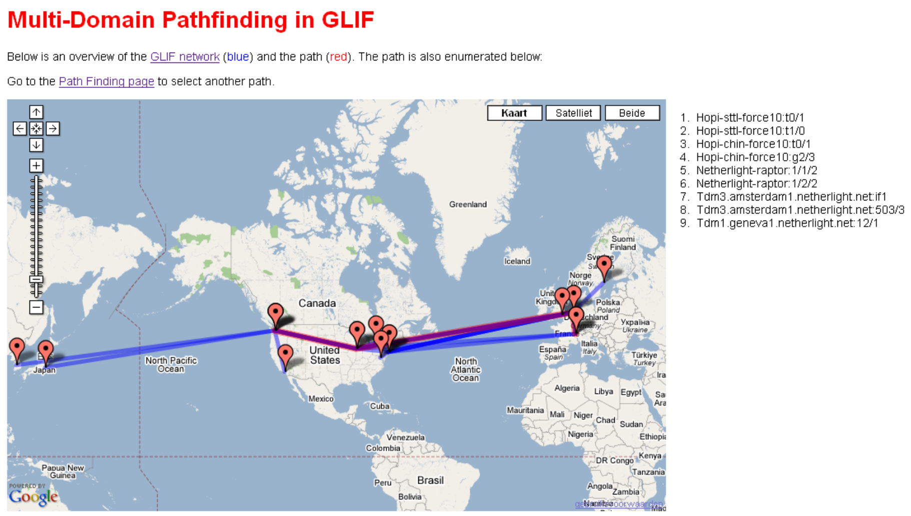
Listing 4.3 shows an excerpt of the abstract description for the Netherlight GOLE. Line 1 describes which virtual device (GOLE) this is. Line 4 to 7 describe which interfaces the (virtual) device has. Interface `netherlight:if1` is described on line 10 to 16. line 11 describes its name and line 12 describes where it is connected to. Finally, line 14 describes the capacity (total bandwidth) of the interface. Lines 18 to 20 provide a pointer for the description of the other side of the connection using the `seeAlso` property.

To find a path within the GLIF the first step is to read this NDL file. The description for the other GOLEs is found by crawling the links contained in the `seeAlso` properties. This way we can determine if a path exists and can be provisioned in the GLIF network.

During SuperComputing 2006 we have shown an application that gathered all the NDL files from the different GOLEs. Using a web interface, a user can select two endpoints from a list, which is generated from the gathered NDL information. After the two endpoints are selected, the application applies the Dijkstra algorithm to find the shortest path between the two endpoints. The network is displayed using Google Maps, as described in section 4.2. The shortest path through the network is then also drawn in the same figure, using highlighted links. A list of hops is also provided next to the map. Figure 4.3 shows the example output for a path between Seattle and Geneva.

4.4.3 Lightpath Monitoring in NetherLight

NDL can play an important role in lightpath monitoring as well. SARA developed Spotlight, a tool for lightpath monitoring in SURFnet6 and in NetherLight. To monitor the lightpaths, SARA uses NDL to specify their topology



Multi-Domain Pathfinding in GLIF

Below is an overview of the GLIF network (blue) and the path (red). The path is also enumerated below:

Go to the Path Finding page to select another path.

1. Hopi-sttl-force10:t0/1
2. Hopi-sttl-force10:t1/0
3. Hopi-chin-force10:t0/1
4. Hopi-chin-force10:g2/3
5. Netherlight-raptor:1/1/2
6. Netherlight-raptor:1/2/2
7. Tdm3.amsterdam1.netherlight.net:if1
8. Tdm3.amsterdam1.netherlight.net:503/3
9. Tdm1.geneva1.netherlight.net:12/1

Figure 4.3: *Pathfinding in GLIF, presented in Google Maps.*

details, and actively query the network elements involved. The output is stored in a network state database with alarm and configuration information.

The Spotlight application gathers all the information on the lightpaths in one place, providing engineers with a single overview of all lightpaths, along with their status. The user can then click on a specific lightpath to see an overview of the configuration, and the status of each of the segments of the lightpath. If a failure is detected somewhere in the lightpath route, this will be clearly indicated using a visualization of the lightpath.

The Spotlight tool has made monitoring of lightpaths in SURFnet6 and Netherlight a lot simpler. Engineers now have a single place where configuration and alarm data is correlated into a single view, allowing them to more quickly and accurately pinpoint the problem with a lightpath. The Spotlight application is available online, see [78].

4.5 Python NDL Toolkit

The applications that we have discussed have been mostly developed in isolation. We soon realised that much can be gained by providing a common toolkit for parsing and creating NDL files.

The pynt[79] provides a complete object model of NDL in Python. The toolkit allows a developer to grab an NDL file, request the toolkit to parse it and directly have all the objects available that were described. The properties of these instances can then be easily queried and updated. The updated description can then be easily exported to an NDL file.

The toolkit consists of six main components:

`pynt` is the basis of the package that has the definitions of the object model, and namespaces.

`pynt.input` contains modules that can parse from the RDF XML syntax, directly from nodes, or from OSPF (see section 4.3).

`pynt.output` provides an output mechanism to the RDF XML syntax, but also to graphs, and VNE configurations (see section 4.6).

`pynt.protocols` is a set of generic modules that provide base 'protocols' for interacting with command-lines, OSPF, or through TL1.

`pynt.technologies` contains a set of pre-defined technology descriptions to make it easier to describe nodes and interfaces of these technologies. Objects from these modules predefine layer properties, and make it easy to use well-known adaptations between layers.

`pynt.algorithms` implements different pathfinding algorithms, such as Dijkstra's shortest path algorithm, which can be used in single layer topologies, and a breadth-first pathfinding algorithm that is suitable for multi-layer pathfinding[13].

4.6 Virtual Network Experiments

Virtual Network Experiments (VNE)[6],[80] is a Python application that we have developed to allow users to easily create a virtual experimental network. The nodes in the network are implemented using User-Mode Linux (UML)[81], which allows for a lightweight virtualization, yet allows users to run their own applications.

VNE takes a topology description in XML. This XML file can be written manually or exported from an NDL file using pynt. VNE launches instances of UML following the provided configuration. These instances are provided with the proper configuration to create the network topology as described in the configuration file. The emulated network is provided using the `vde-switch` tool of the Virtual Distributed Ethernet project[82].

The contribution of VNE is that it makes it greatly simplifies the configuration and management of an emulated network. Configuring and launching UML instances, along with the right network topology is a very complicated task. VNE uses a clear XML syntax to define the configuration and network, allowing users to easily create and modify complex topologies, ready for experimentation.

We have also extended VNE to integrate it with DRAGON. The configuration file and VNE provide the proper configuration for the DRAGON toolkit, so that it is simple to create an emulated network that can be used to easily experiment with GMPLS. VNE was invaluable for us to gain understanding of GMPLS, and to test and validate the OSPF-TE export.

We can combine the OSPF-TE export to NDL, and the export from NDL to VNE. This allows us to sample the topology from any GMPLS network, and then create an emulated version of it using VNE. Both exports have been validated by completing a full circle, i.e. creating a VNE configuration, running that emulation, exporting the OSPF-TE data from that to NDL, and then create a VNE configuration from that again.

4.7 Conclusion

In this chapter we have shown the applications that we have developed that use NDL topology descriptions.

A first contribution of NDL has been that it allowed us to quickly create up to date maps of networks. The maps can be made abstractly using GraphViz, or if Global Positioning System (GPS) information is available, using Google Maps.

We have shown that descriptions in NDL can easily be generated using webforms, or by using our toolkit. The toolkit can also extract topology information in more automatic ways, either by logging into machines and parsing the command-line output, or by extracting it from management traffic. This makes it possible to provide a real-time view of the network.

The information retrieved from the network can support engineers with lightpath planning. With SARA we have developed an application that correlates the NDL topology information with the database of timeslot usage, this allows an engineer to quickly find an available path through the SURFnet6 network.

The configuration information of these lightpaths is also used to provide the engineers with a status overview. The configuration and state of lightpaths is presented in a graphical overview, allowing support engineers to quickly identify the root cause of incidents in the network, and make a better assessment of the consequential loss of services caused by this incident.

Developing the applications above has also inspired us to create the Python NDL Toolkit. The toolkit has made it very easy for us to test new ideas and create new applications. Combined with the Virtual Network Experiments application we can quickly create network topologies and test applications on them.

The applications described in this chapter have provided us with valuable insights about the possible use of NDL descriptions. They have allowed us to test NDL in real-world scenarios and solve problems. Creating the applications has helped us to gain a better understanding of the use-cases for the descriptions, and provided valuable feedback for the NDL schemata.

The applications in this chapter have been mostly aimed at single layer descriptions, multi-layer descriptions are a fairly recent development. Not many networks have described their multi-layer network in detail yet, and we are currently still exploring the possibilities.

Part II

Topology Aggregation in Multi-Domain Networks

Chapter 5

Introduction to Network Topology Aggregation

5.1 Introduction

In the previous chapters we have introduced the Network Description Language and shown its applications. However, applying NDL to real-life inter-domain networks is not straightforward. Network operators do not always wish to share their full topology, either for security, business, or for scalability reasons. On the other hand it is necessary to share some degree of topology data to enable inter-domain lightpath planning. It is possible to create a less detailed view of a network topology using *aggregation*. This means that details of the internal topology are aggregated into a virtual topology, which can then be published to other domains.

Topology aggregation is not a new research topic. It has been applied in ATM (Asynchronous Transfer Mode) networks, where it is also necessary to exchange topologies. Asynchronous Transfer Mode (ATM) is a network framing protocol developed in the mid 1980s. It encapsulates traffic into small fixed-length cells, over which virtual circuits can be defined, creating a circuit switched network. The topologies for ATM networks have been represented as graphs, with attributes for QoS such as capacity, available bandwidth, link delay, et cetera.

The rest of this chapter is organised as follows, in section 5.1.1 we discuss how pathfinding is performed in aggregated topologies. Then in section 5.2 we describe different aggregation strategies. In section 5.3 we summarize three performance studies of common aggregation methods, the first two study ATM, while the latter focuses on purely optical networks. We finish the chapter with a summary and comparison of the different studies in section 5.4.

5.1.1 Hierarchical Routing

One of the first standards on the area of topology aggregation is in the Private Network-to-Network Interface (PNNI) Specification[83] of the ATM forum. As the name of the standard implies, it specifies the interface between neighbouring networks. This standard provides some pointers on how aggregation can be performed, and it also defines how nodes can find routes through the aggregated network.

Routing through aggregated topologies is performed using hierarchical routing. Each node in the network has perfect knowledge of its local domain, and an aggregated view of the global topology. Using this knowledge, the node selects a path with aggregated hops. These hops get expanded at the border nodes into an underlying physical path. Together these expanded paths form a complete path between the node and its chosen end-point.

An example is shown in figure 5.1, we see a path request from node A1 to node C2 at the top, as seen from node A1. Notice that node A1 has complete visibility of its own domain, but only aggregated visibility of the other domains. Node A1 only knows that node C2 is in domain C, it does not know how it is connected to the rest of Cs network.

As soon as the request reaches domain B, the first node in B, B1 maps the path through the aggregated topology to the physical topology in B. B1 then forwards the request to B3, which forwards the request to domain C. In domain C the node C3 maps the path to the physical topology, and the path to C2 is completed. At the bottom of figure 5.1 we show the path mapped to the physical topology.

Note that in the aggregated view nothing is said about the availability inside the domains. If the links in domain B had no available capacity, then the path request would have failed.

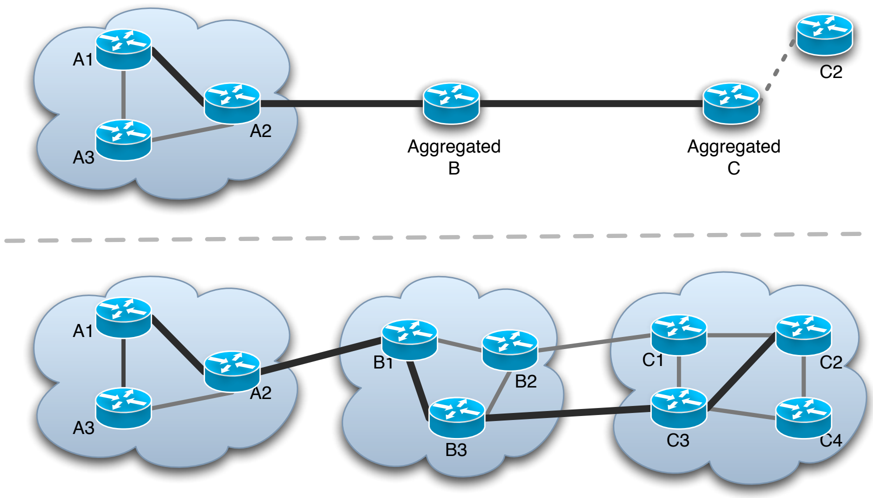


Figure 5.1: *An example of hierarchical routing through the aggregated topology (top) and the physical topology (bottom)*

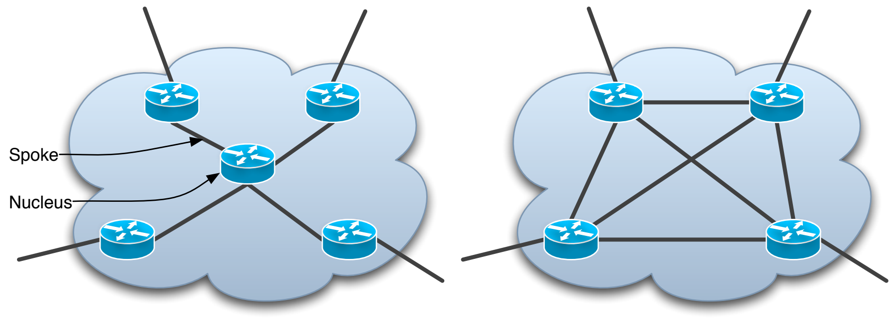


Figure 5.2: *Examples of Symmetric Star (left) and Full Mesh (right) aggregated topologies*

5.2 Topology Aggregation

There are different ways of creating aggregated topologies. In the example in figure 5.1 the domains are aggregated into single nodes and single links. Lee[84] provides an overview of this and two other aggregation methods. Below we describe the three most common aggregation strategies.

The *Simple Node* approach is the aggregation method that we have shown in figure 5.1. This is also called *Symmetric* or *Single Node* approach. The topology of a complete domain is replaced by a single node, which is directly connected to other domains. A single metric is advertised for the connectivity through this node. This single parameter implies that all connectivity through the node is considered to be symmetrical.

In the *Symmetric Star* approach the topology of a domain is aggregated using a central node. The border nodes of the original topology and their inter-domain connections are preserved. The intra-domain connections are represented by virtual links, *spokes*, to a virtual central node, the *nucleus*. All connectivity in this topology runs through the nucleus. This is often referred to as *Star* aggregation. An example is shown in figure 5.2 on the left, where we see an aggregated topology of a domain that is connected to four other domains. This is called a symmetric approach because a default definition for the properties of the spokes is used. All the spokes then have the same properties, unless explicitly specified otherwise.

On the right side of figure 5.2 is an example of the *Full Mesh* aggregation.

This aggregation preserves the most information of the original topology. Like in the Star aggregation, the border nodes are kept in the aggregated topology. Instead of a central node in the aggregated topology, there is a full mesh of connections between the border nodes. These aggregated connections between the border nodes can accurately describe the properties of the path through the domain, thus preserving the most important information for connections crossing the domain.

5.3 Performance Evaluation of Topology Aggregation

An aggregated topology hides details about the intra-domain connectivity. This means that inter-domain pathfinding with aggregated topologies is not always optimal. There are two types of performance impacts that can occur due to this lack of information.

First, in the aggregated topology there is no discernible difference between a domain that has many internal hops and a domain that has only one hop. This means that the paths found in the aggregated topology may not always be the shortest paths in the physical topology.

Second, a path found in the aggregated topology may not always have resources available to map it to the physical topology. This means that the path found in the aggregated topology is a false positive.

The impact on performance of the above effects can be measured in different ways. The rest of this section describes three different studies. Section 5.3.1 describes the study performed by Guo and Matta on a single emulated ATM network[85]. Section 5.3.2 presents the findings of Awerbuch et al. on several emulated topologies with ATM networking[86]. The study on aggregated topologies in an optical network by Liu et al.[87] is described in section 5.3.3. Finally in section 5.4 we compare the results of the three studies.

5.3.1 Performance Evaluation Study by Guo and Matta

Guo and Matta[85] have examined the performance of aggregation methods using the PNNI specification in ATM networks. They have examined the Simple Node, Star, and Full Mesh aggregation methods and analysed the performance on a single topology, randomly divided into domains. In their simulations they use two different traffic workloads:

Uniform : source and destination pairs are uniformly distributed over the network,

Skewed : some nodes are selected as destination for the majority of the connections.

In either case, traffic between two end-points is defined using different 'services'. A service is defined using time-dependent arrival rate, average life-time of the connection, maximum delay, and transmission rates (both average, and peak, including the length of the busy period). An example of a service is a video service, with a life-time of 20 minutes, an average rate of 0.7Mbps, a peak rate of 2.1Mbps, with a busy period of 0.3 seconds, and a statistical delay requirement $P(\text{end-to-end packet delay} > 50\text{msec}) < 10^{-4}$.

They also examine the performance of different route selection methodologies in the Full Mesh aggregation, based on different sets of metrics: utilization, utilization and hop-count, utilization and feasibility, or all three. Since pathfinding based on multiple metrics is NP-complete[88], they apply a heuristic. The authors first find a set of shortest paths, and then pick one of the set using the metric selection. This set is formed by the shortest paths in the aggregated graph, with length $minhop$, and paths of length $minhop + 1$. The decision of picking the right path is performed by the source node for the whole path as well as each of the border nodes for their segment of the path.

Their results show that under the skewed load, the Simple Node performs better or as well as Full Mesh and Star aggregations. Under uniform workload, both Full Mesh and Star outperform Simple Node significantly. In their simulations, the Star approach performs slightly worse than the Full Mesh as the former provides a less detailed view of the available bandwidth.

5.3.2 Performance Evaluation Study by Awerbuch et al.

Another performance evaluation of routing with aggregated topologies is given by Awerbuch et al. in [86]. The authors compare the following aggregation schemes:

Full Mesh The full cost matrix between the border nodes is advertised,

Diameter Star aggregation with a metric of half the diameter of the original network for each spoke,

Average Star aggregation with a metric of half the average cost between all pairs of border nodes,

MST Minimum Spanning Tree of the border nodes,

RST Random Spanning Tree of the border nodes,

Spanner A t-spanner, which is a sub-graph that guarantees a worst-pair distortion of at most a factor t. The authors have used $t = 2$ for their experiments.

These aggregation schemes are examined in combination with two methods for metric values, a constant and an exponential cost function. In the constant case, the link metric is fixed, regardless of the available bandwidth on the link, this is equivalent to (weighted) min-hop routing. The exponential cost function on the other hand uses a metric that increases exponentially as the available bandwidth decreases.

In their simulations they examine both specifically designed topologies, as well as randomly generated networks. The former topologies have been designed to maximize the penalty for routing errors, so that difference in routing schemes is emphasized. They have used ring-like topologies, as well as a self-similar hierarchical topology, which is a three-level hierarchical topology, where each level is a similar multi-stage graph.

The random networks were generated using an adapted Waxman method[89]. In this method nodes are randomly placed on a grid, and the probability of a link between two nodes decays exponentially with their Euclidean distance. They adapted the method to prevent nodes exceeding a degree of four, but the degree must always be at least two.

In the simulation the requests are modelled by a Poisson process, and the holding time is exponentially distributed. The inter-arrival time of the requests are exponentially distributed with a mean of 1 when the traffic load is 100%. The load on the network is defined as (avg. request rate $\times$ avg. hold time)/ capacity of the avg. min. cut (over all source-destination pairs). The results are analysed using two parameters:

Throughput defined as the fraction of attempted connections that are realized,

Control Load the average number of crank-backs required per realized connection.

A *crank-back* is performed when a path that has been found in the aggregated topology is not available in the physical topology. The information regarding this false-positive is then used to locally update the view of the network, and is then used in subsequent attempts to find a correct path. The amount of crank-backs also gives an indication for the set-up delay, because route recalculation when crank-backs occur is a time consuming task.

The results of the simulations confirm the authors' earlier theoretical work[90] that an exponential metric performs much better than a constant metric. In fact, even the worst aggregation strategy in the exponential metric simulation performs better than the best aggregation method in the constant metric simulation.

In their simulations the authors also varied the link-delay, so that topology updates between domains, and reservation requests take longer to travel through the network. The effect of the higher delay is that the performance difference between all the aggregation methods becomes more pronounced.

On ring-like topologies the Random Spanning Tree and the two Star aggregation methods perform worse than the other methods. The Random Spanning Tree method shows significantly the worst performance, with crank-back rates of up to four times higher than the other aggregation methods. The performance difference of the two Star aggregation methods is less pronounced, but still significant. The difference becomes more pronounced in a self-similar hierarchical topology, were the Star methods perform significantly worse than the other methods, including the RST.

Also in the random topologies the two Star aggregation methods and the RST perform the worst. The Star aggregation methods require more crank-backs, with a slightly worse throughput, and the RST method requires significantly more crank-backs while providing a lower throughput.

In the above simulations each aggregated domain topology is updated once the bandwidth availability of any of its constituent links changes. The authors propose a more practical re-aggregation policy called the logarithmic update re-aggregation policy. In this case the bandwidth b of a link is divided into $(log\ b) + 1$ blocks. Re-aggregation of the domain topology is only triggered when the residual bandwidth crosses these division boundaries. With this update scheme in the simulations, there is no significant difference in throughput, and only a slight increase in the number of crank-backs for all aggregation schemes. The only exception is MST, where there is a 5-20% increase in the number of crank-backs with the new re-aggregation policy. This re-aggregation policy requires significantly less amount of updates, and even under high-load settings it requires over 30% less updates than the regular policy.

5.3.3 Aggregated Topologies in Optical Networks

While aggregated topologies have been studied extensively for ATM networks, they have not been studied extensively for pure optical networks. Liu et al.[87] have applied the Simple Node and Full Mesh aggregations to inter-domain WDM networks. In both cases a wavelength availability vector is used to represent the available wavelengths.

The aggregation strategies are tested using two different lightpath provisioning strategies. They use the term *transparent* for lightpaths that have the same end-to-end wavelength, and *translucent* for lightpaths that use different wavelengths using optical-electrical-optical (OEO) conversions.

In the transparent case paths are selected using a K-shortest path algorithm and a widest-shortest approach[91]. A wavelength for this path is then selected using either the most-used or least-used wavelength. The most-used or least-used wavelength selection method mean that the available wavelength on that path are ordered by their use in the rest of the network[92].

In the translucent case paths are also first selected using K-shortest path, but then the candidate path is selected using either minimum hop count, or minimum converter count (based on the aggregated view only). The same wavelength selection strategies are used, but this time only for segments between converters. This makes the translucent routing inherently more complex.

The authors show results of their simulation in OPNET[93], a network simulator tool, on a topology with 9 domains and 19 inter-domain links, using 8 or 16 wavelengths. The results show that selecting the most-used wavelength outperforms selecting the least-used wavelength. For transparent lightpaths Full Mesh performs better than Single Node aggregation, with a difference of up to 45% at 120 Erlang[1] with 16 wavelengths. The authors note that the difference in blocking reduction becomes smaller when the intra-domain connectivity decreases.

In the translucent case, it makes little difference in terms of blocking probability whether paths are selected based on the least number of hops or least number of converters. There is still a difference between the two aggregation schemes, but the difference is smaller than in the transparent case.

The same authors show more results on the same topology in [94]. They show that using conversion capabilities on all nodes provides little improvement over just converting at the border nodes. However, note that in the test topology they are using, there are relatively few non-border nodes.

[1]Erlang is a unit to describe statistical measure of the traffic volume. Traffic of one Erlang refers to a single resource being in continuous use, or two channels being at fifty percent use each, and so on.

5.4 Summary

In this chapter we have discussed different methods for topology aggregation, and described three studies evaluating the performance of pathfinding in aggregated topologies. Unfortunately, all three studies have used a different measure of performance.

The results that Guo and Matta show in their paper use revenue as an indication of the pathfinding performance. They define revenue as the total amount of bandwidth in use by connections at an instant t.

In the study by Awerbuch et al. the performance has been measured by throughput and control load. The throughput is defined by the fraction of attempted connections that were realized. The control load is represented by the average number of crank-backs per realized connection.

Finally in the study by Liu et al. the performance is determined by the blocking probability compared to the load on the network.

The different performance measurements make it hard to directly compare the results of the three studies. However, it is possible to detect a general trend in the results. As can be expected, the Full Mesh aggregation performs best in all studies, and the Single-Node aggregation performs worst in most cases compared to the other strategies.

The Star aggregation strategy has only been examined in the first two studies. Unfortunately these studies provide conflicting results. In the study by Guo and Matta the Star aggregation performs as well as or better than the Full Mesh, while in the study by Awerbuch et al. several Star aggregation strategies are examined, and they all perform significantly worse than Full Mesh.

The first two studies above have also been performed with ATM networks in mind, and this does not readily map to inter-domain optical and hybrid networks. While both ATM and optical networks are circuit based, the availability of circuits is different. A request in ATM takes up a portion of an edge, while in optical networks a request often takes up the whole edge. It is therefore not completely clear what impact topology aggregation has on inter-domain pathfinding in optical and hybrid networks. In the next chapter I describe an emulation experiment and its results to accurately determine this impact.

Chapter 6

Emulations of Aggregated Network Topologies

6.1 Introduction

In the previous chapter we have introduced pathfinding using aggregated topologies. We have discussed several studies that have examined the impact of aggregation on pathfinding. Unfortunately, the combined results of these studies are not conclusive about the effects. Two of the studies conflict on the performance of the Star aggregation method. These studies have also been performed in the context of ATM networks.

In this chapter we examine how topology aggregation can be applied to optical networks. We define how different aggregated views can be created in NDL from full topology descriptions. The Full Mesh, Star and Single Node aggregations can automatically be generated from a full NDL topology description.

To accurately determine the effect of the different aggregation strategies we have implemented an emulation experiment. In this experiment we generate random inter-domain topologies and do pathfinding on them. We compare the results of the same pair-sets for the different aggregation strategies and the control case, using the full topology. We also compare our results with the earlier results of topology aggregation in ATM networks.

This chapter continues by examining different aggregation methods in the context of NDL, and defines how these can be created from full NDL descriptions in section 6.2. The experimental setup is discussed in section 6.3 and section 6.4 shows the results of the experiments. In section 6.5 we finish this chapter with a discussion of the results.

6.2 Aggregation Methods

As we have shown before, topologies of network domains can be aggregated to a different degree. In our experiments we examine three different aggregation strategies, for which we provide the definitions here:

Full View This is the control case, where domains use no aggregation at all and publish their full topologies,

Full Mesh Domains only publish their edge nodes, the connections between the edge nodes are described using a full mesh,

Star Domains only publish their edge nodes, the connections between the edge nodes are described using a virtual central node,

Simple Node The domain is presented as a single node, leaving out all information about the internal network.

We examine the performance of pathfinding in the different aggregation methods with the following assumptions:

Single Layer We assume a single layered network. A good understanding of the impact of aggregation in single-layer topologies might later be of use also in multi-layer networks, where path finding is an even more complex problem [70].

Homogeneous Capacity We assume a single unit of bandwidth for all links in the network. We do allow for multiple connections between nodes, so it is possible to create a mapping for networks with heterogeneous bandwidths.

Infinite Reservations Another assumption is that a reservation is for an infinite amount of time. We generate random topologies, and do multiple passes of the same topology. This means that the total averaged results show the trend of how the network performs under increasing load.

Aggregated Metric The metric of a link is often used to represent the distance of that link. On our topologies the metric for all links is 1, even for links in the aggregated graph which can be based on much longer paths. This is used to minimize the information about the intra-domain connectivity, but also to maximize the distortion caused by the aggregation.

Inter-Domain Pairs In our experiments we only use source-destination pairs that are in different domains. Since nodes always have full knowledge of their own domain, aggregation will not have an effect on intra-domain pathfinding.

Unlabelled Links Previously we described the study by Liu et al. who have researched the effect of aggregation on WDM networks with different numbers of available wavelengths. We are interested in the general effect of aggregation on inter-domain pathfinding. In this study we use unlabelled links.

In the descriptions below we do use a mapping of network topologies onto graphs. In this case this is possible because we only use a single layer topology, and for the aggregation methods it is not necessary to identify different interfaces. The actual aggregated descriptions are created in NDL, complete with interface objects.

6.2.1 Formal Definitions of Topology Aggregation

In the previous chapter we have described what different aggregated topologies look like. However, we have not defined explicitly how to map a full topology to an aggregated version. Below we formally define how we have implemented the aggregation strategies in our experiments.

First some notation and definitions. We define a multi-domain topology as $T = (N, D, L)$:

N is a set of nodes: $N = \{n_1, n_2, \ldots\}$

D is a set of domains: $D = \{d_1, d_2, \ldots, d_x\}$ such that:

- Domains are sets of nodes: $\forall d_i \in D : d_i \subset N$
- All nodes must be part of a domain: $\bigcup_{i=1}^{x} d_i = N$
- Nodes can only be in one domain: $\forall d_i, d_j \in D : d_i \neq d_j \rightarrow d_i \cap d_j = \varnothing$

L is a set of links: $L \subseteq N \times N$, we assume bi-directional connections:
$\forall n_i, n_j : (n_i, n_j) \in L \rightarrow (n_j, n_i) \in L$.

A *Full Mesh aggregation* $T_f = (N_f, D_f, L_f)$ of a topology T is defined as follows:

N_f contains only domain-edge nodes:
$\forall n_i, n_j \in N \exists d_x, d_y \in D : (n_i, n_j) \in L \wedge n_i \in d_x \wedge n_j \in d_y \wedge d_x \neq d_y \rightarrow n_i, n_j \in N_f$,

D_f is a set of domains $D_f = \{d'_1, d'_2, \ldots, d'_x\}$ such that:
$\forall d'_i \in D_f \forall n_x \in d'_i : n_x \in N_f \wedge \exists! d_i \in D \wedge d'_i \subseteq d_i$

L_f is constructed in two steps, first we take all the original inter-domain links:
$\forall n_i, n_j \in N_f \exists d_x, d_y \in D : (n_i, n_j) \in L \wedge n_i \in d_x \wedge n_j \in d_y \wedge d_x \neq d_y \rightarrow (n_i, n_j) \in L_f$,
and secondly we add intra-domain connections based on the original connectivity. Even if multiple paths are available between different edge nodes, we only add a single link:
$(n_i, n_j) \in L_f$ if there is a path $n_i n_x \ldots n_y n_j$ with $n_x, \ldots, n_y \in d$ and $(n_i, n_x), \ldots, (n_y, n_j) \in L$.

A *Star aggregation* $T_s = (N_s, D_s, L_s)$ of a topology T is defined as:

N_s contains domain-edge nodes, plus a centre node for each domain:
$\forall n_i, n_j \in N \exists d_x, d_y \in D : (n_i, n_j) \in L \wedge n_i \in d_x \wedge n_j \in d_y \wedge d_x \neq d_y \rightarrow n_i, n_j \in N_s$
For each of the domains, we add exactly one centre node:
$\forall d \in D \exists! n_d : n_d \in N_s \wedge n_d \notin N$,

D_s is a set of domains $D_s = \{d'_1, d'_2, \ldots, d'_x\}$ such that:
$\forall n_i \in N_s \exists d_x \in D \exists d'_x \in D_s : n_i \in d_x \rightarrow n_i \in d'_x \wedge d'_x \in D_s$ The centre node is also part of its domain: $\forall d \in D \exists n_d \in N_s \exists d' \in D_s : n_d \in d'$,

L_s is constructed in two steps, first we take all the original inter-domain links:
$\forall n_i, n_j \in N_s \exists d_x, d_y \in D : (n_i, n_j) \in L \wedge n_i \in d_x \wedge n_j \in d_y \wedge d_x \neq d_y \rightarrow (n_i, n_j) \in L_s$,
secondly, all the edge nodes are connected to the centre node, if they have available intra-domain connections:
$\forall n_i \in N_s \exists d_x \in D : n_i \in d_x \wedge \exists n_k \in d_x \exists n_{d_x} \in N_s : (n_i, n_k) \in L \rightarrow (n_i, n_{d_x}) \in L_s$.

A *Simple Node aggregation* $T_n = (N_n, D_n, L_n)$ of a topology T is defined as:

N_n contains only one node per domain: $\forall_{i=1}^{x} n_i \in N_n$ with $x = |D|$

D_n contains the same amount of domains, but each only has a single element:
$\forall d_i \in D \exists d'_i \in D_n \exists n_i \in N_n : d'_i = n_i$ and $\bigcup_{i=1}^{x} d'_i = N_n$ with $x = |D_n|$

L_n contains only the inter-domain links:
$\forall (n_i, n_j) \in L \exists d_x, d_y \in D \exists n_x, n_y \in N_n : n_i \in d_x \wedge n_j \in d_y \wedge d_x \neq d_y \rightarrow (n_x, n_y) \in L_n$

6.2.2 Topology Aggregation from NDL Descriptions

We can use these logical definitions to create different aggregated views from a full NDL domain topology description. Below we describe how we select and transform the NDL objects to create the aggregated views.

For the *Full Mesh* we take the device objects that have an inter-domain connection, including their switching matrix. We then describe only the interfaces of those devices that have inter-domain connections. For each device we check whether it is possible to reach the other inter-domain devices through the domain. If so, we create a new interface object for each, with a *connectedTo* statement between them, and we also connect them through the switching matrix. The result is a topology description containing only the edge devices, with their edge interfaces, and an aggregated description of the intra-domain topology through the virtual interfaces.

Initially in an unused network, all edge nodes will be able to reach each other, since domains are usually connected graphs. When more and more links are reserved it is possible that domains become disconnected graphs. The aggregation method Full Mesh, and Star can describe this, Full Mesh without updating will not.

In the *Star* aggregation we also start with the edge devices. We describe only the interfaces of those devices that have inter-domain connections. We add a new virtual device to the aggregated graph for the centre of the domain, the nucleus. For each edge device we check whether it has any intra-domain interfaces with available bandwidth. If it does, we add a connection between the device, and the centre node. This results in a topology description containing only the edge nodes and a (virtual) intra-domain star network.

Also in the Star network, edge nodes will initially be able to reach the intra-domain network. Once the network usage grows, it is possible that an edge node will no longer have available internal connectivity. This is reflected in the Star aggregation, the node then becomes disconnected from the rest of the domain.

This node can still be of service by switching from one of its inter-domain connections to the other.

The *Simple Node* topology is simpler to create: We take the network domain object with the name of the domain, n_d in the mathematical description, and we add a switching matrix object to the network domain object. Then from the nodes with inter-domain connections we select the interface objects of those inter-domain connections. We add these interface objects directly to the domain object. We also copy the inter-domain *connectedTo* properties of the original view into the aggregated view, and connect all the interfaces to the switching matrix. The result is a network domain object with interfaces, which are connected to other domains, and internally connected with a full mesh through the switching matrix.

This Simple Node topology does not publish any details about the intra-domain connectivity. Other domains reading this topology will assume that the intra-domain network is fully connected. Once the network usage increases, it is possible that the intra-domain network is no longer connected. However, there is no way to publish this using the Single Node strategy.

6.3 Experimental Setup

We have implemented an experimental setup for testing the different aggregation methods. The steps in the experiment are as follows:

- Take a number d for the number domains, and n for the number of nodes in each of the domains,
- Generate a random graph G with d domains, and n nodes,
- Create a randomly ordered list l of inter-domain endpoint pairs,
- For each of the aggregation methods:
  - Create a simulated network of the graph G,
  - For each pair (x, y) in l, find a path between x and y in the aggregated view:
    - If there is no path, continue.
    - If there is a path:
      - Translate the path in the aggregated view to a full path,

- If the full path is available, reserve the path and record the result,
- If the full path is not available, record a false positive

In the rest of this section we describe the details of our experimental setup. First we describe how we generate the network graphs and the source and destination pairs, and in section 6.3.2 we describe the pathfinding in aggregated views, and how we translate these paths back to the underlying graph.

6.3.1 Generating the Graphs and Pairs

At the start of each experiment run we generate a random graph and pair-lists. In the experiment we can use different graph sizes to test the impact of the number of domains, and the number of nodes per domain.

There are a large number of ways to generate random graphs. Barabási and Albert have shown in 1999[95] that many complex networks exhibit a scale-free property. That is, the probability $P(k)$ that a vertex in the network interacts with k other vertices decays as a power law, following $P(k) \sim k^{-\gamma}$. The value of γ varies with different types of graphs, but is usually $2 \leq \gamma \leq 3$. In case of the BGP router network, the value of $\gamma \sim 2.3$.

However, it should be noted that it is unclear whether current optical networks, such as the GLIF network, are scale-free. The current networks are too small and nodes have too small degrees to come to a definitive conclusion[96]. We expect that as optical networks grow larger, they will also follow the power-law distribution.

The graphs are generated using the Barabási-Albert algorithm as implemented in the NetworkX Python module[97]. This algorithm will generate graphs with a value of γ that tend to 2.9 ± 0.1. Using this algorithm we generate d graphs with their n nodes. We then generate another graph with the same algorithm with d nodes, and use that as the inter-domain graph, randomly picking nodes from each of the domain graphs to provide the inter-domain links. Analysis of the results of this generation method shows that γ averages to ~ 2.3, which is comparable to the BGP network.

In our experiments we then enumerate all the pairs (x, y) where x and y are in different domains. So in total we have $(d \cdot n) \cdot ((d-1) \cdot n) \cdot \frac{1}{2}$ number of pairs. And before each run we shuffle this pair-list.

The graph is then converted to an emulated network using our `pynt` package. The result of the conversion is the same if the descriptions of the domains would have been imported from NDL descriptions or from OSPF data.

6.3.2 Pathfinding Using Aggregations

We perform four different pathfinding operations using the same shuffled set of endpoints pairs on different views of the same graph. We use a full view, a Full Mesh aggregation with and without updates, a Star aggregation, and a Simple Node aggregation.

As a baseline we perform pathfinding on the graph using complete information of the graph, that is, we work with the *full graph*. This shows the ideal situation which the aggregation methods should approximate. As with all the other aggregation methods, we perform pathfinding using a standard Dijkstra's shortest path algorithm.

In the *Full Mesh* method the topology shows only the boundary nodes. The intra-domain connections are created by taking each boundary node and performing a path find call on it to all other boundary node of that domain. If there is a path between that node and another boundary node, then we add a connection between those two nodes. The inter-domain connections are not aggregated, and used as is. The Full Mesh view is recreated before each pathfinding attempt to reflect the current status of the network.

For each pathfinding attempt on pair (x, y), we add x and y to the aggregated graph. To connect them we iterate over the boundary nodes of their respective domains, and if a path can be found to that boundary node, then we add a connection in the graph.

With the nodes x and y added to the Full Mesh graph, we try to find an inter-domain path between them. If a path is found in the Full Mesh aggregated graph we convert each path through a domain by mapping it back to the underlying path in the full topology. Before moving on the next pair, the source and destination nodes ,x and y, are removed from the graph, if they are not edge-nodes.

We also perform the experiment with the same initial Full Mesh view of the topology, but without any intermediate updates. The updates in this case refer to the updating of intra-domain topology after a path has been found. Without the updates it is possible that a path is found in the aggregated graph where the underlying path is no longer available. In this case we record the result as a false-positive.

In the Star aggregation method we also keep the boundary nodes, and link them to a virtual device, the nucleus. The links from a boundary node to the nucleus is created if the boundary node has any intra-domain connectivity available. After each successful reservation we check the intra-domain connectivity again and perform updates where needed.

The last aggregation method is to collapse domains into a *Simple Node* in the aggregated graph. We have implemented this by creating an aggregated view of the graph with only domains. The inter-domain interfaces of the boundary nodes are added to aggregated graph, along with their connections. As noted before, the Simple Node view does not provide details about the intra-domain network. Once the network usage increases, it is possible that the domain becomes a disconnected graph. If a path request in the aggregated view traverses such a domain and can not be translated to an underlying path, we record the result as a false-positive.

For each pathfinding attempt on pair (x, y) in the Star and the Simple Node aggregation we add x and y to the aggregated graph with a connection to their centre and domain nodes respectively. If a path is found between x and y we convert that path into a full path by replacing each domain hop in the path with the result of a path find operation between the boundary interfaces of that domain. If the complete path can be converted it is reserved. If no path can be found within one of the domains, we record a false-positive. Before moving on to the next pair, the nodes x and y are removed from the graph.

Note that the Full Mesh without update is very similar to the Simple Node method. The Full Mesh without update shows the edge nodes, with a full mesh between them. The Simple Node graph only shows a single point, implicitly assuming full connectivity between all its inter-domain interfaces.

Note also that regardless of what kind of intra-domain updating occurs, all of the aggregation methods do update the inter-domain connectivity.

6.4 Results of the Emulations

In each experiment run, when a path is found, we record the result (success or false-positive), length of that path, and the new resource usage of the network. From each run we have a large set of results (10^4 data points per aggregation method).

Since we are using results from experiments with different graph sizes, we normalize the results by using a relative index on the pair number. So the first pair has a relative index of 1 divided by the total number of pairs. This puts all results on a scale between 0 and 1. The resource usage numbers have also been normalized to their total graph sizes.

We analyse our results using the R statistical analysis tool[98]. We use the coefficient of determination (R^2) to determine the goodness of fit of our fit functions. This is calculated using the formula given in equation 6.1 where

SS_{errors} and SS_{total} are the sum of squares of the errors and the total sum of squares, y_i and $\hat{y}_i$ are respectively the observed and predicted value, and $\bar{y}$ is the mean of the observed values. R^2 indicates the explained variance of a model, and measures how well future outcomes are likely predicted by the model. The result is a number between 0 and 1, where a value of 1 means that the model explains all the variability of the data.

$$R^2 = 1 - \frac{SS_{errors}}{SS_{total}} = 1 - \frac{\sum\limits_i (\hat{y}_i - y_i)^2}{\sum\limits_i (y_i - \bar{y})^2} \tag{6.1}$$

6.4.1 Fit Functions

Before we examine the performance of the different aggregation strategies, we first inspect the behaviour of our control case, pathfinding using the complete view. The length of the paths will behave differently as the network is gradually filled up. Figure 6.1 shows a scatterplot of the path length development. We can see that initially it is fairly stable, the network is still empty, almost any path will succeed, and paths will start at the average path length in that network, and gradually increase. As the network starts to fill up, most requests will still succeed, but the path length peaks as longer and longer detours are taken. This grows until the network becomes nearly saturated, meaning that large parts of the network are in use. Then only small disconnected parts of the network remain available, and the chance of success depends on the distance in the network. The path lengths will gradually decrease to the minimum, i.e. paths between neighbours.

To show the combined effects of the path length and the success rate we plot the development of inter-domain resource usage over the successive requests, see figure 6.2. There are two different behaviours, the initial increase in path lengths, and the slow decrease once the network reaches a saturation point. Therefore we use two different functions for fitting to the results. We determine the split by the path length peak, the peak and everything before is the first part, and afterwards is the second part.

Initially there is a constant success rate, and the path length increases linearly, and the inter-domain usage also increases linearly. We use a linear function starting at zero to fit with, as shown in equation 6.2. This fit explains roughly 90% of the variance, and is shown in figure 6.3.

$$\mathit{InterDomainUsage} = A \cdot \mathit{RelativeIndex} \tag{6.2}$$

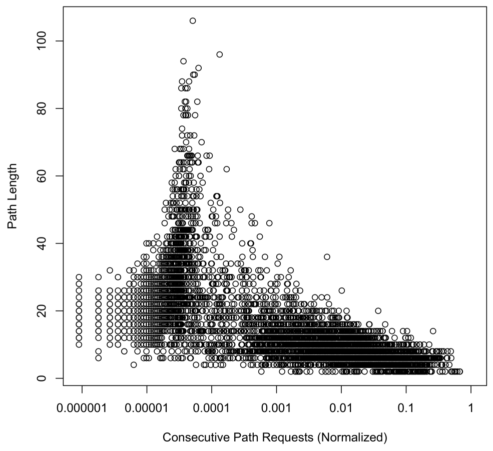


Figure 6.1: *A scatterplot showing the path length distribution in full view pathfinding*

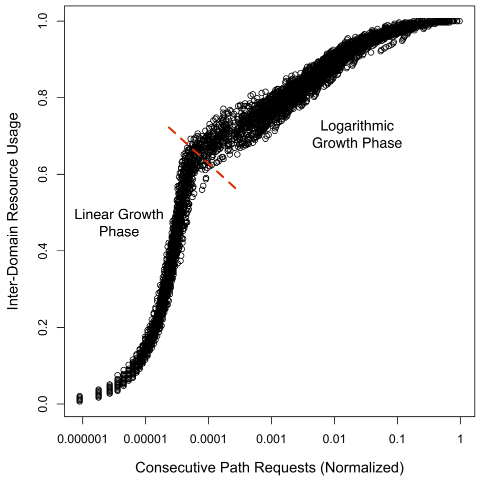


Figure 6.2: *A scatterplot showing the two different phases in inter-domain resource usage distribution in full view pathfinding*

In the second part of the plot the behaviour is dictated by a decreasing success rate, and path length. Initially we presumed that this behaviour could be predicted using a standard growth function towards an asymptote as shown in equation 6.3. In this case the asymptote is a completely filled network where the inter-domain usage is 1.

$$InterDomainUsage = 1 - A \cdot e^{(-B \cdot RelativeIndex)} \tag{6.3}$$

The fitted function shows a reasonable result, explaining about 87% of the variance. However, if we overlay this function to the scatterplot of the results, it clearly shows a different trend than the actual results. The trend shows more of a logarithmic form, so we have also fit the function given in equation 6.4. This fit shows a much better result, explaining over 94% of the variance.

$$InterDomainUsage = A \cdot log(RelativeIndex) + B \tag{6.4}$$

Both fits are shown on the scatterplot in figure 6.4, with the exponential fit in red, and the logarithmic fit in green. The only caveat with the second fit is that it over-predicts when the network has been nearly filled. However, the paths in that final section will all be very short ones, regardless of the aggregation strategy. We expect to see most significant differences in the first part of this second section.

6.4.2 Domain Sizes

In our experiments we have also examined the effect of domain sizes on the results. In figure 6.5 we show the initial fits for $(d = 50, n = 5)$ and $(d = 50, n = 50)$, in figure 6.6 the fits for the second section. Note that the horizontal scale is different for each of the plots. The fitted values (A, B), the error (σ), and the explained variance (R^2) are shown in tables 6.1 and 6.2.

Both graphs show a very similar performance of the aggregation methods. However, the performance difference is larger in the configurations with less nodes per domain. Part of this difference can be explained by the different ratio of the number of pairs to inter-domain resources. However this would only cause the graphs to shift on the scale, keeping the relative difference. The smaller difference between the fitted lines with $(d = 50, n = 50)$ can only be explained by the increase in intra-domain resources. Since we are interested in the performance difference of the different aggregation strategies, we will use small domain sizes.

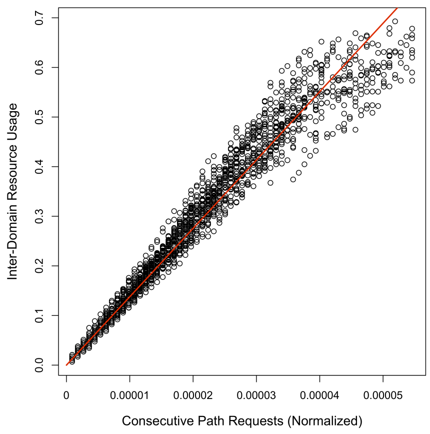


Figure 6.3: *A scatterplot showing only the initial linear growth section, along with the fitted function in full view pathfinding*

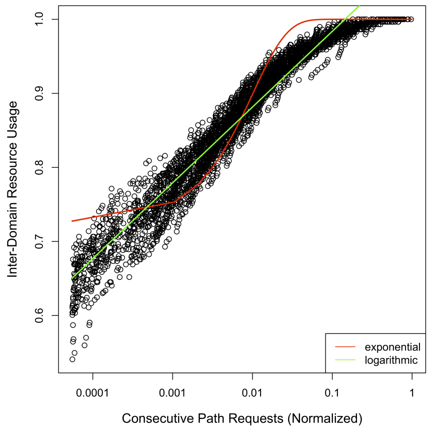
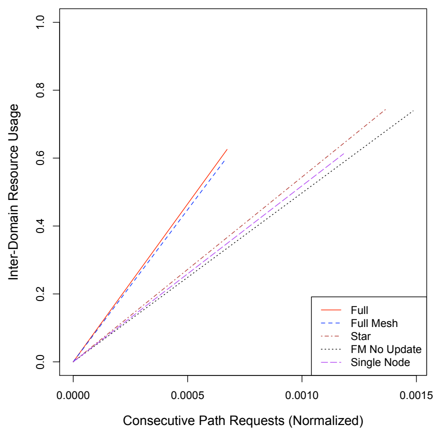
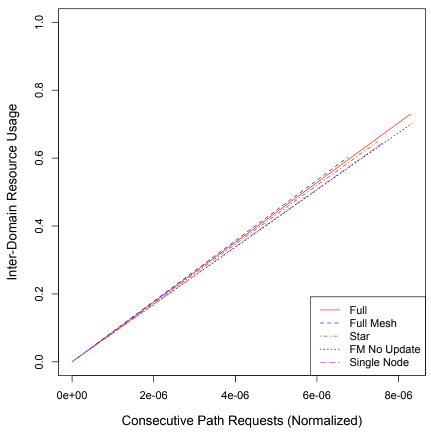


Figure 6.4: *A scatterplot of the inter-domain resource usage distribution in full view pathfinding, along with the exponential fit in red, and the logarithmic fit in green.*



(a) $(d = 50, n = 5)$



(b) $(d = 50, n = 50)$

Figure 6.5: *Plotted fits for the initial growth phase for* $(d = 50, n = 5)$ *and* $(d = 50, n = 50)$

Strategy	A	σ	R^2	A	σ	R^2
Full	930	3.6	0.97	88026	308	0.98
Full Mesh	896	3.3	0.97	89184	332	0.98
FM no updates	497	3.0	0.92	84374	324	0.97
Star	544	3.2	0.93	86646	314	0.98
Single Node	518	3.3	0.92	84715	315	0.98

(a) $(d = 50, n = 5)$ **(b)** $(d = 50, n = 50)$

Table 6.1: *Fitted values in the initial growth phase*

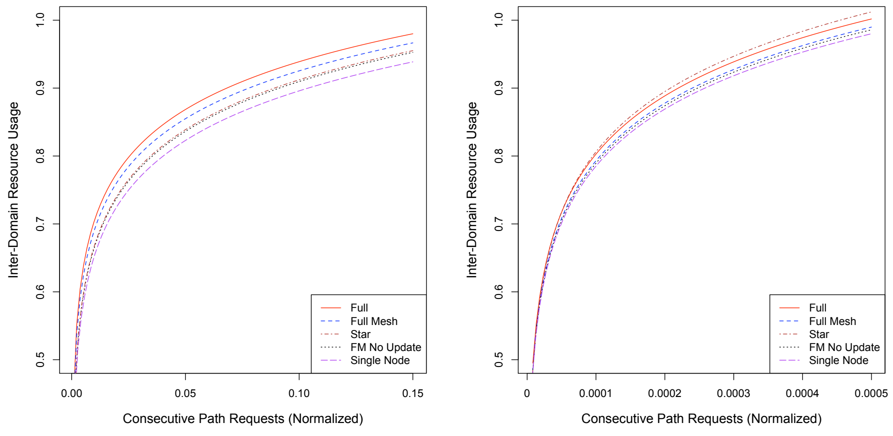
Strategy	**variable**	**value**	σ	$\mathbf{R^2}$
Full	A	0.1015	0.0004	0.89
	B	1.1727	0.0027	
Full Mesh	A	0.1016	0.0004	0.90
	B	1.1594	0.0024	
FM no updates	A	0.1058	0.0003	0.93
	B	1.1535	0.0019	
Star	A	0.1059	0.0004	0.91
	B	1.1561	0.0026	
Single Node	A	0.1053	0.0003	0.94
	B	1.1386	0.0017	

(a) $(d = 50, n = 5)$

Strategy	**variable**	**value**	σ	$\mathbf{R^2}$
Full	A	0.1236	0.0008	0.85
	B	1.9416	0.0095	
Full Mesh	A	0.1223	0.0008	0.85
	B	1.9196	0.0094	
FM no updates	A	0.1220	0.0008	0.85
	B	1.913	0.009	
Star	A	0.1280	0.0008	0.87
	B	1.9853	0.0092	
Single Node	A	0.1211	0.0008	0.85
	B	1.9005	0.0094	

(b) $(d = 50, n = 50)$

Table 6.2: *Fitted values in the logarithmic growth phase*



(a) $(d = 50, n = 5)$ **(b)** $(d = 50, n = 50)$

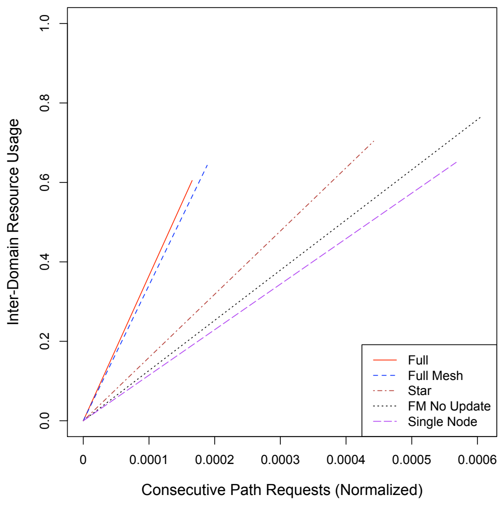
Figure 6.6: *Plotted fits for the logarithmic growth phase for* $(d = 50, n = 5)$ *and* $(d = 50, n = 50)$

The smaller domain sizes also more closely follow the current real-world situation. The global GLIF network currently consists of several dozen networks, most of which consist of only several nodes. The largest domain in GLIF is currently the Internet2[99] network, which consists of about 15 nodes.

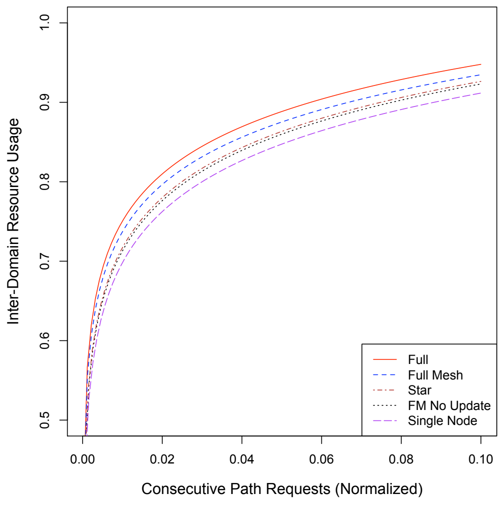
The number of domains does not have an impact on the difference in the aggregation strategies. However, using more domains means that there are will be more nodes in the whole network, so there will be more available pairs. The more pairs, the more fine-grained the results will be. To keep things scalable we have chosen to use scenarios with 150 domains. For the remainder of this chapter we have therefore used a configuration with $(d = 150, n = 5)$.

6.4.3 Results on Inter-Domain Pathfinding

With the fit functions selected, we can examine the performance of inter-domain pathfinding in the different aggregation strategies. In figure 6.7 we show the fits for the results on the graphs with $(d = 150, n = 5)$. Table 6.3 shows the fitted values, along with their errors, and the coefficient of determination.



(a) *Initial growth phase*



(b) *Logarithmic growth phase*

Figure 6.7: *Fits of initial and logarithmic growth phases for* $(d = 150, n = 5)$

Strategy	**A**	σ	$\mathbf{R^2}$
Full	3649	7	0.98
Full Mesh	3409	6	0.98
FM no updates	1263	6	0.90
Star	1591	7	0.9
Single Node	1145	6	0.91

(a) *Initial growth phase*

Strategy	**variable**	**value**	σ	$\mathbf{R^2}$
Full	A	0.0857	0.0002	0.91
	B	1.1451	0.0017	
Full Mesh	A	0.0858	0.0002	0.92
	B	1.1322	0.0015	
FM no updates	A	0.0912	0.0001	0.95
	B	1.1331	0.0011	
Star	A	0.0908	0.0002	0.93
	B	1.1355	0.0015	
Single Node	A	0.0926	0.0001	0.96
	B	1.1250	0.0009	

(b) *Logarithmic growth phase*

Table 6.3: *Fitted values in the initial and logarithmic growth phases for* $(d = 150, n = 5)$

Initial Linear Growth Phase The graph of the first section in figure (a) shows the initial growth. Recall that the boundary for the first section is determined by the peak in the path lengths. The fitted graphs all end around 60% resource usage, however the more aggregated the longer it takes to get there. The results in table (a) also show that the Full Mesh aggregation strategy performs very close to the Full View, and that there is a large gap in performance with the other aggregation strategies. Of these the Star aggregation performs better than the last two, the Full Mesh without updates, and Single node, which both show very similar performance.

Logarithmic Growth Phase The same difference in performance continues in the second section of the fit. The Full Mesh aggregation performs almost perfectly compared to the control. However, here the difference with the other strategies is far less pronounced. The Star aggregation method shows a significantly lower performance, with the Full Mesh without updates just below it. The Single Node aggregation method clearly performs worst of all the aggregation methods.

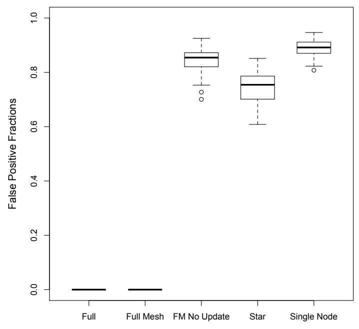


Figure 6.8: *The fraction of false positives in the linear and logarithmic phases for* $(d = 150, n = 5)$

False Positives In figure 6.8 we show the fraction of false positives in a barplot. It should be noted that these false positives only occur in the logarithmic growth phase. None of the aggregation methods show false positives in their linear growth phase.

It is not possible to have false positives with the full view, nor with the

Full Mesh view with updates. In the latter case, the update mechanism always makes the graph reflect the current availability in the network. The number of false positives is extremely high in the aggregation strategies without detailed intra-domain connectivity in the logarithmic growth phase. When the Single Node is used, only 1 in 10 attempts will result in an actual path. Even with the Star aggregation, which shows some intra-domain details, over 75% of the attempts is a false positive result. This clearly shows that once the network resources becomes less and less available, detailed knowledge of intra-domain connectivity is required in order to provide accurate results for inter-domain pathfinding .

False Negatives We have also examined the false negatives in the aggregation strategies. To examine this, we have slightly adjusted our experiments. At first we perform a normal run on the full graph, but now we record which pairs form a successful path. Then we use only these pairs in the aggregated views of the graph. Figure 6.9 shows the results, the diagonal describes perfect performance. The difference between the diagonal and each of the aggregation methods is the number of false negatives. In this case the plots were created by using non-linear least squares fitting with the following function:

$$\#\mathit{SuccessfulPaths} = A \cdot \mathit{index} + B \cdot \mathit{index}^2 \tag{6.5}$$

The graph in figure 6.9 shows that initially the Full Mesh strategy shows very little false negatives. This number increases as more and more paths are reserved, even up to the point where it is finally overtaken by the other aggregation strategies.

In figure 6.10 we show a box-plot of the path lengths for the aggregation strategies with only the success pairs. There we can see that the paths in Full Mesh are significantly longer than in the other strategies, and they are also slightly longer than in the control. These longer paths take up more resources, and therefore make it less likely that successive paths will succeed. In the other strategies the successful paths are somewhat shorter. The initial longer paths fail, keeping more resources available so that the later shorter paths will succeed.

6.5 Discussion and Conclusion

In this chapter we have examined what kind of impact aggregation has on the performance of inter-domain pathfinding. We have described different aggregation methods, and performed experiments to test and quantify this impact.

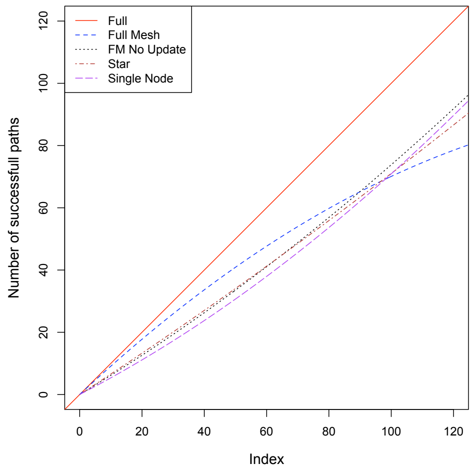


Figure 6.9: *The fitted functions describing the sum of successful path requests*

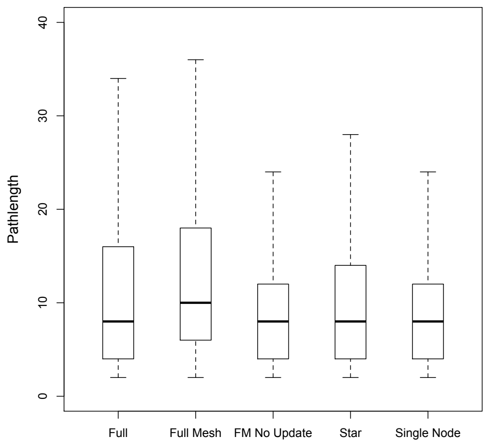


Figure 6.10: *A box plot for the lengths of only the paths that succeeded in the full view*

Strategy	**variable**	**value**	σ	$\mathbf{R^2}$
Full Mesh	A	0.934	0.004	0.98
	B	0.00233	0.00005	
FM no updates	A	0.603	0.004	0.98
	B	0.00134	0.00004	
Star	A	0.652	0.004	0.98
	B	0.00057	0.00005	
Single Node	A	0.517	0.004	0.98
	B	0.00191	0.00004	

Table 6.4: *Fitted values and their error for false negatives in* $(d = 150, n = 5)$

Our analysis in the previous section clearly show that aggregation does indeed have an impact on the performance of inter-domain pathfinding. In the initial linear growth phase the Full Mesh aggregation strategy performs close to the Full View. The other aggregation strategies perform significantly worse.

This difference in performance becomes much smaller in the logarithmic growth phase, where the growth of inter-domain resource usage in all aggregation strategies is very similar. However in this phase the number of false positives in the aggregation methods without accurate intra-domain connectivity is extremely high. This means that once the network becomes reasonably filled, these aggregation strategies become almost unusable without a way of filtering out these false positives.

A possible way of using the information from false positives is by using crank-backs. This method of locally updating the view of the topology using false-positive information ultimately creates a similar view on the graph as the Full Mesh method does. The difference is that with crank-back the majority of the effort of creating the updated graph lies with the requester. The resulting graph is also not shared with the other domains. The load of pathfinding is then shifted from the domains, performing less updates, to the source, which has to perform the crank-backs. We have seen from the false-positive results that clients will very often have to perform these crank-backs.

An argument that is often used in favour of aggregation is scaling: finding paths in large detailed graphs takes more time than finding a path in an aggregated graph. This argument fails to take the cost of constructing and updating the aggregated graph into account. In the case of the Full Mesh graph with updates, the cost of maintaining the graph is distributed over all the domains. The total distributed processing time is then higher than finding a path in a full detailed graph. In the Star, Simple Node, or Full Mesh without it is less hard to maintain the aggregated topologies, but these views show a very large number of false-positives. To get a reasonable performance in these strategies, crank-backs will have to be used, which will be very time consuming.

It is somewhat difficult to compare our results to the results of Awerbuch et al. since they use crank-backs. However, the general performance trends in their results are similar to ours. Full Mesh ('Complete' in their terminology) performs best, while both their Star aggregation methods show a slightly worse performance.

Comparing our results to the results of Guo and Matta, we see a significant difference in the performance of the Star aggregation. In their study the Star performs almost equally with the Full Mesh aggregation, while both in Awerbuch et al. and our results the Star aggregation performs significantly worse. Unfortunately, we cannot reproduce their results, even when using their topology we see a significant difference between the Full Mesh and Star aggregations. Our results are shown in figure 6.11 and fitted values are in table 6.5. The high inter-domain connectivity in their topology makes the network saturation occur very late in our results, this is why we only use the linear fit.

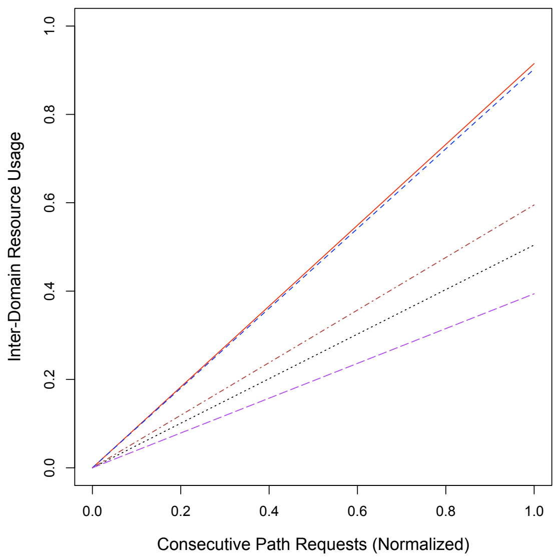


Figure 6.11: *The performance of the different aggregation methods on the topology of Guo and Matta.*

Strategy	**A**	σ	$\mathbf{R^2}$
Full	0.915	0.001	0.99
Full Mesh	0.902	0.001	0.99
FM no updates	0.504	0.001	0.98
Star	0.595	0.001	0.98
Single Node	0.393	0.001	0.98

Table 6.5: *Fitted values and their error for the topology of Guo and Matta*

Chapter 7

Summary and Conclusion

The provisioning of an inter-domain lightpath is currently still a complicated process. The user must formulate a request to the provider. The provider then checks availability in the network, and must figure out a path through the inter-domain network. Each of the domains then check their availability in turn, and if the destination is reachable, the lightpath is provisioned. Once the lightpath is working, it is delivered to the end user.

The different parties involved in the provisioning process mostly communicate ad hoc through telephone and email. This allows many opportunities for mistakes, or misunderstanding, complicating the provisioning and the monitoring processes. This thesis provides contributions towards facilitating the inter-domain network provisioning process. We have divided this in two parts, first creating a clear and inter-operable description of the network, and second to examine the impact of aggregating topology information on the performance of pathfinding.

In the first part of this thesis we have examined the problem of describing computer networks. In chapter 2 we have described requirements for creating interoperable descriptions of computer networks. We have used these requirements to evaluate existing ways of representing computer networks. None of these representations satisfy all of the requirements.

In chapter 3 we have presented our Network Description Language (NDL). The beginning of the chapter describes the first version of NDL, which provides a simple ontology to describe network topologies. While the vocabulary is small, it is already very powerful.

Based on our experiences with the first version of NDL and based on related work on GMPLS, we have extended NDL to describe multi-layer networks. After extensive study of the ITU-T G.805 standard we have concluded that it is possible to create a technology-independent description of multi-layer networks with NDL. Inspired by the theory in ITU-T G.805 we have created multi-layer NDL. Multi-layer NDL provides a vocabulary to independently describe technologies. These technology descriptions can then be referenced from multi-layer topology descriptions.

In chapter 4 we have shown several of our applications that show new possibilities with the use of NDL. We have gained valuable experience and feedback from our development of applications and tools to support and manage real-world networks using NDL. Our choice for RDF was first put to the test when we developed generators and validators. Descriptions in RDF can be very verbose, but our generators and validators proved to be simple and adequate solutions. Furthermore, our Python NDL Toolkit (pynt) has lowered the boundary of developing applications using NDL. The toolkit has also allowed us to rapidly prototype ideas and has been a valuable input for the continued improvement of NDL.

In summary, NDL provides a distributed information model for the description of topologies for inter-domain pathfinding. So this allows us to give a positive answer to the first research question: *It is possible to create a distributed information model for the description of topologies for inter-domain pathfinding.*

Network operators do not always wish to share their full topology, either for security, business, or for scalability reasons. On the other hand it is necessary to share some degree of topology data to enable inter-domain lightpath planning. A solution is to publish aggregated versions of the network topologies. In the second part of this thesis we have examined the impact of aggregation on the performance of inter-domain pathfinding.

In chapter 5 we have described different methods for aggregating domain topologies, the Full Mesh, Star and Single Node strategies. We have summarized three earlier publications on pathfinding in aggregated topologies. Unfortunately these studies use different measures for performance, and are therefore hard to compare. Even when comparing the general trends, they show some conflicting results on the performance of the Star strategy.

In order to better quantify the performance impact of topology aggregation in optical networks, we have performed our own experiments. Chapter 6 describes our experimental set up, and shows the results of putting different strategies to the test. This has allowed us to accurately determine the impact of different aggregation strategies on inter-domain pathfinding.

Based on our results we can give an affirmative answer to the second research question: *topology aggregation does indeed have an impact on pathfinding, albeit not a great impact.*

If the Full Mesh strategy is used, and the topology description is often updated to reflect the current availability, then there is almost no difference in performance compared to having all the available information. Of course, a Full Mesh aggregation that is often updated maintains all the information relevant for inter-domain pathfinding. If no updates are performed, then the Star aggregation should be used. The pathfinding will suffer an impact from the aggregation, but the Star strategy performs better than a Full Mesh without updates. The Single Node strategy performs worst, even worse than the Full Mesh without updates. The experiments and our analysis of the results show the way that topology aggregation has an impact on inter-domain pathfinding, providing an answer to the second research question.

7.1 The Road Ahead

7.1.1 RDF Infrastructure Descriptions

In our view the next step in network descriptions is the definition of ontologies that can cover the whole end-to-end infrastructure: from the actual content being distributed to the Storage elements holding the data to the CPUs rendering the images and the display to visualize it. Figure 7.1 illustrates the concept of RDF representation of all this elements forming the overall infrastructure.

Our vision is that a media content locator will be able to consume the descriptions of all the architectural components that form the end-to-end infrastructure. From this information it will build the optimal paths from the storage elements to the visualization displays, making the inter-domain lightpath provisioning a piece in the overall orchestrated effort. The use of RDF allows easy extension of the topology and domain schema with other schemata.

Our vision is that an application will be able to consume the descriptions of all the architectural components that form an end-to-end infrastructure. This information includes computing resources, storage resources, visualization resources, network resources, content descriptions, et cetera. All resources can be linked with loose couplings to allow a meta-scheduler application to orchestrate all resources together in a combined effort [7].

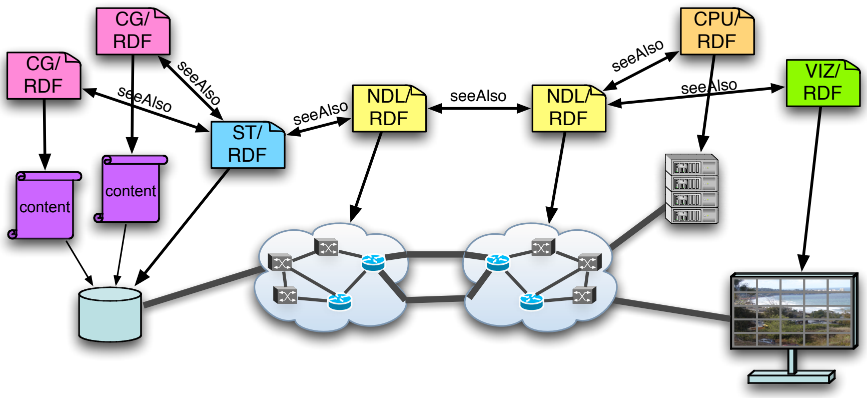


Figure 7.1: *RDF Infrastructure Descriptions*

7.1.2 Topology Aggregation

In our experiments we have not used a crank-back algorithm, however we expect it to give similar results as the Full Mesh view with updates. The crank-back algorithm provides a way of using information about false-positive failures to do recalculation. Effectively this allows a requester to get the same level of information for path computation as with constant updates. The difference is that the main burden of computation is then moved from the domain to the client. Another point for future research is inserting a delay in the propagation of domain updates, so that we can confirm earlier results of Awerbuch et al.

Another open issue is mixed aggregation strategies, our experiments used the same aggregation strategy for all domains. In practice different domains will make different choices for topology aggregation. It could be investigated what effect this will have on the performance of the inter-domain network pathfinding.

Multi-layer pathfinding in optical networks is currently a topic of research[70]. For now, this pathfinding is based on full topology information, since this is a very hard problem in itself. An open issue is defining aggregation strategies for multi-layer topologies. It is far from trivial to apply aggregation to multi-layer topologies. Besides the connectivity information that is aggregated, the encoding and the adaptation capabilities must also be considered.

Appendix A

Translation Specification of OSPFv2 LSAs to NDL

A.1 Introduction

The OSPF protocol is one of the main protocols in use in routers today. It defines a way for routers to discover and communicate with each other, in order to collectively and simultaneously learn the network topology. The routing algorithms in OSPF require that each of the routers maintain a database of the full topology.

Messages between routers are exchanged using OSPF packets. OSPF version 2 defines five different kinds of packets.

1. Hello
2. Database Description
3. Link State Request
4. Link State Update
5. Link State Acknowledgment
6. Opaque Link State Announcements

The first two packets are used in the discovery and first synchronization process. When a new router joins the OSPF topology it sends out a Hello discovery packet. To kick start the joining router an adjacent router sends over his complete database using the Database Description packet.

The next three are used to request and send updated information about the topology. The topology information itself is represented in a structure called a Link State Announcement (LSA). In this appendix we define how we translate all the information in the LSAs to the NDL syntax. The other data in the OSPF packets is only used to ensure correct delivery of the LSAs. We assume that the LSAs we receive have been delivered correctly, so we ignore the other data.

There are different types of LSAs:

1. Router LSA
2. Network LSA
3. Summary LSA (IP Network)
4. Summary LSA (Autonomous System Boundary Router)
5. Autonomous System External LSA

In the rest of this appendix we first examine the generic header of LSAs, and then successively examine each type of LSA.

A.2 Generic LSA Header

The generic LSA Header structure is shown in figure A.1. The diagram shows the fields of the header and their lengths in bits, using lines of 32 bits length.

The first field (LS Age) is an integer value that, together with the LS sequence number, is used to ensure that topology data eventually goes 'stale' and is removed. The Options field is used to relay optional capabilities of routers[1], these are not relevant to the topology, so we ignore them.

The last two fields (LS Checksum, and length) are used to ensure correct delivery of an LSA.

The value of LS type is used to distinguish between the different types of LSAs. The type of the LSA also impacts the interpretation of the Link State ID and the Advertising Router fields.

[1]The options field contains the following bits: `E` (flooding of AS-External-LSAs), `MC` (forwarding of multicast), `N/P` (Handling of type 7 LSAs), `EA` (handling External Attribute LSAs), `DC` (handling demand circuits).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            LS age             |    Options    |    LS type    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Link State ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Advertising Router                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     LS sequence number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         LS checksum           |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.1: *The generic LSA Header*

A.3 LSA Type 1: Router LSAs

Each router describes his links to an area using a single router LSA. The LSA describes the status and metric cost of the interfaces of the router.

The first type of LSAs are Router LSAs. The values of `Link State ID` and `Advertising Router` are in this case identical and set to the sending Router's ID. Following the LSA Header the content of a Router LSA is shown in figure A.2.

The first line contains three bit-flags, surrounded by padding. The bits represent properties of the router sending the LSA:

`V` Virtual link endpoint; The router is an endpoint of virtual links, which have this area as Transit area.

`E` External; This bit is set when the advertising router is an AS boundary router.

`B` Border; The advertising router is an area border router when this bit is set.

In summary these bits are only relevant for distributing routes through different areas, and are not directly related to topology. The flag bits is followed by the specification of the number of links present in the LSA. Each link is of a type between one and four, and the meaning of each field depends on the type:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    0    |V|E|B|        0      |            # links            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Link ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Link Data                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     # TOS     |            metric             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      TOS      |        0      |          TOS  metric          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Link ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Link Data                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
```

Figure A.2: *The Router LSA body structure*

1. Point-to-Point connection to another router (unnumbered)

   `Link ID`: Neighboring router's Router ID

   `Link Data`: The interface's MIB-II ifIndex value

2. Connection to a transit network

   `Link ID`: IP Address of the Designated Router

   `Link Data`: The router interface's IP address

3. Connection to a stub network

   `Link ID`: IP network/subnet number

   `Link Data`: The IP network/subnet mask

4. Virtual Link

   `Link ID`: Neighboring router's Router ID

   `Link Data`: Router interface's IP address

Each Link also carries a metric section, which specifies the link metric (`metric`), and the number of additional metrics (`# TOS`). Each TOS is defined on a separate line containing the IP Type of the service (`TOS`), followed by a padding byte and the TOS-specific metric information. The TOS metrics are not used widely, they are only included for backwards compatibility with OSPFv1.

A.3.1 Translation to NDL

From a Router LSA we can extract the following topological information:

- There is a device R with name `dev + Advertising Router`. We add the prefix here to avoid the clash with the interface of the router with that IP-address.
- For each link segment we can conclude that the router R *hasInterface* I with metric `metric`.
- The name of the interface I depends on the link type value:
  1. `p + Advertising Router + p + Link ID` This is an unnumbered interface, and is *connectedTo* an interface I' called `p + Link ID + p + Advertising Router`.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Network Mask                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Attached Router                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                                |
```

Figure A.3: *The body structure of a Network LSA*

2. `Link Data` *connectedTo* a broadcast segment BC called `bc + Link ID`.
3. `stub + Advertising Router + net + Link Id` The interface is *connectedTo* a broadcast segment BC called `stub + Link ID`.
4. `Link Data` *connectedTo* interface I' of device R', called `dev + Link ID`. The name of interface I' can only be determined once we receive a matching Link description of R'.

Note that we have ignored the TOS metric. While the TOS metric fields are still present in OSPFv2 LSAs, they are only supported for backward compatibility and are never used in practice.

A.4 LSA Type 2: Network LSAs

Network LSAs are originated for broadcast and NBMA networks in areas where there are two or more routers. The Network LSA is sent by the designated router for the area. The `Link State ID` is in this case set to the IP address of the DR's interface in the network, and the `Advertising Router` is the DR's router ID. Following the LSA header, the Network LSA is rather simple, as shown in figure A.3.

The `Network Mask` contains the mask for the network, after which the IDs of the attached routers are listed, including the DR.

A.4.1 Translation to NDL

From a Network LSA we can extract the following topological information:

- There is a router R with router ID `Advertising Router`, which we call `dev + Advertising Router`.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Network Mask                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      0        |                  metric                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     TOS       |                TOS  metric                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
```

Figure A.4: *The body structure of a Summary LSA*

- *R hasInterface I*, which is called `Link State ID`. This interface is connected to the broadcast segment *BC* called `bc + Link State ID`.
- For each of the values of `Attached Router` we can conclude:
  - There is a device R' called `dev + Attached Router`.
  - R' has also sent out a Type 1 LSA, listing (an) interface(s) I', which is already *connectedTo* broadcast segment *BC*.
  - For each of the above interfaces I' of R' we add a *connectedTo* from the broadcast segment *BC* to that interface.

A.5 LSA Type 3 & 4: Summary LSAs

LSA Types 3 and 4 are summary LSAs. These LSAs are originated by area border routers and are used to distribute network routes to destinations outside the area. Note that this means that these LSAs are only sent when the OSPF network contains multiple areas. Type 3 LSAs are used for routes to other areas , while type 4 are used for AS external routes. An example is a default route to an AS border router, which means that any traffic for which no other route is defined is routed to that AS border router.

The only difference in the summary LSAs is in the meaning of the `LSA Link ID`, for type 3 this contains the IP network number, while in type 4 this contains the AS boundary router's Router ID. The `Advertising Router` field is always the area's border router. The structure of the summary LSAs after the generic header is shown in figure A.4.

The `Network Mask` contains the mask of the route that is being advertised, and the `metric`, `TOS`, and `TOS metric` are of the same format as in type 2 LSAs.

A.5.1 Translation to NDL

From a type 3 Summary LSA we can extract the following topological information:

- There is a device R with router ID `Advertising Router`, which we call `dev + Advertising Router`.
- R *hasInterface* I called `Advertising Router + if + LSA Link ID`. This interface has a *metric* of `metric`.
- Interface I is *connectedTo* a Network Domain ND called `nd + LSA Link ID`.

From a type 4 Summary LSA we can extract the following:

- There is a device R named `dev + Advertising Router`, and a device R' named `dev + LSA Link ID`.
- R is connected through an abstracted link with R', with interface I .
- I has a *metric* of `metric`.

Note that we have ignored the TOS metric. While the TOS metric fields are still present in OSPFv2 LSAs, they are onlly supported for backward compatibility and are never used in practice.

A.6 LSA Type 5: AS External LSAs

The last type of OSPF v2 LSAs are AS External LSAs. These LSAs are originated by AS boundary routers to describe destinations external to the AS. Contrary to other LSAs the AS External LSAs are flooded over the entire routing domain. When multiple areas are used, the information from a type 5 LSA is not sufficient, because routers in other areas will not know how to reach the boundary router. In this case the type 4 LSAs are used to distribute additional reachability information.

The `Advertising Router` is the router ID of the AS boundary router, and the `Link State ID` is used to describe the destination network number, or `0.0.0.0` for the default route. The structure of the AS External LSA after the header is shown in figure A.5

The fields have the following meaning:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Network Mask                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|     0       |                  metric                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Forwarding address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      External Route Tag                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|    TOS      |                TOS  metric                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Forwarding address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      External Route Tag                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
```

Figure A.5: *The body structure of an AS External LSA*

`Network Mask` The network mask for the destination network or `0.0.0.0` in case of the default route.

`E` This flag signals whether the metric is a Type 1 (off) or 2 (on) external metric. In case of type 2, the metric is always considered to be higher than any internal metric.

`metric` The value of the metric.

`Forwarding Address` Data for the described destination should be forwarded to this address. If it is set to `0.0.0.0`, use the address `Advertising Router`.

`External Route Tag` This field is not used by the OSPF protocol itself, but is used for communication between AS boundary routers.

`TOS` For each TOS value, different destinations can be defined, using the similar fields as above.

A.6.1 Translation to NDL

From a type 5 AS External LSA we can extract the following topological information:

- There is a device R with Router ID `Advertising Router`, which we call `dev + Advertising Router`.
- If `Forwarding Address` equals `0.0.0.0`, the router is in the same area, and we already know how R is connected to the rest of the network.
- R *hasInterface* I with an abstract connection to the interface I' `Forwarding Address` a router R'.
- A Network Domain ND called `nd + Link State ID`.
- I has a *metric* of `metric`.

Note that we have ignored the TOS metric. While the TOS metric fields are still present in OSPFv2 LSAs, they are only supported for backward compatibility and are never used in practice.

Appendix B

Translation Specification of OSPFv2 Traffic Engineering LSAs to NDL

B.1 Introduction

OSPFv2 has been extended with traffic engineering options, called OSPF-TE [100]. Together with RSVP-TE, this forms the basis for an implementation of MPLS-TE [101] and is also used in GMPLS. The OSPF TE extension is implemented using a generic extension method of OSPFv2, the Opaque LSA [102]. There are three types of Opaque LSAs, defined by their flooding scope:

- Link-state type 9 denotes a link-local scope. Type 9 Opaque LSAs are not flooded beyond the local (sub)network.
- Link-state type 10 denotes an area-local scope. Type 10 Opaque LSAs are not flooded beyond the borders of their associated area.
- Link-state type 11 denotes that the LSA is flooded throughout the Autonomous System (AS). The flooding scope of type 11 LSAs are equivalent to the flooding scope of AS-external (type 5) LSAs.

In this appendix we will only describe type 10 (area-local) LSAs. Together with type 9, these are widely used in practice. Type 9 LSAs are only used for direct communication between adjacent routers, they do not add topological information. Currently, type 11 LSAs are not used.

We will first describe the format of type 10 LSAs, followed by what kind of information values are transported through them. Finally we will describe how to translate these values into NDL. The structure of the Opaque LSAs is shown in figure B.1

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            LS age             |     Options   |   10 or 11 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Opaque Type  |               Opaque ID                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Advertising Router                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      LS Sequence Number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         LS checksum           |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                      Opaque Information                       |
+                                                               +
|                              ...                              |
```

Figure B.1: *The structure of an Opaque LSA*

The only difference with the regular OSPF header is that the `Link ID` field has been replaced with `Opaque Type` and `Opaque ID`. The `Opaque Type` is an 8-bit integer value to describe the type of the Opaque LSA, values 0-127 must be registered, and values 128-255 are available for experimental and private use. The `Opaque ID` is a 24-bit integer type-specific ID value.

For almost all TE extensions type 10 LSAs are used, with `Opaque Type` 1, the `Opaque ID` value is an arbitrary value used to maintain multiple TE LSAs. The body of these LSAs (`Opaque Information`) are structured using type-length-value elements (TLVs).

As the name suggests, a TLV consists of a `Type` and a `Length` field (both 16-bit integers), followed by a `Value` field of `Length` octets long. Note that the value of a TLV can also be other TLVs, these are then called `sub-TLVs`

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Value...                           |
.                                                               .
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B.2: *The Type-Length-Value structure*

Values often define a certain kind of bandwidth. These are all expressed in *bytes* per second using the standard IEEE floating point notation:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|    Exponent   |                  Fraction                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

`S` is the sign, `Exponent` is the exponent base 2 in 'excess 127' notation, and `Fraction` is the mantissa - 1, with an implied binary point in front of it. Thus, the above represents the value: $(-1)^{\texttt{S}} * 2^{\texttt{Exponent}-127} * (1 + \texttt{Fraction})$

B.2 Area-Local Opaque LSAs

In this section we describe the two currently defined TLVs for area-local opaque LSAs (type 10), and their sub-TLVs as defined by RFC 3630[100], and 4203[103].

1. `Router Address` This contains a stable IP address for advertising router, on which it can always be reached on the control plane.

2. `Link` This TLV describes a single link, and contains a set of sub-TLVs, described in the list below.

Below we list the first 10 sub-TLVs (1–9 and 11, 10 is not assigned). The sub-TLVs 14, 15, and 16 are described separately, because they are not that straightforward (12 and 13 are also unassigned).

1. `Link Type` (1 octet) The value is either `1` (point-to-point) or `2` (multi-access). This sub-TLV is mandatory.
2. `Link ID` (4 octets) An identifier for the other end of the link. For point-to-point it is the router ID of the neighbor, for multi-access it is the interface address of the designated router (i.e. the `Link State ID` of the Router LSA). This sub-TLV is mandatory.
3. `Local Interface IP Address` (4N octets) The address(es) of the local interface of this link, N is the number of addresses.
4. `Remote Interface IP Address` (4N octets) The address(es) of the remote interface of this link. If it is multi-access, the value is either `0.0.0.0` or the router may choose not to send this sub-TLV.
5. `Traffic Engineering Metric` (4 octets) The TE metric of the link, this may be different from the standard OSPF metric.
6. `Maximum Bandwidth` (4 octets) The true capacity of the link.
7. `Maximum Reservable Bandwidth` (4 octets) The maximum reservable capacity in this direction. This may be greater than the true capacity.
8. `Unreserved Bandwidth` (32 octets) The amount of bandwidth available for reservation in each of the eight priority levels, starting with 0. Each value must be less than or equal to the maximum reservable bandwidth.
9. `Administrative Group` (4 octets) A bit mask assigned by the administrator. Each set bit corresponds to a group that the interface belongs to. This starts at group 0, and is also called 'Resource Class' or 'Color'.

11 `Link Local/Remote Identifier` (8 octets) GMPLS also supports unnumbered links, but these have to be identified in some way. This TLV contains the local and the remote identifiers for the endpoints of this unnumbered link.

An additional sub-TLV is the `Link Protection Type` sub-TLV (type 14). It has a length of 4 octets, but only the first of the octets is currently used. The meaning of the possible values of the first octet is as follows:

`0x01` *Extra Traffic*, this link is protecting another link or links, and LSPs on this link will be lost if any of these fail,

`0x02` *Unprotected*, there is no protection for this link, and LSPs will be lost if the link fails,

`0x04` *Shared*, there are one or more links of type *Extra Traffic* protecting this link, however, these are shared between one or more links of type *Shared*,

`0x08` *Dedicated 1:1*, there is one dedicated link of type *Extra Traffic* protecting this link,

`0x10` *Dedicated 1+1*, there is one dedicated link protecting this link, which is not advertised.

`0x20` *Enhanced*, this link is protected by a scheme better than *Dedicated 1+1*, for example by a 4 fiber ring BLSR.

Another sub-TLV that is relevant to link protection is the sub-TLV `Shared Risk Link Group` (type 16). This sub-TLV has a length of 4N octets, where N is the number of groups this link belongs to. A shared risk link group (SRLG) is a group of links that share a resource whose failure may affect all links in the group, for example two fibers running in the same conduit. An SRLG is identified by a 32 bit number, that is unique within the domain.

The purpose of the SRLG identification is to allow requests for multiple diversely routed LSPs, that also do not share any SRLGs, so as to minimize the risk of failure.

B.2.1 Translation to NDL

In this section we describe how we translate the information in Opaque LSAs to NDL. We use letters to denote objects. In NDL these objects are identified using URIs, these names are built up using the URI of the document or namespace, a pound sign (#), followed by the name of the object. Below we reference to names using only the latter part of the URI. We reference NDL properties using *italics*, the exact meaning of these properties can be found in the NDL papers [1, 3], or the NDL Homepage [74].

From the header of any Opaque LSA we learn:

- There is a device R, named `dev + Advertising Router`
- Router R *hasInterface* I named `Advertising Router`.

Depending on the type of TLV in the Opaque LSA we can learn additional information. For example from a type 1 TLV (Router Address) we can learn only one simple fact:

- Router R *hasInterface* I named $RouterAddress$.

On the other hand, a type 2 TLV carries a lot more information:

- Device R *hasInterface* I,
- If the value of `Link Type` is `1` (point-to-point):
  - There is a link L, named `Link ID`,
  - There is an interface I' *connectedTo* L,
  - Interface I is *connectedTo* L,
  - If the sub-TLVs `Local Interface IP Address` and `Remote Interface IP Address` are defined:
    - Interface I has the address(es) `Local Interface IP Address`, if the interface I was not named yet, then the first address is used as name,
    - Interface I' has the address(es) `Remote Interface IP Address`, if the interface I' was not named yet, then the first address is used as name,
  - Otherwise, the link is unnumbered, and the sub-TLV `Link Local/Remote Identifiers` must be present:
    - I is named `Link Local Identifier`,
    - I' is named `Link Remote Identifier`,
- If the value of `Link Type` is `2` (multi-access):
  - There is a broadcast segment BC named `bc + Link ID`,
  - Interface I is *connectedTo* link L,
  - Link L is `switchedTo` broadcast segment BC,
  - Interface I has the address `Local Interface IP Address`, if the interface I was not named yet, then the first address is used as name,

- Link L has a *metric* of `Traffic Engineering Metric`,
- Link L has a *capacity* of `Maximum Bandwidth`,
- Link L has a *protectionType* of `Link Protection Type`,
- Link L has a *sharedRiskGroup* property with value `Shared Risk Link Groups`.

Currently NDL does not yet support the concept of reservable and unreserved bandwidth. The reason for this is that the reservation information is more dynamic than the network topology. Our idea is that the best way to provide the user with up to date information is to have a pointer to a certain service, where the information about the reservable bandwidth of links can be obtained. We currently also do not translate the administrative groups. We currently have no experience what the value is used for in practice.

Note that it is possible to simplify this somewhat and use a single *connectedTo* statement between the two interfaces. However, it is then not possible to use the *protectionType* and *sharedRiskGroup* properties.

B.3 Switching Capability

The last sub-TLV that we describe is also the most complex; which is why we dedicate a separate section to it: the `Interface Switching Capability Descriptor` (ISCD)(type 15), which has a variable length. The purpose of this TLV is to describe the switching capabilities of both interface in that link of the advertising router, as well as the switching capabilities of the routers' switching matrix. The format of this sub-TLV is described in figure B.3.

The values of the `Switching Capability` and `Encoding` fields are the same as used in the request signalling [65]. The `Switching Capability` (Switching Cap) field contains one of the following values:

1 – Packet-Switch Capable-1 (PSC-1),

2 – Packet-Switch Capable-2 (PSC-2),

3 – Packet-Switch Capable-3 (PSC-3),

4 – Packet-Switch Capable-4 (PSC-4),

51 – Layer-2 Switch Capable (L2SC),

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Switching Cap |   Encoding    |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Max LSP Bandwidth at priority 0              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Max LSP Bandwidth at priority 1              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Max LSP Bandwidth at priority 2              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Max LSP Bandwidth at priority 3              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Max LSP Bandwidth at priority 4              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Max LSP Bandwidth at priority 5              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Max LSP Bandwidth at priority 6              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Max LSP Bandwidth at priority 7              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Switching Capability-specific information              |
|                  (variable)                                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B.3: *The structure of an Interface Switching Capability Descriptor*

100 – Time-Division-Multiplex Capable (TDM),

150 – Lambda-Switch Capable (LSC),

200 – Fiber-Switch Capable (FSC).

The four PSC values are used to express hierarchy of LSPs tunneled within LSPs.

The `Encoding` field is an integer field, where the value means that the link has the following encoding type:

1 – Packet

2 – Ethernet

3 – ANSI/ETSI PDH

5 – SDH ITU-T G.707 / SONET ANSI T1.105

6 – Digital Wrapper

7 – Lambda (photonic)

8 – Fiber

9 – FiberChannel

For each of the eight priority levels, the sub-TLV gives the maximum bandwidth this link can support for LSPs. Contrary to the `Maximum Bandwidth` value, these values are designed to be dynamic. In the future these bandwidth specifications will replace the `Maximum Bandwidth` sub-TLV described earlier. For backward compatibility the `Maximum Bandwidth` value may be set to the priority 7 bandwidth.

The last section of the sub-TLV contains information specific to the switching capability value:

`Packet-Switch Capable` The specific information for PSC switching capabilities is structured as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Minimum LSP Bandwidth                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Interface MTU       |            Padding            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The `Minimum LSP Bandwidth` specifies the minimum bandwidth that an LSP must request. The supported LSP bandwidths depend on the encoding type. The `Interface MTU` specifies the largest size packets that are supported by this interface.

`Layer-2 Switch Capable` does not carry any extra information.

`Time-Division-Multiplex Capable` The specific information for TDM switching capabilities uses the following structure:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Minimum LSP Bandwidth                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Indication  |                 Padding                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The `Minimum LSP Bandwidth` specifies the minimum bandwidth that an LSP must request. The `Indication` contains an integer value used to indicate whether the interface supports Standard SONET/SDH (`0`), or Arbitrary SONET/SDH (`1`).

`Lambda-Switch Capable` does not carry any extra information.

The `Interface Switching Capability Descriptor` sub-TLV may occur multiple times, for example to express that an interface supports different encodings.

B.3.1 Translation to NDL

The `Interface Switching Capability Descriptor` is very complex to translate. The ISCD describes capabilities as part of the link, and the interface, while NDL describes this as part of the device. The interpretation of the values in the ISCD also depend on the values of the related ISCD of the interface on the other side of the link, which may have different values[1]. Therefore the translation below uses both the local ISCD (describing interface I, of router R) and the remote ISCD (interface I' and R'). The values of the remote ISCD are referenced with an accent (e.g. `Encoding'`).

It is possible that the two ISCDs of a link carry different encoding values for each side. This implicitly signals that one of the two interfaces is able to do an adaptation from its encoding to the (lower) other encoding. A lower encoding layer means a higher `Encoding` value.

NDL expresses adaptations using interface objects at different layers, therefore we introduce interfaces with a layer suffix, and define the relations between them. Once the common layer between the two interfaces is identified, we define

[1] We assume that the link is bidirectional.

an equality with the interfaces at that layer and the original interfaces, so that the connection is described on the right layer, with all adaptations.

Switching Cap	**Encoding**	**NDL Layer**
Packet-[1-4]	Packet	IP
Layer-2	Ethernet	Ethernet
TDM	ANSI PDH SONET/SDH	TDM
–	Digital Wrapper	?
Lambda	Lambda	Lambda
Fiber	Fiber	Fiber
–	FiberChannel	?

Table B.1: *The relation between the different layer definitions.*

The relation between `Encoding`, `Switching Capability` and NDL Layers is shown in Table B.1[2]

The translation of these interfaces and layers is as follows:

- There is an *interface* $I_{\texttt{Encoding}}$, named `Local Interface + Encoding`,
- Interface $I_{\texttt{Encoding}}$ is at NDL layer `Encoding`,
- There is an *interface* $I'_{\texttt{Encoding}'}$, named `Local Interface' + Encoding'`,
- Interface $I'_{\texttt{Encoding}'}$ is at NDL layer `Encoding'`,

Next we compare the `Encoding` values and introduce the relevant interfaces at the right layers, and define the proper equalities:

- If `Encoding` $>$ `Encoding'`, then:
  - There is an *interface* $I_{\texttt{Encoding}'}$, which is at NDL layer `Encoding'`,
  - There is an *adaptation* $Adap(\texttt{Encoding}, \texttt{Encoding}')$ between interface $I_{\texttt{Encoding}}$, and $I_{\texttt{Encoding}'}$,
  - Interface $I_{\texttt{Encoding}'}$ and I are defined to be equal,
  - Interface $I'_{\texttt{Encoding}'}$ and I' are defined to be equal,

[2]The table shows a merry mixture of layer names, technologies, and protocols

- If `Encoding` $<$ `Encoding'`, then:
  - There is an *interface* $I'_{\texttt{Encoding}}$, which is at NDL layer `Encoding`,
  - There is an *adaptation* $Adap(\texttt{Encoding}', \texttt{Encoding})$ between interface $I'_{\texttt{Encoding}'}$, and $I'_{\texttt{Encoding}}$,
  - Interface $I'_{\texttt{Encoding}}$ and I' are defined to be equal,
  - Interface $I_{\texttt{Encoding}}$ and I are defined to be equal,
- If `Encoding` $=$ `Encoding'`, then:
  - There is an *interface* $I_{\texttt{Encoding}}$, which is at NDL layer `Encoding`,
  - There is an *interface* $I'_{\texttt{Encoding}}$, which is at NDL layer `Encoding`,
  - Interface I and $I_{\texttt{Encoding}}$ are defined to be equal,
  - Interface $I'_{\texttt{Encoding}}$ and I' are defined to be equal,

The actual switching is done by the device, these translations do not depend on the other ISCD, so we define them for the general case:

- There is a *switchMatrix* SM, which is at NDL layer `Switching Capability`,
- Device R *hasSwitchMatrix* SM,
- The switching-matrix SM *hasInterface* $I_{\texttt{SwitchingCapability}}$,
- Depending on the difference between the layers of `Encoding` and `Switching Capability`, introduce interfaces and adaptations as above, however, no equalities need to be defined.

NDL currently does not have a way to express any of the information in the switching capability-specific fields.

The exact adaptation functions used to go from one GMPLS layer to the other in NDL are currently still an open issue. OSPF-TE simply does not provide enough information to deduce the exact behavior of the devices.

Appendix C

List of Abbreviations

ANSI American National Standards Institute

ANSI/ETSI PDH Plesiochronous Digital Hierarchy (There are two interoperable versions of PDH, ratified by ANSI and ETSI)

ASN.1 Abstract Syntax Notation One

ATM Asynchronous Transfer Mode

BGP Border Gateway Protocol

CIM Common Information Model

CPU Central Processing Unit

DAS-3 Distributed ASCI Supercomputer 3

DMTF Distributed Management Task Force[38]

DRAGON Dynamic Resource Allocation over GMPLS Optical Networks

ETSI European Telecommunications Standard Institute

e-VLBI Very Long Baseline Interferometry

GLIF Global Lambda Integrated Facility[20]

GMPLS Generalized Multi-Protocol Label Switching

GOLE GLIF Open Lightpath Exchange

GPS Global Positioning System

IEEE Institute of Electrical and Electronics Engineers[104]

IETF Internet Engineering Task Force[105]

IP Internet Protocol

ITU-T Telecommunication Standardization Sector (coordinates standards on behalf of the ITU)

ITU International Telecommunication Union

LHC Large Hadron Collider

LSA Link State Announcement (Messages that are exchanged in OSPF)

MST Minimum Spanning Tree

MTU Maximum Transmission Unit (The largest data unit size that a data protocol (e.g. IP) can carry)

NDL Network Description Language

NEC network enabled capabilities

NM-WG Network Measurements Working Group

NREN national research and education network

OGF Open Grid Forum[106]

OSPF Open Shortest Path First

OSPF-TE Open Shortest Path First - Traffic Engineering (An extension of OSPF)

PNNI Private Network-to-Network Interface

pynt Python NDL Toolkit

RDF Resource Description Framework

RFC Request For Comments (an IETF memorandum on Internet systems and standards)

RST Random Spanning Tree

SDH Synchronous Digital Hiearachy

SNMP Simple Network Management Protocol

SONET Synchronous Optical Networking

SPARQL SPARQL Protocol and Query Language for RDF

SQL Structured Query Language

STP Spanning Tree protocol

STS Synchronous Transport Signal (Part of the SONET standard)

TCP Transmission Control Protocol

TDM Time-Division Multiplexing

TITAAN the Theatre Independent Tactical Army & Airforce Network

TL1 Transaction Language 1

UML User-Mode Linux

UML Unified Modeling Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

UTF-8 Unicode Transformation Format 8-bit

VLAN Virtual Local Area Network

VNE Virtual Network Experiments

WDM Wavelength-Division Multiplexing (A technology which multiplexes several wavelengths over the same optical fiber)

XML Extensible Markup Language

List of Author’s Publications

[1] Jeroen van der Ham, Freek Dijkstra, Franco Travostino, Hubertus Andree, and Cees de Laat: *Using RDF to Describe Networks. Future Generation Computer Systems, Feature topic iGrid 2005* (October 2006). http://staff.science.uva.nl/~vdham/research/publications/0510-NetworkDescriptionLanguage.pdf.

[2] Jeroen van der Ham, Paola Grosso, and Cees de Laat: *Semantics for Hybrid Networks Using the Network Description Language* (March 2006). Accepted as poster for SuperComputing 2006, http://staff.science.uva.nl/~vdham/research/publications/0603-NetworkDescriptionLanguage.pdf.

[3] Jeroen van der Ham, Paola Grosso, Ronald van der Pol, Andree Toonk, and Cees de Laat: *Using the Network Description Language in Optical Networks.* In *Tenth IFIP/IEEE Symposium on Integrated Network Management* (May 2007). http://staff.science.uva.nl/~vdham/research/publications/0606-UsingNDLInOpticalNetworks.pdf.

[4] Freek Dijkstra, Bert Andree, Karst Koymans, Jeroen van der Ham, and Cees de Laat: *A Multi-Layer Network Model Based on ITU-T G.805. Computer Networks* (June 2007). http://staff.science.uva.nl/~fdijkstr/publications/G805_Multilayer_Model.pdf.

[5] Jeroen van der Ham, Freek Dijkstra, Paola Grosso, Ronald van der Pol, Andree Toonk, and Cees de Laat: *A Distributed Topology Information System for Optical Networks Based on the Semantic Web. Optical Switching*

and Networking, 5(2-3):85–93 (June 2008). http://www.science.uva.nl/~vdham/research/publications/0703-ApplicationsOfNDL.pdf.

[6] Jeroen van der Ham and Gert Jan Verhoog: *Virtual environments for networking experiments*. Technical report, Master System- and Network Engineering (2004). http://staff.science.uva.nl/~vdham/research/publications/anp-jvdh-gjv.pdf.

[7] Paola Grosso, Jeroen van der Ham, Freek Dijkstra, and Cees de Laat: *Semantic Models for Optical Hybrid Networks – Lightpaths Across Domain Boundaries*. In Paul Cunningham and Miriam Cunningham (editors), *Proceedings of eChallenges 2007* (October 2007). http://www.science.uva.nl/~vdham/research/publications/0710-SemanticModels.pdf.

Bibliography

[8] Tiziana Ferrari, Jim Austin, Peter Clarke, Martyn Fletcher, Mark Gaynor, Richard Hughes-Jones, Tom Jackson, Gigi Karmous-Edwards, Peter Kunszt, Mark J. Leese, Jason Leigh, Paul D. Mealor, Inder Monga, Volker Sander, Ralph Spencer, Matt Strong, and Peter Tomsu: *Grid Network Services Use Cases from the e-Science Community.* OGF Grid Final Documents 122, Open Grid Forum (December 2007). `http://www.gridforum.org/documents/GFD.122.pdf`.

[9] Carola A. van Iersel, Harry J. de Koning, Gerrit Draisma, Willem P. T. M. Mali, Ernst, Kristiaan Nackaerts, Mathias Prokop, Dik, Mathijs Oudkerk, and Rob J. van Klaveren: *Risk-based selection from the general population in a screening trial: Selection criteria, recruitment and power for the Dutch-Belgian randomised lung cancer multi-slice CT screening trial (NELSON). International Journal of Cancer*, 120(4):868–874 (2007). doi:10.1002/ijc.22134.

[10] *Electron Microscopy research at Leiden University Medical Center.* `http://www.lumc.nl/rep/cod/redirect/1050/research/electron_microscopy.htm`.

[11] *Worldwide LHC Computing Grid.* `http://lcg.web.cern.ch/LCG/`.

[12] *LHC Computing Grid Optical Private Network.* `http://lcg.web.cern.ch/LCG/activities/networking/nw-grp.html`.

[13] Freek Dijkstra: *Modelling of Multi-Layer Transport Networks.* Ph.D. thesis, University of Amsterdam (2009).

[14] Arpad Szomoru, Andy Biggs, Mike Garrett, Huib Jan van Langevelde, Friso Olnon, Zsolt Paragi, Steve Parsley, Sergei Pogrebenko, and Cormac Reynolds: *From truck to optical fibre: the coming-of-age of eVLBI.* In R. Bachiller, F. Colomer, J.F. Desmurs, and P de Vicente (editors), *Proceedings of the 7th European VLBI Network Symposium*, pages 257–260. Joint Institute for VLBI in Europe (JIVE), Toledo, Spain (October 2004). `http://www.oan.es/evn2004/WebPage/ASzomoru.pdf`.

[15] Valeriu Tudose, R.P. Fender, M.A. Garrett, J.C.A. Miller-Jones, Z. Paragi, R.E. Spencer, G.G. Pooley, M. van der Klis, and A. Szomoru: *First e-VLBI observations of Cygnus X-3. Monthly Notices of the Royal Astronomical Society: Letters*, 375:L11–L15 (February 2007). doi:doi:10.1111/j.1745-3933.2006.00264.x. `http://arxiv.org/abs/astro-ph/0611054`.

[16] A. Rushton, R.E. Spencer, M. Strong, R.M. Campbell, S. Casey, R.P. Fender, M.A. Garrett, J.C.A. Miller-Jones, G.G. Pooley, C. Reynolds, A. Szomoru, V. Tudose, and Z. Paragi: *First e-VLBI observations of GRS 1915+105. Monthly Notices of the Royal Astronomical Society: Letters*, 374:L47 (2007). doi:doi:10.1111/j.1745-3933.2006.00262.x. `http://arxiv.org/abs/astro-ph/0611049`.

[17] *StarPlane Project.* `http://www.starplane.org/`.

[18] *Distributed ASCI Supercomputer 3 DAS-3.* `http://www.cs.vu.nl/das3/`.

[19] Franco Travostino, Paul Daspit, Leon Gommans, Chetan Jog, Cees de Laat, Joe Mambretti, Inder Monga, Bas van Oudenaarde, Satish Raghunath, and Phil Yonghui Wang: *Seamless live migration of virtual machines over the MAN/WAN. Future Generation Computer Systems*, 22(8):901–907 (2006). ISSN 0167-739X. doi:http://dx.doi.org/10.1016/j.future.2006.03.007.

[20] Global Lambda Integrated Facility (GLIF): `http://www.glif.is/`.

[21] Cees de Laat, Erik Radius, and Steven Wallace: *The Rationale of the Current Optical Networking Initiatives. Future Generation Computer Systems*, 19(6):999–1008 (August 2003). doi:10.1016/S0167-739X(03)00077-3. `http://www.sciencedirect.com/science/article/B6V06-48V83MF-5/2/d8aac1d72ec497da8c83c4a07fdfec0c`.

[22] Robert Patterson and Maxine D. Brown: *GLIF world map* (August 2005). Visualization by Robert Patterson, the National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign. Data compilation by Maxine Brown, University of Illinois at Chicago. Earth texture provided by NASA, http://visibleearth.nasa.gov., http://www.glif.is/publications/#info.

[23] Cees de Laat and Johan Blom: *User-Level Performance Monitoring Programme*. In *TERENA Network Conference 2000*. Lisbon, Portugal (May 2000). http://www.terena.org/events/archive/tnc2000/proceedings/8B/8b4.ppt.

[24] *TNO*. http://www.tno.nl/.

[25] Hans Keus: *Netforce Principles: An Elementary Foundation of NEC and NCO*. In *10th CCRT Symposium* (June 2005). http://handle.dtic.mil/100.2/ADA463913.

[26] Ltc T. Sierksma, Maj J. Hoekstra, Berry Jansen, Bert Boltjes, and Jaap van den Oever: *Geographical based Situational Awareness in Military Mobile Domain*. In *Military Communications Conference (MILCOM)*, pages 1–7 (2007). doi:10.1109/MILCOM.2007.4455345.

[27] *HP Openview*. http://openview.hp.com/.

[28] *Dynamic Resource Allocation Controller (DRAC)*. http://www.nortel.com/drac/.

[29] Iljitsch van Beijnum: *BGP*. O'Reilly Media, Inc. (2002). ISBN 9780596002541.

[30] Jerry Sobieski and Tom Lehman: *Common Service Definition*. Technical report, Mid-Atlantic Crossroads (MAX) (2005). http://dragon.maxgigapop.net/twiki/bin/view/DRAGON/CommonServiceDefinition.

[31] Rene Hatem, Almar Giesberts, and Erik-Jan Bos: *The ordering and fault resolution process for multi-domain lightpaths across hybrid networks*. Technical report, Global Lambda Integrated Facility (GLIF) (July 2006). http://www.glif.is/working-groups/tech/fault-resolution-0.9.pdf.

[32] Lars Fischer, Tom Lehman, Ronald van der Pol, and Thomas Tam: *GLIF Lightpath Identifier Proposal*. Technical report, Global Lambda Integrated Facility (GLIF) (August 2008). `http://www.glif.is/list-archives/all/pdfhxTQwd49Ef.pdf`.

[33] Niels Roosen: *Fault Detection and Isolation on Transport Networks*. Master's thesis, University of Amsterdam (September 2008). `http://www.science.uva.nl/research/sne/files/ntroosen-lmon.pdf`.

[34] A. Pras and J. Schoenwaelder: *On the Difference between Information Models and Data Models*. RFC 3444 (Informational) (January 2003). `http://www.ietf.org/rfc/rfc3444.txt`.

[35] J. Case, R. Mundy, D. Partain, and B. Stewart: *Introduction and Applicability Statements for Internet-Standard Management Framework*. RFC 3410 (Informational) (December 2002). `http://www.ietf.org/rfc/rfc3410.txt`.

[36] IETF: *Netconf working group*. `http://www.ops.ietf.org/netconf/`.

[37] DMTF: *Common Information Model (CIM)*. `http://www.dmtf.org/standards/cim/`.

[38] *Distributed Management Task Force (DMTF)*. `http://www.dmtf.org/`.

[39] Adrian Farrel and Igor Bryskin: *GMPLS: Architecture and Applications*. Morgan Kaufmann, first edition (2006). ISBN 978-0-12-088422-3.

[40] J. Zurawski, M. Swany, and D. Gunter: *A Scalable Framework for Representation and Exchange of Network Measurments*. In *2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Tridentcom 2006)* (March 2006). `http://acs.lbl.gov/~dang/tmp/trident.pdf`.

[41] *Network Measurements Working Group (NM-WG)*. `http://forge.gridforum.org/sf/projects/nm-wg`.

[42] J. W. Boote, E. L. Boyd, J. Durand, A. Hanemann, L. Kudarimoti, R. Łapacz, N. Simar, and S. Trocha: *Towards multi-domain monitoring for the European research networks*. *Computational Methods in Science and Technology*, 11(2):91–100 (2005). ISSN 1505-0602. `http://www.man.poznan.pl/cmst/2005/v_11_2/02-Boote.pdf`.

[43] Marcin Wolski, Stanislaw Osinski, Paweł Gruszczynski, Maciej Labedzki, Anand Patil, and Ian Thomson: *common Network Information Service Schema Specification*. Deliverable GN2-07-045v4, GÉANT (April 2007). http://www.geant2.net/upload/pdf/GN2-07-045v4-DS3-13-1_common_Network_Information_Service_Schema_Specification.pdf.

[44] *GÉANT2*. http://www.geant2.net/.

[45] Mauro Campanella, Radek Krzywania, Afrodite Sevasti, and Stella-Maria Thomas: *Generic Domain-centric Bandwidth on Demand Service Manager*. Deliverable GN2-08-129, GÉANT (August 2008). http://www.geant2.net/upload/pdf/GN2-08-129-DS3-3-4_Functional_Specification_and_Design_of_Generic_Domain-centric_BoD_Service_Manager.pdf.

[46] *Network Markup Language Working Group (NML-WG)*. http://forge.gridforum.org/sf/projects/nml-wg.

[47] Deepankar Medhi and Karthikeyan Ramasamy: *Network Routing*. Elsevier (2007). ISBN 978-0-12-088588-6.

[48] Marc Blanchet, Florent Parent, and Bill St-Arnaud: *Optical BGP (OBGP): InterAS lightpath provisioning* (March 2001). http://www.viagenie.ca/ietf/draft/draft-parent-obgp-01.txt.

[49] Elliotte Rusty Harold and W. Scott Means: *XML in a Nutshell*. O'Reilly Media, Inc., third edition (2004). ISBN 978-0596007645.

[50] Erik Ray: *Learning XML*. O'Reilly Media, Inc., second edition (2003). ISBN 978-0596004200.

[51] *The Semantic Web*. http://www.w3.org/2001/sw/.

[52] *Resource Description Framework (RDF)*. http://www.w3.org/RDF/.

[53] Eric Prud'hommeaux and Andy Seaborne: *SPARQL Query Language for RDF* (2005). http://www.w3.org/TR/rdf-sparql-query/.

[54] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee: *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard) (June 1999). Updated by RFC 2817, http://www.ietf.org/rfc/rfc2616.txt.

[55] T. Berners-Lee, R. Fielding, and L. Masinter: *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986 (Standard) (January 2005). `http://www.ietf.org/rfc/rfc3986.txt`.

[56] *Dublin Core Metadata Initiative*. `http://www.dublincore.org/`.

[57] Dave Beckett and Brian McBride: *RDF/XML Syntax Specification* (February 2004). `http://www.w3.org/TR/rdf-syntax-grammar/`.

[58] *Friend of a Friend (FOAF) Project*. `http://www.foaf-project.org/`.

[59] *Dublin Core Metadata Initiative*. `http://www.dublincore.org/`.

[60] Steve DeRose, Eve Maler, and David Orchard: *XML Linking Language (XLink) Version 1.0*. Technical report, World Wide Web Consortium (W3C) (June 2001). `http://www.w3.org/TR/xlink/`.

[61] Franco Travostino: *Using the Semantic Web to Automate the Operation of a Hybrid Internetwork*. In *GridNets conference proceedings* (October 2005).

[62] J.F. Shoch: *Inter-networking Naming, Addressing and Routing*. In *IEEE COMPCON*, pages 72–79 (1978).

[63] Jerome H. Saltzer: *On The Naming and Binding of Network Destinations*. *Local Computer Networks*, pages 311–317 (1982). Later re-published as RFC 1498.

[64] J. Noel Chiappa: *Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture*. Internet-draft (expired) (1999). `http://ana.lcs.mit.edu/~jnc/tech/endpoints.txt`.

[65] L. Berger: *Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description*. RFC 3471 (Proposed Standard) (January 2003). Updated by RFCs 4201, 4328, 4872, `http://www.ietf.org/rfc/rfc3471.txt`.

[66] David Goldberg: *What Every Computer Scientist Should Know About Floating-Point Arithmetic*. *ACM Computing Surveys*, 23(1):5–48 (March 1991).

[67] K. Shiomoto, D. Papadimitriou, JL. Le Roux, M. Vigoureux, and D. Brungard: *Requirements for GMPLS-Based Multi-Region and Multi-Layer Networks (MRN/MLN).* RFC 5212 (Informational) (July 2008). http://www.ietf.org/rfc/rfc5212.txt.

[68] *Generic functional architecture of transport networks.* Recommendation ITU-T G.805, International Telecommunication Union (ITU) (March 2000). http://www.itu.int/rec/T-REC-G.805/.

[69] Freek Dijkstra: *NDL Techonology Schemata.* http://www.science.uva.nl/research/sne/ndl/?c=20-Technology-Schemas.

[70] Fernando Kuipers and Freek Dijkstra: *Path Selection in Multi-Layer Networks.* *Computer Communications*, 32(1):78 – 85 (2008). doi:10.1016/j.comcom.2008.09.026. http://staff.science.uva.nl/~fdijkstr/publications/multilayer-pathselection.pdf.

[71] *Graphviz – Graph Visualization Software.* http://www.graphviz.org/.

[72] *NetherLight.* http://www.netherlight.net.

[73] *Google Maps API Homepage.* http://code.google.com/apis/maps/.

[74] Jeroen van der Ham and Freek Dijkstra: *Network Description Language Homepage.* http://www.science.uva.nl/research/sne/ndl/.

[75] *Dynamic Resource Allocation via GMPLS Optical Networks.* http://dragon.maxgigapop.net.

[76] Ronald van der Pol and Andree Toonk: *Lightpath Planning and Monitoring in SURFnet6 and NetherLight.* In *TERENA Network Conference 2007*. Lynby, Denmark (May 2007). https://noc.sara.nl/nrg/publications/LightpathPlanningAndMonitoring.pdf.

[77] Ronald van der Pol and Andree Toonk: *Lightpath Planning and Monitoring.* In *eChallenges Conference 2007*. The Hague, The NetherlandsH (October 2007). https://noc.sara.nl/nrg/publications/E-Challenges-v1.4.pdf.

[78] Ronald van der Pol: *Spotlight – NetherLight lightpath status.* http://noc.netherlight.net:8080/spotlight/.

[79] *Python NDL Toolkit (pynt).* `http://ndl.uva.netherlight.nl/trac/ndl/`.

[80] *Virtual Network Experiments toolkit(VNE).* `http://ndl.uva.netherlight.nl/trac/vne/`.

[81] Jeff Dike: *User Mode Linux.* Prentice Hall PTR, Upper Saddle River, NJ, USA (2006). ISBN 0131865056.

[82] *Virtual Distributed Ethernet (VDE).* `http://vde.sourceforge.net/`.

[83] *Private Network-Network Interface Specification.* Technical report, ATM Forum (1996). `http://www.ipmplsforum.org/ftp/pub/approved-specs/af-pnni-0055.001.pdf`.

[84] Whay C. Lee: *Topology aggregation for hierarchical routing in ATM networks. SIGCOMM Computer Communications Review*, 25(2):82–92 (1995). ISSN 0146-4833. doi:10.1145/210613.210625.

[85] Liang Guo and Ibrahim Matta: *On State Aggregation for Scalable QoS Routing.* In *Proceedings of the ATM Workshop*, volume 6, pages 306–314 (1998). `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.8296`.

[86] B. Awerbuch, Y. Du, B. Khan, and Y. Shavitt: *Routing through networks with hierarchical topology aggregation.* In *Third IEEE Symposium on Computers and Communications (ISCC)*, pages 406–412 (1998). doi:10.1109/ISCC.1998.702556.

[87] Q. Liu, M. A. Kök, N. Ghani, and A. Gumaste: *Hierarchical routing in multi-domain optical networks. Computer Communications*, 30(1):122–131 (December 2006).

[88] Zheng Wang and Jon Crowcroft: *Quality-of-service routing for supporting multimedia applications. Selected Areas in Communications*, 14(7):1228–1234 (1996). doi:10.1109/49.536364.

[89] B. M. Waxman: *Routing of multipoint connections. Selected Areas in Communications*, 6(9):1617–1622 (1988). doi:10.1109/49.12889.

[90] B. Awerbuch, Y. Azar, and S. Plotkin: *Throughput-competitive on-line routing.* In *SFCS '93: Proceedings of the Proceedings of 1993 IEEE*

34th Annual Foundations of Computer Science, pages 32–40. IEEE Computer Society, Washington, DC, USA (1993). ISBN 0-8186-4370-6. doi:10.1109/SFCS.1993.366884.

[91] Qingming Ma and P. Steenkiste: *On path selection for traffic with bandwidth guarantees. IEEE International Conference on Network Protocols*, 0:191+ (1997). ISSN 1092-1648. doi:10.1109/ICNP.1997.643714.

[92] Hui Zang, Jason P Jue, and Biswanath Mukherjee: *A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. Optical Network Magazine*, 1(1):47–60 (January 2000).

[93] *OPNET the network simulator tool.* `http://www.opnet.com/`.

[94] N. Ghani, Qing Liu, D. Benhaddou, N. S. V. Rao, and T. Lehman: *Multidomain optical networks: issues and challenges - Control plane design in multidomain/multilayer optical networks. IEEE Communications Magazine*, 46(6):78–87 (2008). doi:10.1109/MCOM.2008.4539470.

[95] A. L. Barabási and R. Albert: *Emergence of scaling in random networks. Science*, 286(5439):509 – 512 (1999).

[96] Carol Meertens and Joost Pijnaker: *Are Optical Networks Scale-Free?* Technical report, Systems and Network Engineering Master, University of Amsterdam (July 2007). `http://staff.science.uva.nl/~delaat/sne-2006-2007/p31/report.pdf`.

[97] Aric Hagberg, Dan Schult, and Pieter Swart: *NetworkX Python Package (version 0.36).* `https://networkx.lanl.gov/wiki`.

[98] R Development Core Team: *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria (2008). ISBN 3-900051-07-0, `http://www.R-project.org`.

[99] *Internet2.* `http://www.internet2.edu/`.

[100] D. Katz, K. Kompella, and D. Yeung: *Traffic Engineering (TE) Extensions to OSPF Version 2.* RFC 3630 (Proposed Standard) (September 2003). Updated by RFC 4203, `http://www.ietf.org/rfc/rfc3630.txt`.

[101] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus: *Requirements for Traffic Engineering Over MPLS.* RFC 2702 (Informational) (September 1999). `http://www.ietf.org/rfc/rfc2702.txt`.

[102] R. Coltun: *The OSPF Opaque LSA Option.* RFC 2370 (Proposed Standard) (July 1998). Obsoleted by RFC 5250, updated by RFC 3630, `http://www.ietf.org/rfc/rfc2370.txt`.

[103] K. Kompella and Y. Rekhter: *OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS).* RFC 4203 (Proposed Standard) (October 2005). `http://www.ietf.org/rfc/rfc4203.txt`.

[104] *Institute of Electrical and Electronics Engineers (IEEE).* `http://www.ieee.org/`.

[105] *Internet Engineering Task Force (IETF).* `http://www.ietf.org/`.

[106] *Open Grid Forum (OGF).* `http://www.ogf.org/`.

Summary

Communication over computer networks is an important part of our society today: we make phone calls, send emails, and surf the web. All these processes are enabled by the physical infrastructure of wires and fibers in the ground, combined with networking devices that communicate electronically.

Scientific research has also adapted to make use of the increased network connectivity. Researchers are sharing data sets, computing infrastructure and specialized equipment, all through the network. However, more and more scientific applications need better quality of services than the regular packet-switched Internet can offer. Such applications may produce so much traffic that if they use the regular Internet, they fail to run smoothly or disrupt other Internet traffic. These applications require dedicated network connections.

In the last few years a number of national research and education networks have moved to hybrid network infrastructures based on optical networking. The term 'hybrid network' means that the network carries the regular Internet traffic, and a different part is used to provision lightpaths, dedicated connections, to support researchers.

Lightpaths can span multiple administrative domains. A typical network connection between two universities first crosses a campus network, then a national research network, then goes through an international peering to another research network, then another campus network and finally to an internal network within a building. Different persons and organisations administrate all these networks.

Ideally, the scientific applications that need to move large amounts of data, or require a smooth network service simply ask for a lightpath and it is provided automatically. Currently, the configuration of a lightpath is performed manually and can take up to several weeks. The path through the topology must be specified and details communicated to all the network providers before the

lightpath can be configured.

The first part of this thesis describes the Network Description Language, a model for describing complex network topologies and technologies. The model defines a clear terminology for describing network topologies which can be linked to other descriptions and other resources. In the same way topology descriptions can be linked to descriptions of other networks, creating a distributed view of the global topology. These linked topologies can then be used to facilitate the configuration of lightpaths and optical networks.

Ideally all the network providers maintain the topology information so that there is a complete overview of possible connectivity and network details. However, network operators tend to be protective of detailed topology information, because of scalability, security, or policy reasons. Fortunately, it is also possible to share an aggregated view of the topology. This means that only some details of the network are published.

There are different levels of abstraction that can be used for topologies: from collapsing a whole network domain to a single point, to providing detailed information about the edge nodes and their internal connectivity. In part two of this thesis I describe an emulation to find out what kind of effect this aggregation has on the accuracy of inter-domain pathfinding.

From the results we can see that publishing detailed information about edge nodes and their internal connectivity shows almost no difference with pathfinding on unaggregated topologies. Collapsing a domain to a single node is clearly the worst performing solution. From the results we can conclude that with higher levels of abstractions, the accuracy of inter-domain pathfinding degrades significantly.

Samenvatting

Communicatie via computer netwerken is een belangrijk deel van ons leven geworden: we bellen, sturen emails en surfen op het web. Al deze processen worden mogelijk gemaakt door de fysieke infrastructuur van kabels en glasvezel in de grond, samen met de netwerkapparatuur die hierover communiceren.

Wetenschappelijk onderzoek maakt ook gebruik van deze toegenomen connectiviteit. Onderzoekers delen grote databestanden, computerfaciliteiten en gespecialiseerde apparatuur, allemaal via netwerken. Een belangrijke ontwikkeling is dat steeds meer wetenschappelijke toepassingen een betere servicekwaliteit nodig hebben dan het normale, pakketgeschakelde internet kan bieden. Sommige van zulke toepassingen genereren erg veel verkeer. Zoveel zelfs dat als dat verkeer over het Internet zou gaan, het niet vloeiend zou lopen, of het andere verkeer op het Internet kan verstoren. Dit soort toepassingen hebben aparte verbindingen nodig, buiten het Internet om.

In de afgelopen paar jaar zijn een aantal nationale onderzoeksnetwerken overgestapt op een hybride netwerk infrastructuur gebaseerd op optische netwerken. De term ‘hybride netwerk’ betekent dat het netwerk enerzijds gebruikt wordt voor het reguliere internetverkeer en dat een ander gedeelte van datzelfde netwerk gebruikt wordt voor ‘lichtpaden’, speciaal opgezette verbindingen voor wetenschappelijke toepassingen.

Lichtpaden kunnen door meerdere netwerken gaan. Een typische netwerkverbinding van een universiteit naar een andere gaat eerst door een campus netwerk heen, dan door het nationale onderzoeksnetwerk. Vervolgens gaat de verbinding via een internationale koppeling naar een ander onderzoeksnetwerk, weer een campus en uiteindelijk door het netwerk binnen een gebouw. Al deze verschillende netwerken worden beheerd door andere personen en instellingen.

Idealiter vraagt een wetenschapper die grote hoeveelheden data wil versturen om een lichtpad en wordt dat automatisch gegeven. Op dit moment worden zulke lichtpaden vooral nog met de hand ingesteld en dat kan enkele weken duren. Het pad door het netwerk moet worden vastgesteld en daarna moeten de details worden gecommuniceerd naar alle netwerk-providers, voordat het lichtpad ingesteld kan worden.

Het eerste deel van dit proefschrift beschrijft de *Network Description Language*, een model voor het beschrijven van complexe netwerken en technologieën. Dit model definieert een duidelijke terminologie om netwerktopologieën te beschrijven die kunnen worden gekoppeld aan andere beschrijvingen van bijvoorbeeld apparatuur. Op een zelfde manier kan een netwerkbeschrijving ook gekoppeld worden aan de beschrijvingen van andere netwerken, zodat je een gedistribueerde omschrijving krijgt van het globale netwerk. Deze gekoppelde beschrijvingen kunnen dan gebruikt worden om het instellen van lichtpaden en optische netwerken makkelijker te maken.

In het ideale geval zijn netwerkbeschrijvingen van alle verschillende netwerk-providers openbaar, zodat een compleet overzicht van de mogelijkheden en netwerkdetails beschikbaar is. Helaas geven netwerkbeheerders de details van hun volledige netwerk niet graag vrij vanwege schaalbaarheid, veiligheid of policy-redenen. Het is echter ook mogelijk om een geabstraheerde beschrijving van een netwerk te maken. Dit betekent dat alleen bepaalde details van het netwerk gepubliceerd worden.

Er zijn meerdere manieren om een abstractie te maken van een netwerkmodel: van volledig geabstraheerd, waarbij het hele netwerk kan worden platgeslagen tot één punt, tot bijna geen abstractie, met een beschrijving van de randpunten met een omschrijving van de interne verbindingen. In het tweede deel van dit proefschrift beschrijf ik een emulatie-experiment om er achter te komen wat voor een effect abstractie heeft op het vinden van paden door meerdere domeinen.

Uit de resultaten blijkt dat gedetailleerde informatie over de randpunten en hun connectiviteit bijna geen verschil geeft met de resultaten gebaseerd op volledige informatie. In de resultaten is ook te zien dat een heel netwerk afbeelden op één punt duidelijk de slechtste oplossing is. Met hogere abstractieniveaus neemt dus ook de nauwkeurigheid van het vinden van paden af.

Acknowledgements

The days of solitary research are long gone, and this research could not have been possible without the help and support of a great number of people.

First of all, I will be forever indebted to my first supervisor, Cees de Laat, for making this project possible, for the excellent support along the way, and for sharing his inspiring vision. I would also like to thank Paola Grosso for her supervision, support, and endless patience in discussions and proof-reading. Many thanks to my promotor, Peter Sloot, for his comments, discussions, and pushing me to make scientific experiments. I am also very grateful to the committee, Pieter Adriaans, Henri Bal, Hans Keus, Rob Meijer, Dimitra Simeonidou, for inspecting my thesis.

A special thanks goes to Freek Dijkstra, for being a fun colleague, an excellent sparring partner, the support in making this thesis, and for having a good listening ear. This work would also not have been possible without the help of all my other colleagues of the ~~AIR~~ SNE group that I have had over the years: Alexis, Arie, Bas, Bert, Carol, Damien, Erik, Fred, Guido, Hans, Jaap, JP, Leon, Li, Matthijs, Mihai, Ralph, Rob, Rudolf, Sander, Steven, and Yuri. And also my colleagues at SARA: Andree and Ronald.

Thank you to Hans Keus, Ronald Poell and Bert Boltjes of TNO, for their good discussions, support and pointers to related work in the defence area.

Many thanks to the people at MAX and Internet2 that I worked with. Chris Tracy, you are a great friend to have, and you have always been extremely supportive. Jarda Flidr, I really enjoyed our road-trip, and our (cynical) discussions. A special thank you to John Vollbrecht, for our endless discussions, which sometimes forced me to re-examine things. Also my thanks go out to Tom Lehman and Jerry Sobiesky for their enlightening discussions.

A big thank you to everyone in the GLIF community for making it a great community which is open to new ideas, provides feedback, valuable data and experiences upon which we can all build our research.

I would also like to thank Karst Koymans, Jeroen Scheerder and Vincent van Oostrom for their inspirational courses and encouraging me to pursue a career in research.

Thanks also goes out to Judith, you have supported me for a long time.

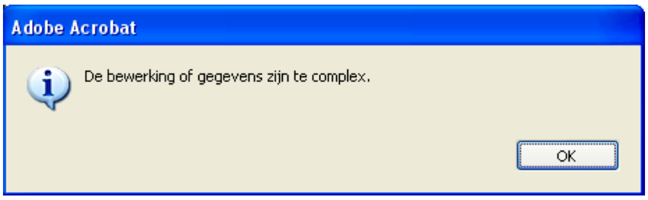
I am grateful to Jochem, Rudy, Sander, Sander, Staf and Tom of the Second ~~Wednesday~~ Friday club for the long evenings filled with great fun and games over the years.

Alex, thank you for your support, and for being there when I needed you. This is another step in our tropical island retirement plan!

I owe a great deal to my parents, Johan and Digna, my brothers, Joost and Sjoerd, and my grandparents. Thank you for supporting me all these years.

Finally, I am extremely grateful to and for Ela. You are showing me that there is a lot of beauty in the world, even in just small and simple things.

Last Minute Addendum: A big thank you to the printer for acknowledging that my research is still too complex for computers to understand.


Adobe Acrobat

De bewerking of gegevens zijn te complex.

Figure C.1: *'The calculation or data are too complex.'*