NTNU – Trondheim
Norwegian University of
Science and Technology

Open network topology services

Roy Sindre Norangshol

Master of Science in Informatics
Submission date: June 2013
Supervisor: Svein Erik Bratsberg, IDI
Co-supervisor: Olav Kvittem, UNINETT
Vidar Faltinsen, UNINETT
Morten Brekkevoll, UNINETT

Norwegian University of Science and Technology
Department of Computer and Information Science

OPEN NETWORK TOPOLOGY SERVICES

ROY SINDRE NORANGSHOL

UNINETT is the national research IP network operator in Norway. UNINETT provides universities, university colleges and research institutions with access to the global internet as well as access to a range of online services. UNINETT also offers counselling and act as secretary and coordinator in collaborative activities between the institutions interconnected by UNINETT.

UNINETT is a multilevel network that is built from cables, fibers, lambdas, VLANS, IP networks and VPNs. There is need to solve the problem of documenting the global research network infrastructure that allows for common tools and for exchange of information between universities and research networks. There has been several initiatives to address part of these problems like : In Norway with NAV a network managment system for campuses, in the Nordic area at NORDUnet, in Europe with Géant projects as well as American/Internet2 activities.

The task is to survey state of the art of tools and data models in this area, and recommend an open system architecture containing data models, exchange protocols and available system components . A protoype implementation that demonstrates the capabilities of the architecture should be built if time allows it.

Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology
June 2013 – version 1.0

ABSTRACT

This master project examines whether there is an existing model for describing network topologies in abstract and generic manner. I also looked for networking protocols for exchanging network topologies and handling of dynamically creation of circuit connections across domains. I've also been working on a prototype for visualization of network topologies using Network Administration Visualized (NAV) as a data backend, and further to check the possibilities to use the found topology model in my prototype.

My findings shows that there is progress towards creating a standard topology model to describe network topologies in an abstract and generic manner. There is also progress in creating a network architecture with networking protocols for exchanging network topologies across domains and providing a connection reservation service to handle creation of dynamically circuit connections. Prototype shows there is lots of ideas for further works on what to implement in regards of the found network topology model and networking systems that was found.

SAMMENDRAG

Dette masterprosjektet undersøker om det finnes en eksisterende modell for å beskrive nettverkstopologier på en abstrakt og genrell måte, samtidig som det har blitt søkt etter en nettverksarkitektur og nettverksprotkoller for å utveklse nettverkstopologier på kryss av domener mellom nettverksoperatører og kunne dynamisk reservere dedikerte nettverkstilkoblinger mellom to punkter. Jeg har også jobbet med prototyping av nettverksvisualisering hvor jeg har undersøkt hvordan dette kunne knyttes opp mot NAV, og senere muligheten for å bruke modellen for å beskrive nettverkstopologier i prototypen.

Funnene mine viser at det allerede er storsatsing på området for å beskrive nettverkstopologier og det jobbes med en nettverksariktektur og nettverksprotokoller for utveksling av topologier og opprettelse av dynamiske nettverkstilkoblingsreservasjoner på kryss av nettverksdomener og prototypen viser store muligheten for videre arbeid av hva som kan implementeres i forhold til nettverksmodellen og nettverkssystemene som ble funnet.

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [12]

PREFACE

This report presents the work done for my master's thesis at the Department of Computer and Information Science at the Norwegian University of Science and Technology. The work was done during the fall 2012 and spring of 2013 and concludes my master's degree in Informatics.

I would like to thank UNINETT, especially Olav Kvittem, Morten Brekkevoll and Vidar Faltinsen for assisting and giving interesting problem to dive into and providing valuable feedback and a office space to quietly hack and thinkering in.

I also would like to give a shout out at all who is participating in working groups under the Open Grid Forum working towards open standards but especially the Network Markup Language Working Group and Network Service Interface Working Group.

Last but not least I would like to thank my supervisor Svein Erik Bratsberg for invaluable guidance and feedback through the semesters.

NTNU – Trondheim
Norwegian University of
Science and Technology

Trondheim, June 2013

Roy Sindre Norangshol

CONTENTS

LIST OF FIGURES

ACRONYMS

UML Unified Modeling Language

ISP Internet Service Provider

RDF Resource Description Framework

DWDM Dense Wavelength Division Multiplexing

NOC Network Operations Center

GLIF Global Lambda Integration Facility. International colloboration to bring Layer-1 capabilities to resarch/education community on world-wide basis.

GOLE GLIF open lightpath exchange, comprised of one or more network devices performing lightpath switching.

NAV Network Administration Visualized

VLAN Virtual Local Area Network

NREN A National Research and Education Network

STP Service Termination Point

SDP NSI Service Demarcation Point (external relations)

NSA Network Service Agent

NSI Network Service Interface

NSI-WG NSI Working Group

NSI-CS NSI Connection Service

NSF Network Services Framework

NRM Network Resource Manager

RA Request Agent

PA Provider Agent

AG The Aggregator

uPA Ultimate Provider Agent

uRA Ultimate Request Agent

cNIS common Network Information System

NDL Network Description Language

NML Network Markup Language

NML-WG NML Working Group

OGF Open Grid Forums

VxDL Virtual private eXecution infrastructure Description Language

RDBMS Relational Database Management System

AMD Asynchronous Module Definition

MVC Model-View-Controller

MVP Model-View-Presenter

SVG Scalable Vector Graphics

DOM Domain Object Model

XML Extensible Markup Language

VLBI Very-long-baseline interferometry

Part I

INTRODUCTION

1 INTRODUCTION

As experimentation and new developments is at high pace in research networks, especially the increasing focus on offering applications circuit-switched connections (also known as lighpaths) that can provide guaranteed network services. Due to this increasing focus for creating circuit-switched connections dynamically, there is also a demand for complex provisioning of such paths across intra domain and inter domain. For network provisioning to work across inter domains there is need for defining a standard for sharing network topology information and a standard which is designed for dealing with dynamically creation of circuits (connections) that transit several transport network providers.

As described in section 2 it is difficult to make guaranteed bandwidth allocations over IP (layer 3) over the Internet, due to no reservation for bandwidth for the current connection, and your probably sharing the connection pool with several other ISPs.

A National Research and Education Network (NREN) has the unique possibility to test new technologies when it comes to routing and making a reserved connection. NRENs often have good infrastructure built on fiber optics, often referred to as Dense Wavelength Division Multiplexing (DWDM) networks as each lighpath (or lambdas as they often are called) are provisioned especially for dedicated links to use a dedicated wavelength. This makes NRENs have a possibility to easily dedicate a dedicated layer 1 connection without interferences from any other traffic sources and resembles of being a circuited switched network instead of packet switched network which is required for applications that are sensitive towards throughput (the time the transfer takes), jitter (variation in arrival time at the destination of packets) and delays (the time for a packet to get to the destination). [6]

A typical fiber in single-mode can hold around 40 lambdas à 100gbps using DWDM, and a fiber cable can hold several hundred fibers.

NREN has already initiated something called Global Lambda Integration Facility. International colloboration to bring Layer-1 capabilities to resarch/education community on world-wide basis. (GLIF) which is world wide laboratory for application and middle ware development which utilize the powers of DWDM networks.

1.1 CONTEXT

In the following scenario: «How do I get from network A to network E, while I know this involves going through networks B, C and D?»

Example of use case where such a scenario is an often asked question: Two hospital teams need a video link between em so team 1 can assist team 2. Now this team 1 is located at St. Olav's University Hospital in Trondheim, Norway and need to assist team 2 who is doing a operation at a patient in Daejeon, South Korea.[3][1] This video link required a certain bandwidth, and a minimum latency so the two teams of hospital doctors can assist each other live under the operation.

"Historically, connections across these transport networks have been reserved and provisioned in a variety of ways. The most common approach is manual provisioning – typically performed by a network engineer."[8]

The network operators knows that we are in need of a dedicated connection to ensure it qualifies the bandwidth and latency requirements. It would be sad if the hi-definition video links has too little bandwidth and the image just looks like squared pixels and renders the image useless. Secondly if the latency requirements is not met, time critical decisions as «should I cut this blood vessel now!?» which requires a fast reply might end up having high consequences as the message delivered to the other team was delayed due to latency.

As of today such creation of dedicated circuits involves manual communication between several operators with manual suggestions on how to get from network A to network E through all involving parts (network B, C, and D) in the best manner to qualify all the requirements. Now we are interested in making this process faster, which involves in finding a standard for describing network topology, and a system to automatically reserve connections and do the peering between all involved parties so network A reaches network E qualifying all the requirements.

1.2 PERSONAL MOTIVATION

I've always been fascinated about networks and networking. How data can make it's way from A to C via B, how computers are able to talk to each other online, identified by the «magical» numbers known as IP addresses. How does this end up of being ones and zeros and transmitted over the cable. With my bachelor degree in computer engineering specialized in networking and network architectures and have been working at the open source project NAV where my contributions have been working on visualizing the known network topology for the network it monitors it. It felt normal for me to continue with related work in my thesis which is based on networking & I find networking to be a really interesting field. UNINETT also helps with having a great working staff with key knowledge about networking as one of the tasks UNINETT has is act as the the Norwegian NREN.

1 This is a true story, and was done manually by each NREN to qualify for all the demands required for the video link

1.3 GOALS

My main goal for the thesis can be relate to this following scenario: «How do I get from A to E via BCD which satisfy a requirements of throughput and latency in a automated way?». This is interesting as it involves a lot of difficult aspects you have to think about. I've focused on visualization of network topologies and make the visualization application support extendability, so in the future it should be easy to add support for aiding other network provisioning software to provide a special topology which only exports the resources you want to export. There is also interesting problems such as path finding, but I've focused on the following goals:

You can read more about SDP and network provisioning in chapter 4

- Finding/creating a way to describe network topologies in generic way including support of describing multi-layer networks and multi-domain networks.
- Finding/creating a system that deals with dynamically creation of connections (circuits) which support multi-domain.
- Prototyping of visualization together with a chosen topology model
- Prototyping/testing topology model towards a network provisioning system

2 INTRODUCTION TO NETWORKING

Internet now a days is mostly using packet-switched method for moving data through a network. There is two common methods for moving data through a network, the first one being packet switching, and the other one is circuit switching. The differences between packet switching and circuit switching is that in circuit switching resources needed along the path for the communication between the hosts are reserved end-to-end for the duration of the communication session. Resources in packet switching does not get reserved compared to circuit switching, and resources are used on demand and this might lead to congested link as a consequence or even packet loss. [14]

A congested link means that packets have to wait in a queue (a buffer) before it gets sent, or packets might even be dropped[14]

Internet uses it's best effort to deliver packets as fast as it can, but it does not make any guarantees as the architecture of the Internet is built upon packet switching which is cheaper equipment and less complex then circuit switching. Packet switches uses (mostly) something called store-and-forward for its transmission which practically says it has to read the whole packet before it is forwarded and if the link is congested this will add even more delay to the packet before it is forwarded and again can lead to packet loss. An analogy to real life example would be coming to the restaurant and you have to wait at the entrance before you can go to your table.[14]

Packet loss is analogy with the waiter telling you there is no table available when you enter the restaurant, and you have to leave and try again.[14]

Problems with packet switching when it comes to implementing premium services is that packet switching requires complex signaling software to coordinate the operation of enabling a end-to-end circuit along the end-to-end path for the reservation of of end-to-end bandwidth.[14]

This is why circuit switching has caught the attention again of the NREN world, where each NREN have large computing grids built upon fiber optics as a transit medium which already requires circuit switching.

2.0.1 *Peering*

Peering is an important part of the Internet, as the Internet is a «network of networks» and ISPs[1] need to know where to forward packets to outside it's network. End users are normally connected to a tier 3 or tier 2 ISP and is also known as access ISPs as they provide access to the rest of the Internet on behalf of the user. Tier 1 ISPs is known for being connected to every other tier-1 ISP on the Internet and has a large bandwidth available for their links. Tier 1 ISPs are also known

1 Internet Service Providers

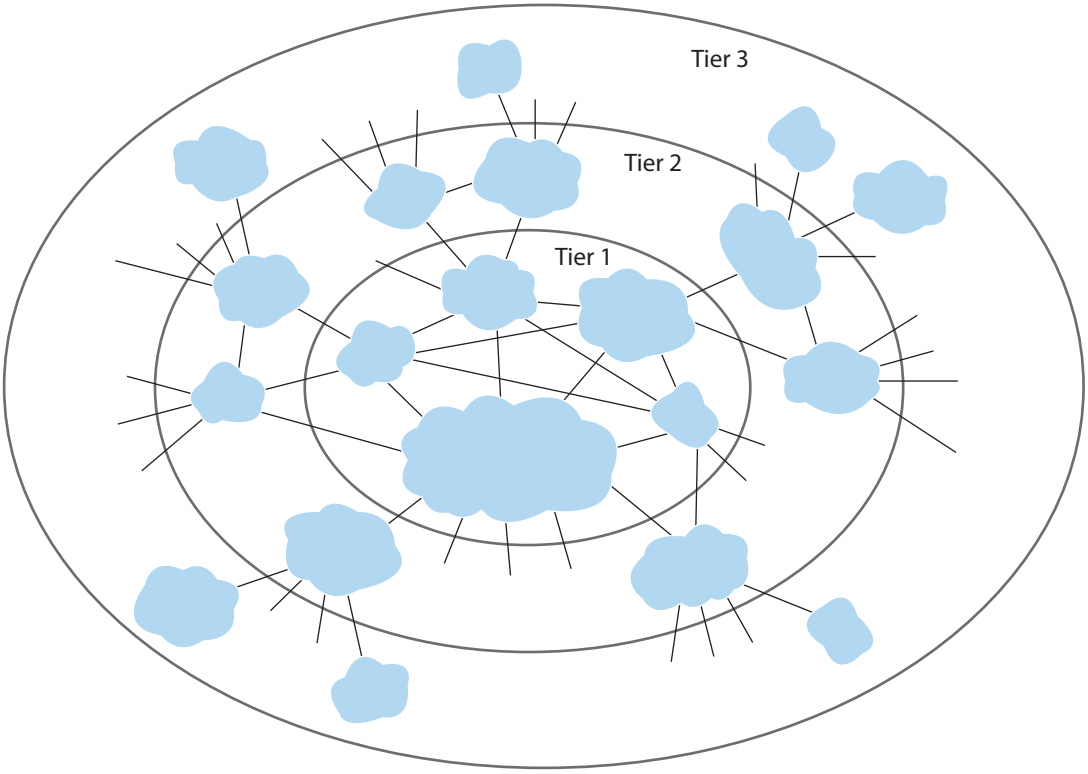

Figure 1: Interconnection of ISPs between tiers[14]

as Internet backbone networks and lives at the top of the tiered hierarchy of ISPs. It's also connected to a large portion of tier-2 ISPs and other customer networks while also are known to have international coverage. Tier 2 ISP is known for having a country coverage and probably connected only to a few Tier 1 ISPs. See figure 1. [14]

2.0.2 *Networks are layered*

As network topologies easily gets very complicated, we humans try to figure out a structure to understand the complex systems involved. Now picture a airline system, how would you describe all the actions required to get from destination A to B? We would first try to separate these actions that is done by you or someone takes for you, so from flying from destination A to B we would start by first buying a ticket. After purchasing a ticket you would reach for the check in process. Next on the schedule is to figure out your boarding schedule and find your correct gate (load) and jump a board the airplane when the gate opens. Next steps would be the airplane to take of, and it would be routing in air until it reaches it's destinate airport for landing. Next step would be to exit the gate (unload) and go claim your baggage. These actions can be seen in figure 2 which shows all the actions taken. [14]

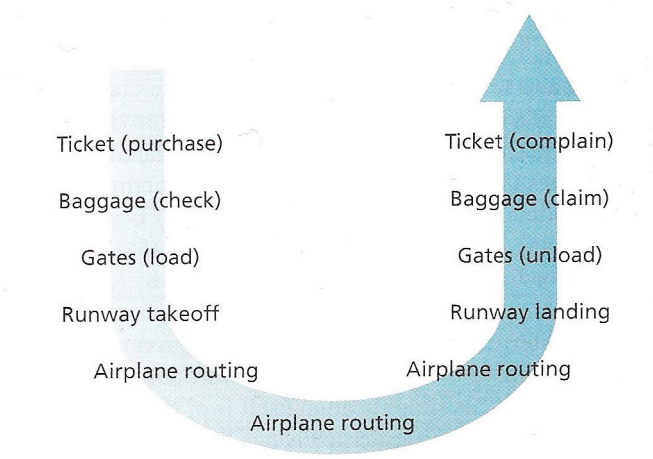

Figure 2: Airplane actions[14]

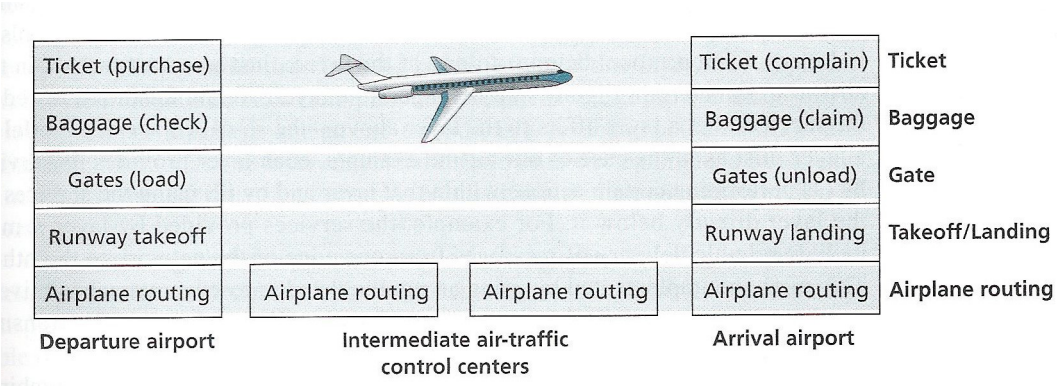

Figure 3: Airplane layers[14]

Now we have amount of actions that is required to take us from A to B by flight, but it's not quite the analogy we're looking for as we are looking for a structure. We notice we can separate actions into a layers. At entering the airport you can purchase a ticket, and before leaving the airport at your destination you can complain about your ticket. Next layer is the baggage routine, your leaving the baggage at the baggage band when checking in, and your claiming your baggage after you have left the gate. The baggage can be separated into it's own layer, the same you can do for loading and unloading at the gate. Next is taxing at both airports, for take off and landing. And when the airplane is in the air we are routing. See figure 3 [14]

Take notice of that each layer is separated from each other, the ticketing layer takes care of getting a user ticket, or complain about it if the service wasn't satisfied as you thought it should be. Baggage layer can only be checked in after you have a obtained a ticket from the ticketing layer. So we notice each layer offers functionality for the layer above it, just as routing the airplane to it's destination is done after the loading at the gate and runaway for the takeoff is done. [14]

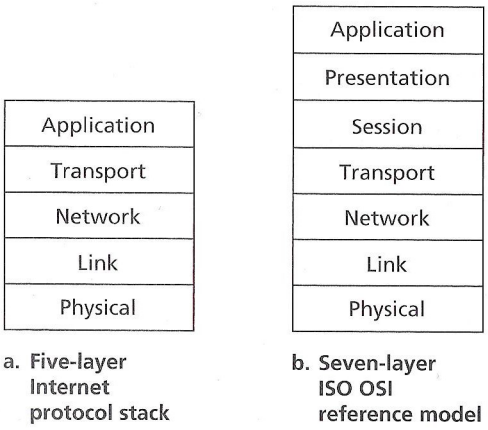

Figure 4: Internet Model and OSI reference model [14]

We can do the same layered architecture for describing network topologies. As of today network architects have used the OSI reference model which separates the network topology in segments so it is easier so see the flow. OSI reference model contains of seven layers, but we'll cut the 3 at the top (Application, Presentation, Session) and call it Application. We then end up with Application, Transport, Network, Link and Physical layers and is commonly referred to as the Internet Model. [14] See figure 4

Before we go into a short intro from each layer in the Internet protocol stack, it is important to remember what we have just thought. We divided things up in layer using the service model, and that a layer is offering services for the layer above. Each layer itself then provides it's services by performing certain actions within a layer and by using the services available from the layer below it. An example is IP delivering unreliable edge-to-edge messages while the TCP layer above it has functionality to detect and retransmit lost messages.[14]

We'll give a short intro and a top-down instruction about these five layers.

2.0.3 *Application layer (layer 5-):*

This is where network applications and their application-layer-protocol reside in the Internet Model. Common application layer protocols is HTTP and DNS, so the service translating domain names to IP-addresses is an application service living in this layer. So whenever the application service requires to communicate with another host it will use the protocol defined in the application layer to exchange packets of information between the hosts. These packets of information at the application layer is referred to as a *message*. [14]

2.0.4 *Transport layer (layer 4):*

The transport layer job is to transfer application-layer *messages*. There is two common transport protocols available for the most used network layer (layer 3) on the Internet, IP. These are TCP and UDP. A quick intro of these is that TCP has a slighter higher overhead but provides a connection-oriented service to it's applications and provides guarantees of delivery of application layer messages sent to it's destination. TCP also comes with flow control which ensures the sender and receiver speed is matching and it also breaks its packets into smaller segments to throttle network transmission rate if the network gets congested. [14]

Unlike UDP protocol which is a connectionless service to it's application and is simply does a «Fire and forget» of the message it carries from the application layer. It has no service for guaranteed of reliability, no flow control or no congestion control. Packets wrapped in a transport protocol is commonly referenced to as a *segment*. [14]

2.0.5 *Network layer (layer 3):*

IP (Internet Protocol) is such a protocol. It is responsible for moving network-layer packets from one host to another host. This packets are commonly referred to as *datagrams*.

When sending a *datagram* it has a network layer destination (IP-address) and it's *segment* from the transport layer. A human analogy would be mailing a letter where the post address would be the network layer destination and the letter itself is the *segment*.

The Internet network layer protocol also contains routing protocols that tells which path or route the *datagrams* should take from source to the destination. As Internet is a network of networks, and within a network the network administration choose to run h(is|er) choice of a routing protocol for how *messages* should be delivered inside it's own network and over to the next if it's destination is outside it's own network. [14]

2.0.6 *Link layer (layer 2):*

The network layer (The Internet) routes a *datagram* through a series of routes of source and destinations. Moving *datagram* from a host to another host it has to relay on the link layer to transfer the *datagram*. For each node the network layer passes the datagram to the link layer which delivers the datagram to the next node along the route and when it arrives at the next node the link layer passes the datagram back up to the network layer. This continues until it the network layer figures out it reaches it destination and it passes it's *segment* up to the transport layer and all the way back up the the application layer. [14]

As you see, much how like we interpreted the layers on «how to fly from A to B».

Packets inside the link layer we referrer to as *frames*, and when we're sending frames in the link layer protocol the frames get handed of to the Physical layer below to do the transfer between two links. [14]

Ethernet is the most common link layer protocol. [14]

2.0.7 *Physical layer (layer 1):*

It's job is to move entire *frames* from the link layer above from one network segment to another adjacent network segment. As a binary bit is moved across a link in many different ways, the physical medium has multiple physical layer protocols to deliver these frames. These implementations is done in hardware (network cards) and is also dependent on the actual transmission medium which for example can be twisted pair copper wires or fiber optics for example. [14]

Part II

TOPOLOGY MODEL

3

WHY SEARCH FOR A MODEL?

As experimentation and new developements is at high pace in research networks, especially the increasing focus on offering applications circuit-swithced connections (alsa known as lightpaths) that can provider guarnteed network services. Due to this increasing focus for creating circuit-switched connections dynamically, there is also a demand for complex provisioining of such paths across intra domain and inter domain. For network provisioning to work across inter domains there is need for defining a standard for sharing network topology information.

inter domain used to describe interaction between domains. It is most commonly used in the fields of multicasting and routing between internets, or as a substitute for the term inter-server. Intra-domain is the interconnection of servers within a single domain.

Today all products has used their own models (which only describes their required metadata for the applications purpose) to describe network topologies, these be products such as NAV, common Network Information System (cNIS), perfSONAR to name a few application suits that requires network topology descriptions and even as of today still using their own models and not a shared conceptual model for represetning a network topology. These be any applicatinos of type provisioning, monitoring and visualizing. [6]

It doesn't get better when you have multiple applications or instances of the same software which requires to do work on the same topology information and it's state isn't shared between the applications. In for example network provisioning where a user is creating a dynamically allocated circuit of a connection crosses multiple domains, it would be difficult to monitor these dynamically allocated circuits in every domain if their not using the same topology model across all systems.[11]

3.1 REQUIREMENTS

To describe a network topology we need to manage to describe it as a layer independent network topology, figure out the common properties that are common across multiple network technologies and define it mechanism in a schema and define it as a standard. The standard for describing a topology must support extendability so other third parties (application suits) is able to use the topology description as a much and plug in extra information as required for their application. Example for a visualizing application for network provisioning, we would need to know the circuit-switched path and involved nodes, throughput, jitter, and connection reservation time to have a live updated visualization. It is also important it supports multi-layer as mentioned above and multi-domain, and scalable.[11]

3.2 MODELS FOR DESCRIBING NETWORK TOPOLOGIES

A few models for describing network topologies exists already, such as cNIS, Network Description Language (NDL), Virtual private eXecution infrastructure Description Language (VxDL) , perfSONAR and NML. The last one is a combined effort by the Open Grid Forums (OGF) to create a open standard for describing network topologies.[11]

3.2.1 *Common Network Information Service (cNIS)*

cNIS is a collection of software used for providing a common shared database/storage for all relevant network information about a single administrative domain. cNIS uses RDBMS schema's for storing the relevant topology information. cNIS also works together with AutoBahn which is a network provision service implementing Network Service Interface (NSI) version 1.[6] This is much like how NAV works minus the network provisioning service.

3.2.2 *Network Description Language (NDL)*

NDL goal is to provide lightpath provisioning applications to exchange topology information intra and inter domain, and is a ontology which uses Resource Description Framework (RDF) schema's to categorize it's information: network topologies, network technology layers, network device configuration, capabilities and network topology aggregations. NDL is modular with different set of schema's: topology schema which describes devices, interfaces and connections between them on a single layer. Layer schema describes generic properties of network technologies and the relation between network layers. Capability schema describes device capabilities. Domain schema describes administrative domains, services within a domain and how to give an aggregated view of the network in a domain. The physical schema describes the physical aspects of network elements.

NDL main usages is generation of network maps, lightweight offline path finding and multi-layer path finding and last network topology information exchange. [6]

3.2.3 *perfSONAR topology*

perfSONAR is a software for network measurement and in the early stages it topology schema was tightly coupled with network interface and application layer endpoints etc. It has sense changed to support hybrid networks and it's schema is defined to more general purpose elements such as domains, nodes, links, ports and services. XML namespaces is used for specifying more detailed information about elements, such as Ethernet link or HTTP Web Service. perf-

SONAR topology supports adding new technology specific elements by bumping versions of the elements or add a new namespace for a new technological-specific element. [6]

3.2.4 *Network Markup Language (NML)*

NML Working Group (NML-WG) was founded in the beginning of 2007 in February with Martin Swany and Paola Grosso as group chairs in the working group that is part of the OGF. Later Freek Dijkstra replaced Paola as of it is now in 2013.

«OGF is an open community committed to driving the rapid evolution and adoption of applied distributed computing. Applied Distributed Computing is critical to developing new, innovative and scalable applications and infrastructures that are essential to productivity in the enterprise and within the science community. OGF accomplishes its work through open forums that build the community, explore trends, share best practices and consolidate these best practices into standards.» mission statement from their website[1].

NML is a combined effort initiated by the authors of cNIS, NDL, VxDL, perfSONAR to describe a network topology and it's main work is based on NDL RDF schema, Unified Modeling Language (UML) schema used for perfSONAR network descriptions and UML schema used in the cNIS project for GÉANT2 network descriptions. NML UML class diagram you can see in figure [6, 11]

GÉANT is the pan European data network dedicated to the research and education community.

3.3 WHY NML?

As already large known projects for networking dealing with both provisioning, visualization and monitoring has agreed on working together to build an open and free standard for describing network topologies this is a good choice instead of going for vendor-locked or create-yet-another-format for descriptions of network topologies who might not deal of describing hybrid network topologies that can work in a hybrid environment that is required for provisioning applications to exchange topology information intra and inter domain which NML can do. NML describes network topologies in an abstract and generic way and has extendibility for further improvements [6, 11]

3.4 EXPLAINING THE MODEL

The basic schema for NML is generic enough for describing the physical (layer 1) or packet-oriented networks as NML describes logical connection-oriented network topologies. Al thought the first version of the NML base document doesn't deal with signal degradation or

1 http://www.ogf.org/About/abt_overview.php

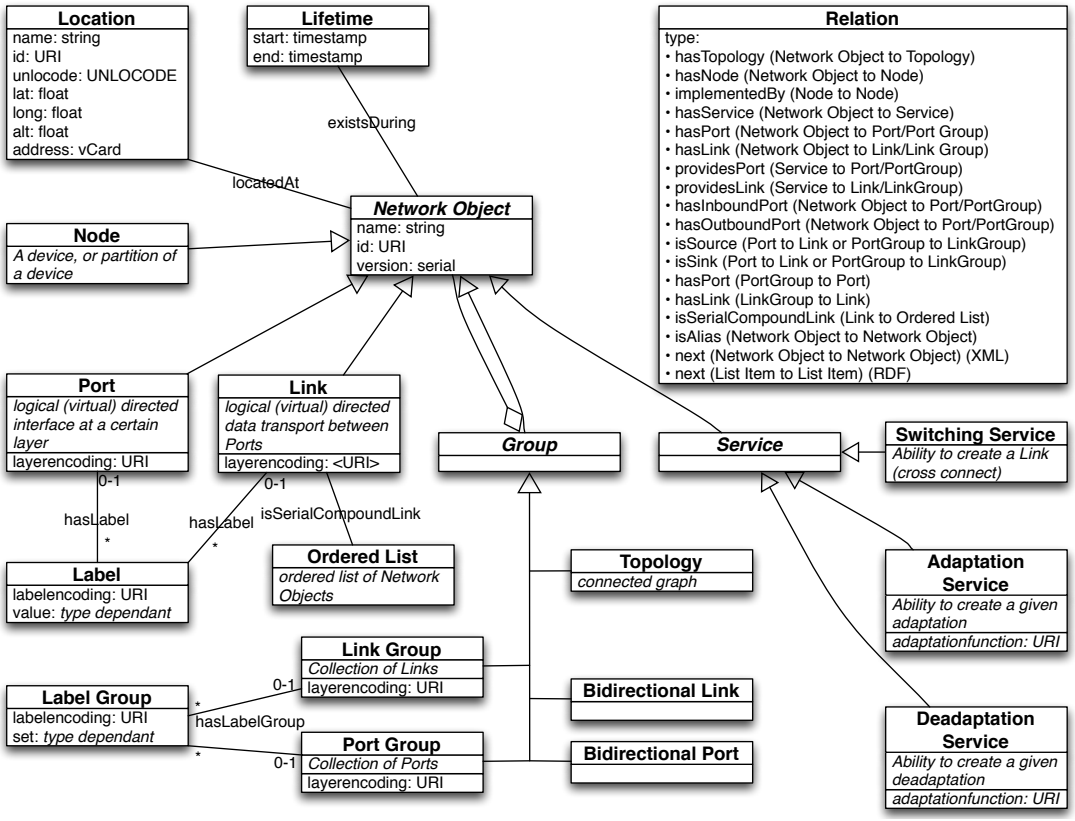

Figure 5: UML class diagram of NML[10]

complex routing tables but can easily be added later as NML extensions to the NML base model.[10]

NML can describe hybrid network topologies including both multi-layer networks and multi-domains networks.[10]

Multi-layered network can be a virtualized network running on top of another, but can also be using different technologies. Multi-domain network descriptions can include aggregated or abstracted network topologies. In short NML can not only describe a static network topology, but also it's capabilities and its configuration.[10]

It is also important to understand NML only attempts to describe the data plane of a computer network, and not the control plane. Network provisioning software using network provisioning standards and network monitoring standards can easily be tied together with NML by easily extending the model.[10] See figure 6for a overview over NML relations.

Topology is a set of connected Network Objects which means there is, or it is possible to create a data transport between any two Network Objects in the same Topology, provided there is no policy, availability or technical restrictions.[10]. Topology is the concept for describing a network domain, and Link with multiple sources and sinks for the concept of a local area network.

The top level class *Network Object* is used for being subclassed into components such as *Node*, *Port*, *Link*, *Service* and *Group* ans is the basic abstract class of the schema. It contains a persistent globally unique URI[4] to referrer to a given network, a human representable string and it's version contains a time stamp.[10]

Node describes a generally a connected device, or part of, the network. Doesn't necessarily means a physical device as it may be a virtual device or group of devices (when used in aggregations).[10]

A *Port* defines connectivity from a network Object to the rest of the network. A port is unidirectional and doesn't necessarily correspond to a physical interface. It represents a logical transport entity at a fixed place in the network.[10]

unidirectional means the data is only allowed to flow in one direction

Link describes a unidirectional data transport from each of it sources to all of it's sinks. Links are connected to Network Object's as *isSink* or *isSource* relations. *Link* object can also have a relation to a List of Links (*isSerialCompoundLink*) which describes that the Link represents a path through the network implemented by that (ordered) List of *Links*. This is useful for path finding.[10]

A *service* describes a capability of the network, the base schema defines three services: *SwitchingService*, *AdaptionService* and *DeadaptionService*. Basically a service describes how the behavior can be changed dynamically.[10]

A *switching service* has the ability to create a new *Links* from any of its inbound *Ports* to any of its outbound *Ports*.[10]

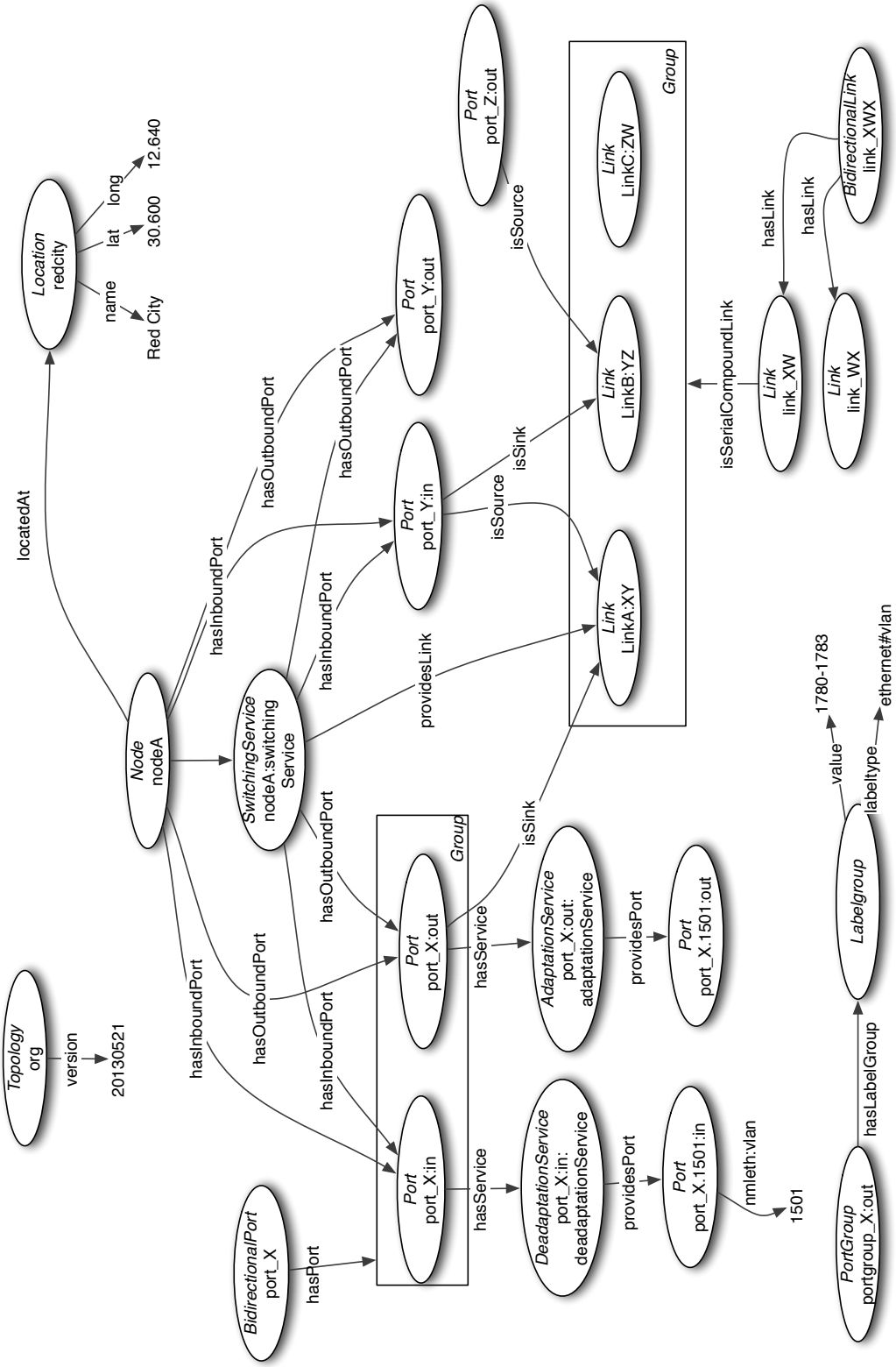

Figure 6: NML relations[10]

AdaptionService describes the capability that data from one or more *Ports* can be embedded in the data encoding of one other *Port*. In other words it describes a multiplexing adaptation function, meaning that different channels (the client layer ports) can be embedded in a single data stream (the server layer port). For example multiplexing several Virtual Local Area Network (VLAN)s over a single trunk port.[10]

De-Adaption Service describes the capability that data of one or more ports can be extracted from the data encoding of one other port. This is basically the opposite/reverse of *AdaptionService*. So it describes a demultiplexing adaption function, meaning that different channels (the client layer ports) can be extracted from a single data stream (the server layer port). For exampling demultiplexing several VLANs from a single trunk port. [10]

Last we have the abstract *Group* object which describes a collection of objects. Any object can be part of a *group*, including another *group*. NML base schema have five different *Group* objects: *Topology*, *Port Group*, *Link Group*, *Bidirectional Port* and *Bidirectional Link*. [10]

Port Group is unordered set of *Ports* and *Link group* is unordered set of Links.[10]

BidirectionalPort is a group of two (unidirectional) *Ports* or *PortGroups* together forming a bidirectional representation of a physical or virtual port. [10] NML examples can be found in the NML base schema document[10], and also my own composed one in the appendix A.

Part III

PROTOCOLS FOR NETWORK PROVISION ARCHITECTURE

4 NETWORK SERVICES FRAMEWORK

As we have NML-WG for NML, there is NSI Working Group (NSI-WG) which is a working group put together to work on a framework that deals with provisioning standards and how to deliver predictable deterministic connectivity services between network domains. They also focus on making sure this works at a global scale. This framework is known as Network Services Framework (NSF).

NSF and it's protocols is again a result of high demand for a consensus standard for doing dynamically connection reservations for network transport resources. Provisioning must be automated in order to scale in a global environment. NSI aim to be the open standard and is a working group in the community wide OGF.

4.1 NSI

NSI v1.0 runs today on GLIF automated GLIF open lightpath exchange, comprised of one or more network devices performing lightpath switching. (GOLE) and have already in 2012 reserved proximally $2.5 * 10^5$ connections[17] in 2012, and this is mostly only NRENs and with the release of NSI v2.0 plans to incorporate the use of using NML descriptions for describing topologies. [9]

NSF describes the architectural elements that manage the service requests while NSI is the interface between NSAs which acts as a glue to exchange NSI protocol messages over the NSI interface to enable end to end transport provisioning in NSI. NSI architecture deals with decoupling of the Service Plane from the Data plane. [8]

NSI is designed for dealing with dynamically creation of circuits (connections) that transit several transport network providers. [18]

The architecture of NSI does not specify transport technologies used within each domain, as it is the task of the Network Resource Manager (NRM) who handles internal provisioning and that makes NSI highly capable of handling multi-layer, multi-domain and multi-service data transport environments and suits good for heterogeneous networks. [17]

As NSI itself does not deal with how it's assigning intra-network resources, the NSA will instruct it's NRM which has the tasks for dealing with how two Service Termination Point (STP)s are actually connected. As mentioned above, NSI is for dealing with inter-domain connections.[17]

NSI describes a network topology with juxtaposition to each one another's network service domains. This is basically the NSI topol-

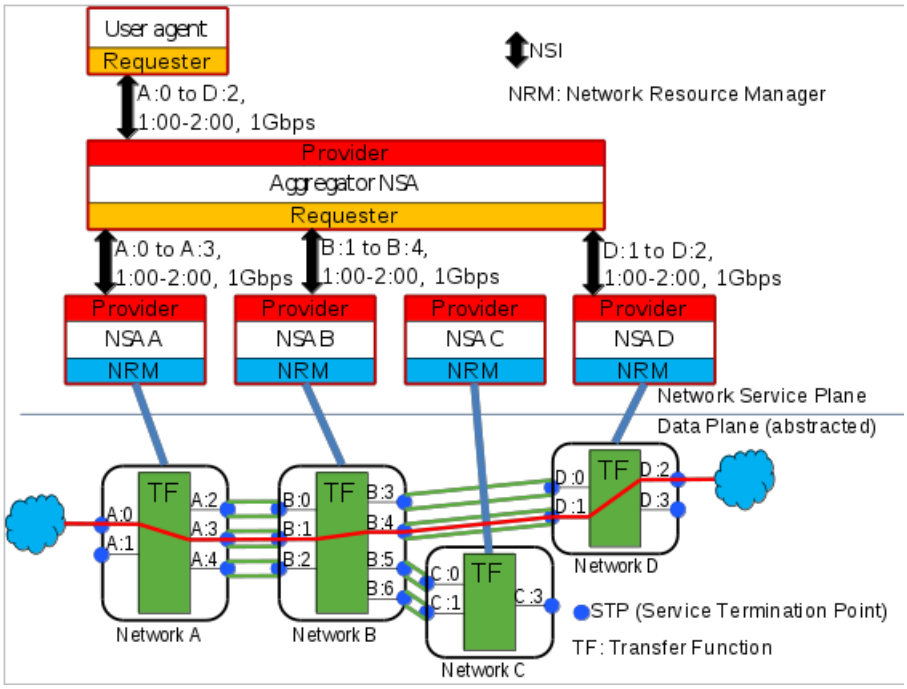

Figure 7: NSI-CS overview[18]

ogy model which is roughly translated into a derivative «resource graph» consisting of resources and stitching relations between NSAs (NSI networks). [17]

4.2 TOPOLOGY

NSI topology model is translated into a derivative resource graph consisting of resources and stitching relations between NSAs as NSI Model assigns ownership of all physical components to one network or the other.[17]

Technology agnostic inter-domain STPs are defined and mapped logically to internal physical components. In NSI the external relations between NSA are SDPs. SDPs simply describes a grouping of two adjacent STPs belonging to different Networks. Also the NSI topology model it is usually to describe the topology with SDPs by hiding all internal structure and only show the peering SDP relations between the networks. [17][8][7]

4.3 CONNECTION SERVICE

NSI-CS is the protocol for handling connection reservations which deals with support of reservation, creation, management and removal of dynamically circuit connections. The physical connections created by NSI-CS v2.0 only support point-to-point but can be either unidi-

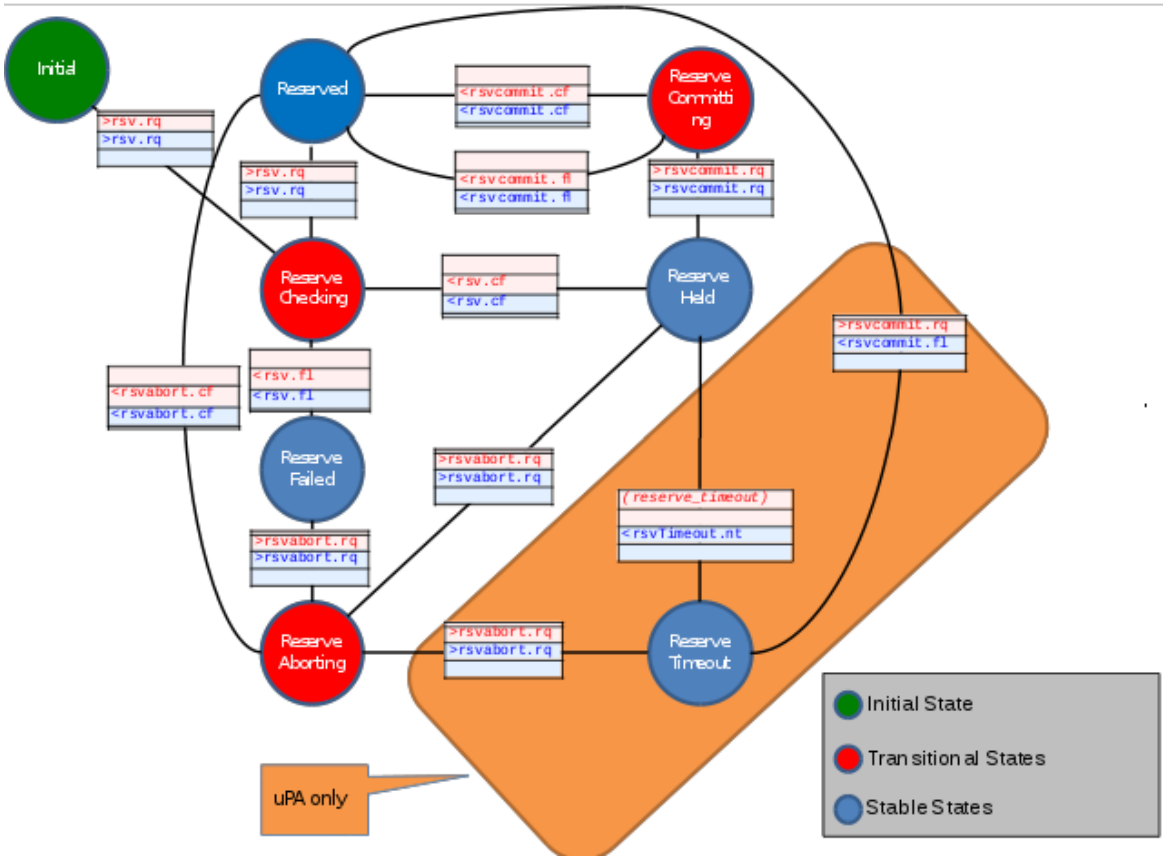

Figure 8: Reservation state machine[7]

rectional or bidirectional. Support for *anycast* will be supported in v3.0.[17][7]

The main logic in NSI is built up of 3 state machines which acts as a virtual map that describes how the protocol events progress a connection through it's life cycle.[17][7]Main one being the reservation state machine (see figure 8) which has the tasks handling the reserve request for a connection request. You specify *from* and *to* together with *start time* and *end time*. Requirements as labels (VLAN tagging) or bandwidth requirement is optional to supply in the connection request. When the request arrives it first checks if the resources are available and if resources are available the resources are held until the second phase arrives which is the commit phase. The requester has the choice of either accepting the connection reservation or to abort it.

4.4 NSA

NSAs acts as service who handles peering relationships. NSA has two main services they provide, it's either acting as a Request Agent (RA) which sends a request to a Provider Agent (PA) or acting as an PA answering requests from RA. While NSA can inherit a number of roles

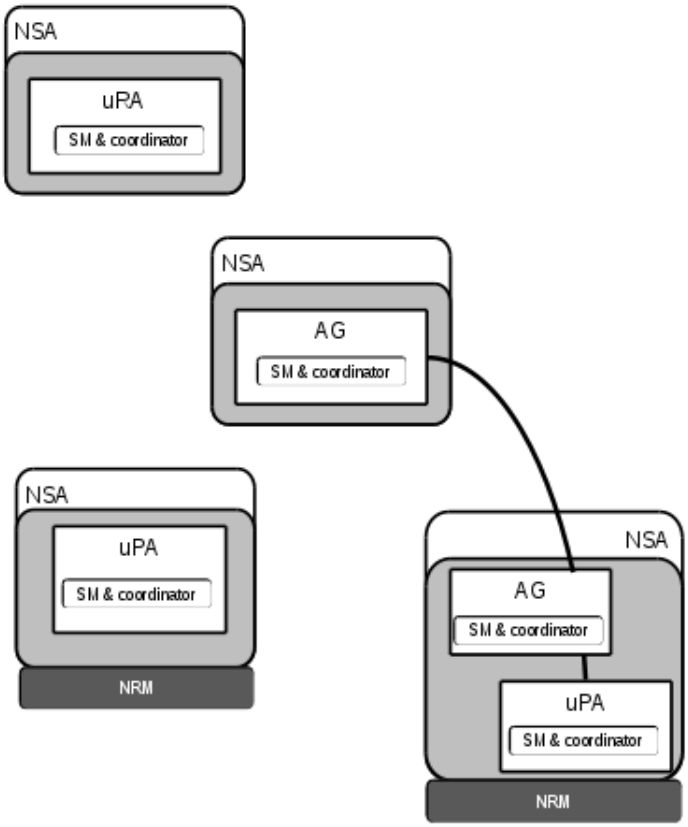

Figure 9: NSA overview[7]

such as: Ultimate Request Agent (uRA) which is the originator of a service request. The Aggregator (AG) which has a subset of children NSA and acts as a aggregator for responses from each of it's children. Or Ultimate Provider Agent (uPA) which is acting as both a RA and PA.[17][8]

Keep in mind NSI only deals with SDPs between NSA so NRMs has the role of managing the Data Plane resources (STPs). This is typically equipment vendor's own network management system.[17][8] [7]

4.5 CONCERNS

A presentation in the OGF mentions a few implications and worries about NSI and asks a few questions. This is from the NEXPReS project which have developed an NSI client which can request paths from multiple NRENs. One of the main concerns is the difficulty of debugging lightpaths, as there is few layer 2 diagnostic tools available, and debugging an dynamic connection with NSI is going to be even worse, and the end-user might not even know which route their path is taking. Since NSI is treated as a 'black box' approach and the way it hides the underlying NRM it is impossible to know if the new request

will take the same path between the same endpoints and traverse the same links and equipment. This means proper debugging has to be done with network engineers on the phone while they can investigate and check their NRM to figure out where the traffic actually flows. [15]

Which use cases is the shared vision for NSI? Will NSI be available for end-users, or will it only be a protocol spoking between network providers? If it will be available for end-users, end-users generally doesn't have equipment which supports the programmability features that make up an NSI connection, but there is ongoing research if OpenFlow[1] will allow users to do the required network configurations in response to NSI state changes. What we don't need is yet another piece of software which connects it self to the router via a telnet prompt. [15]

They also wonder if it will be possible to force the use of low level STP in the top-level NSI request, as NSI can be stacked several layers deep with NSAs acting as an aggregator which again needs to ask other NSAs to create a cross-domain path.[15] I guess it's up to each NREN / ISP to either provide a «super premium» low level STPs in the top level NSI request, if there is demand for such a feature. It is also interesting with the following scenario where two end-users have requested to generate 250 Mb/s of traffic and have been reserved over the same 1Gb/s link. These end-users will not be able to co-exists on a 1Gb/s shared link, due to end-users can only send traffic at two rates: either full line-rate of their network card or zero. This means traffic will burst out in full line rate, and then have periods of complete silence. This means when the micro bursts between users appears in the same phase will throttle the network performance and causing up to 50% packet loss and cause a drift until their out of phase again where the network would perform flawlessly again.[15] The last scenario has actually been observed when NEXPReS project were using e-Very-long-baseline interferometry (VLBI)[2], but it is not a directly problem due to the NSI but a problem end-users should be aware of when it comes to bandwidth guarantees. [15]

«By deploying Premium service, do we want to supplement the Internet best-effort service or to replace it?»[2]

Another concern is this how we want the Internet to be? Will we end up with an Internet where end users not option in for paying for premium based services for access to certain services you will always be met with: «Please wait a minute as we are buffering your youtube video...»? Will we all end up being dependent on circuit reservations for getting a good Internet service? Will the deployment of premium services degrade the best-effort service as we know the Internet as of today? [2]

1 http://www.openflow.org/wp/learnmore/
2 http://en.wikipedia.org/wiki/Very-long-baseline_interferometry

Part IV

DESIGN AND DEVELOPMENT

5 APPLICATION CONCEPT: NETMAP

Browsing and visualize a network topology, should be extendable to suit both use cases for network provisioning and network monitoring. First of all it should be able to view a network topology, in simple words a graph consisting of nodes and it's relations between em.

In network provisioning the tool can be used to view the internal topology and selected which components (STPs) of the network topology should be exposed (SDPs) and exported to the network provisioning system.

For network monitoring you would like to use Netmap as a key monitoring tool and act as a dashboard for important overview of your network. Are any of your links in your network topology congested? Do we detect any topology errors as mismatching network interface speeds between two nodes or any other irregularity? Which VLANs are available over this link between node A and B? Netmap could have the option to be used as both an dashboard and interactive monitoring tool in a Network Operations Center (NOC).

5.1 REQUIREMENTS

Developing on the prototype of visualization of a topology, there were a few important requirements. It should be easily accessible by using a browser, and the drawing should be implemented using Scalable Vector Graphics (SVG) which is an open standard and has unlimited scaling due to graphics are drawn as vectors. It was also important to try make Netmap independent so it could run as a stand alone web application. Another important thing is that the visualization of the network topology requires algorithms for automatic node positions in an aesthetically pleasing way so it is easy to get tactical and logical overview over the network topology.

5.2 TECHNOLOGIES

Netmap uses quite a few frameworks to ease the maintenance of the web application. JavaScript is the only available client scripting language available in the browsers without using plugins that we would like to avoid. Code is written in JavaScript following the module[1] pattern which Netmap is loading using the Asynchronous Module Definition (AMD)[2] pattern which gives clear advantages when devel-

1 http://www.adequatelygood.com/Javascript-Module-Pattern-In-Depth.html
2 http://requirejs.org/docs/whyamd.html

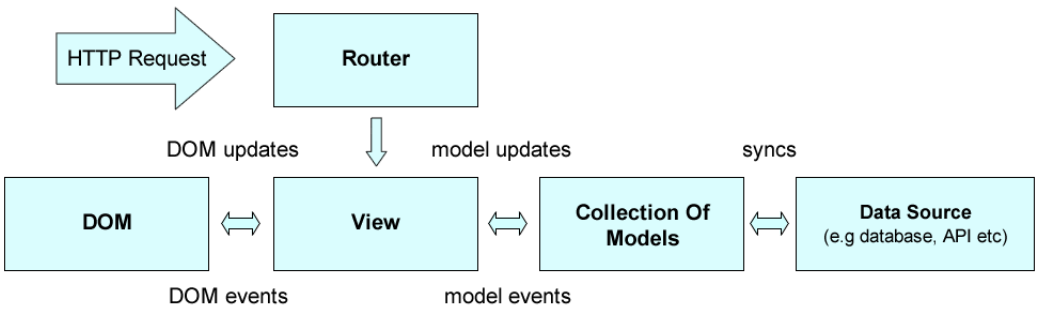

Figure 10: Flow in Backbone.js[16]

oping in JavaScript. Backbone.js[3] helps us making a structure in our web application. It resembles both the Model-View-Presenter (MVP) pattern and Model-View-Controller (MVC) pattern.[16] See figure 10 for the flow in backbonejs applications.[4] jQuery[5] which helps with cross browser compatibility in JavaScript when working with DOM[6] and underscore.js[7] which has a handful of good helpers to more easily write code in JavaScript.

Template resources are also loaded as text resources in requirejs using requirejs's internal text module so they can be declared as dependencies and loaded asynchronously just like any other modules when using the AMD pattern. Handlebarsjs[8] is used for building semantic and logic-less templates effectively without frustration.

SVG[9] is used for generating the visualized and interactive Netmap with the help of D3JS.[10] D3JS provides data binding between the model and SVG Extensible Markup Language (XML) who is attached in Domain Object Model (DOM) which makes it easier to do SVG-manipulations. D3JS also provides a force-directed graph layout algorithm which position the nodes in the graph in an aesthetically pleasing way so the space between all the edges are of more or less equal length and has as few crossings as possible.[11][13]

5.3 NETMAP PROTOTYPE

Main application represents of 3 main views that gets rendered on the main page. This is navigation view on the left side, draw map view in the center and info view on the right side. The views in the sidebars

3 http://backbonejs.org/
4 See "MVP or MVC?" chapter at the reference[16] for details
5 http://jquery.com/
6 http://www.w3.org/DOM/
7 http://underscorejs.org/
8 http://handlebarsjs.com
9 http://www.w3.org/Graphics/SVG/
10 http://d3js.org/
11 https://github.com/mbostock/d3/wiki/Force-Layout

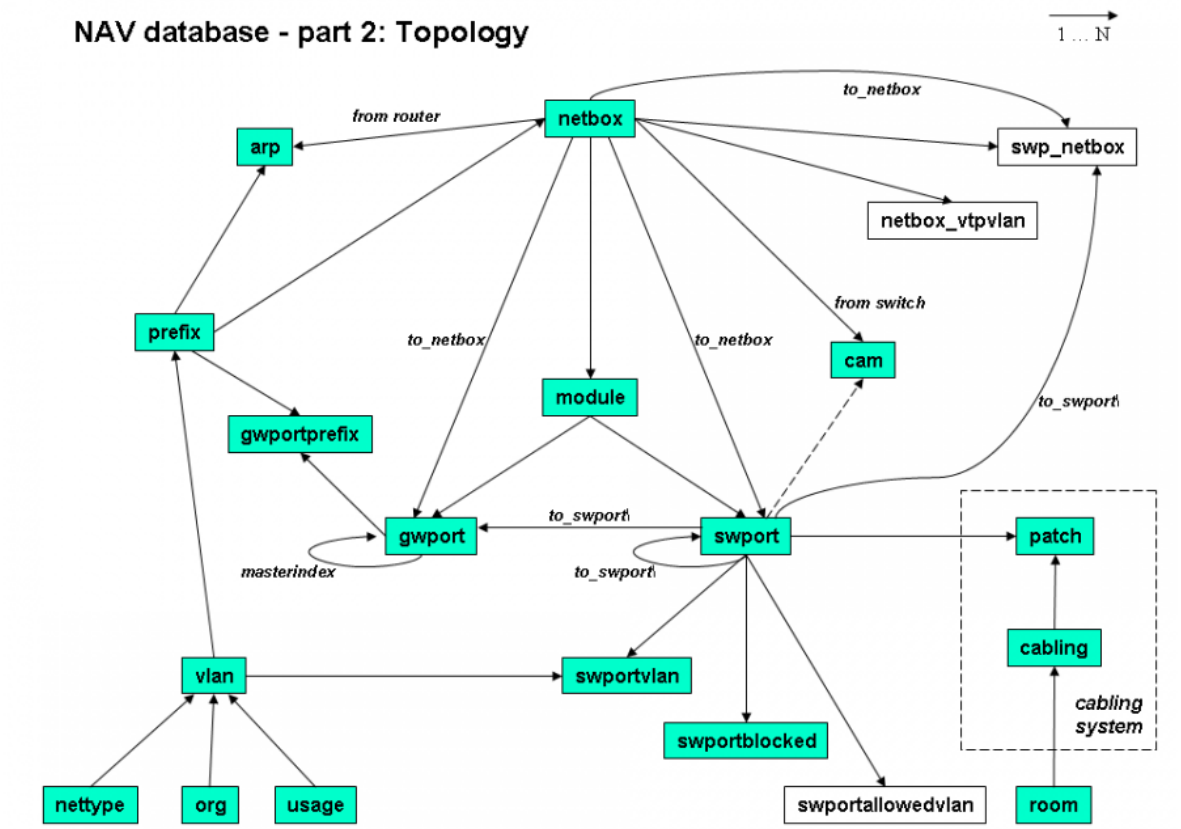

Figure 11: NAV topology RDBMS schema[1]

are widget holders who can hold widgets which triggers events back and forth from the main draw map view. Netmap has a collection of widgets as you can see in figure 15 which can be attached in any of the sidebars. In the center you have the interactive visualization of the network topology.

Since this is a prototype I focused on using NAV as a data provider. Netmap can ask NAV to build a layer 2 and layer 3 graph for it's known network topology. This is represented in the graph as netboxes acting as nodes and shows the links between them.[12] As NAVs topology graph doesn't include VLANs, we extract it from the *SwPortVlan* table (see figure 11) to attach VLAN and other relevant meta data which is useful to display. The flow from NAV to Netmap and a viewable topology looks like figure 12 & 13.

. In the end the prototype ended up with supporting many capabilities such as: displaying link capacity and link load between edges in the graph (see figure 14) , filter of netboxes based on netboxes, filter out orphans[13], search for a netbox in the network topology, con-

12 See https://nav.uninett.no/doc/howto/debugging-topology.html#how-nav-builds-physical-topology-information for how NAV builds layer 2 and layer 3 topology

13 orphans is netboxes who doesn't have any neighbors

Figure 12: NAV topology to NAV Netmap topology

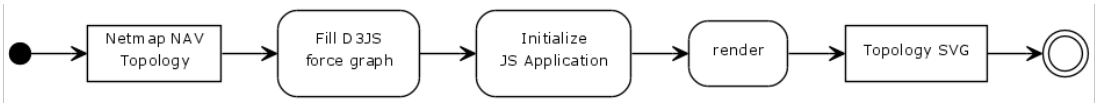

Figure 13: NAV Netmap Topology to visualized SVG

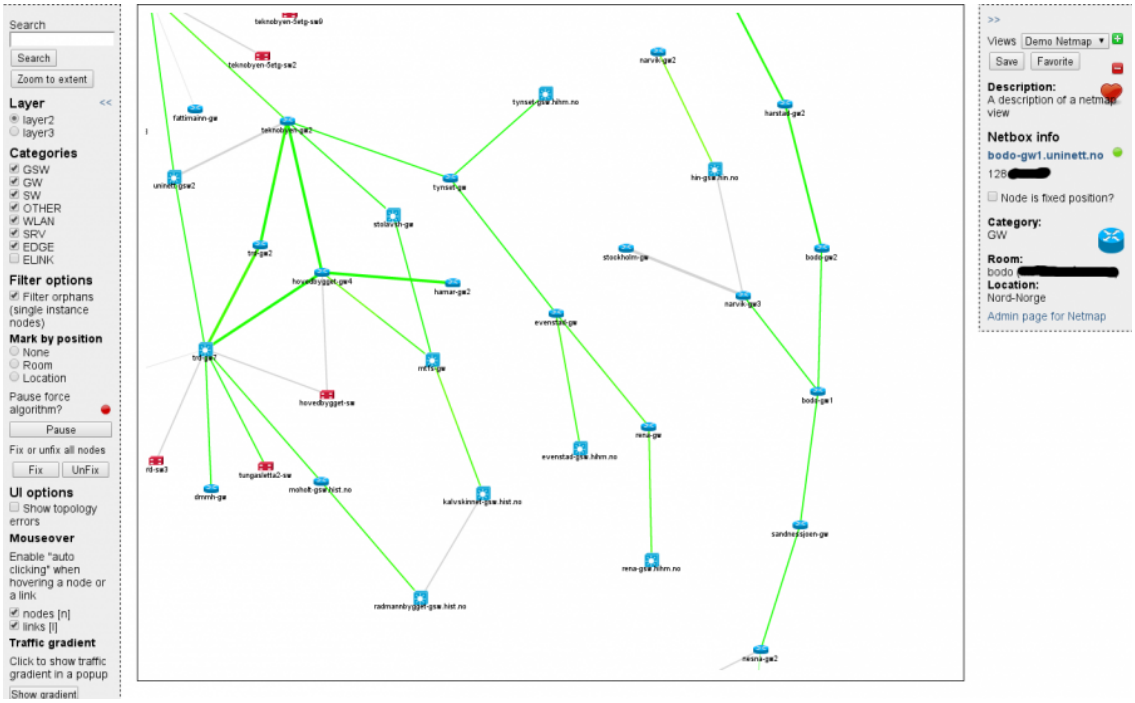

Figure 14: Early version of Netmap, showing link load

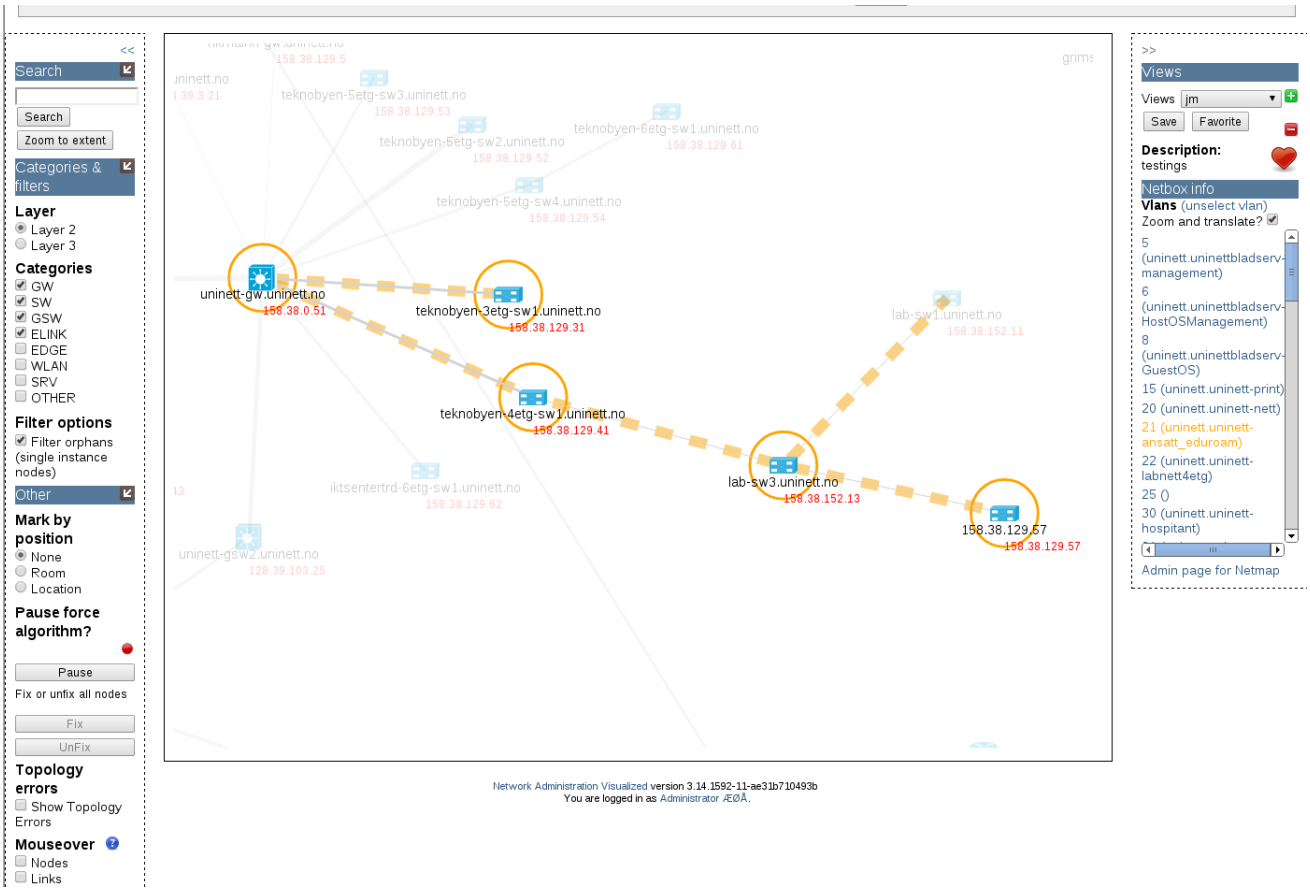

Figure 15: Netmap with VLAN selection

trol of the layout algorithm as it is quite CPU intensive, mark VLAN with zoom and translating into its bounding box (see figure [14]) , drag around and position the nodes at your will and save your views, updating of network topology on given intervals, and even more features which is cosmetic features as full screen support etc. Netmap is extendable as you can make widgets which trigger events for the main drawing view can react on and visa versa.

I also experimented exporting NAV topology as NML, I've included a short demo between 4 nodes (see figure 16). The generated NML you can see in appendix A and is validated against the NML base schema.

```
rockj@luna:~/master$ xmllint --schema nml-base-may.xsd nml_
    example_nav.xml
nml_example_nav.xml validates
```

A quick explanation of the generated NML is as follows: at the beginning we list up the following unidirectional links (edges) between nodes. After this each inbound and outbound port for the interface gets created. If the link contains VLANs there will added an *Adap-*

14 It is missing link load due to the collecting engine was not running during the time of screenshoot

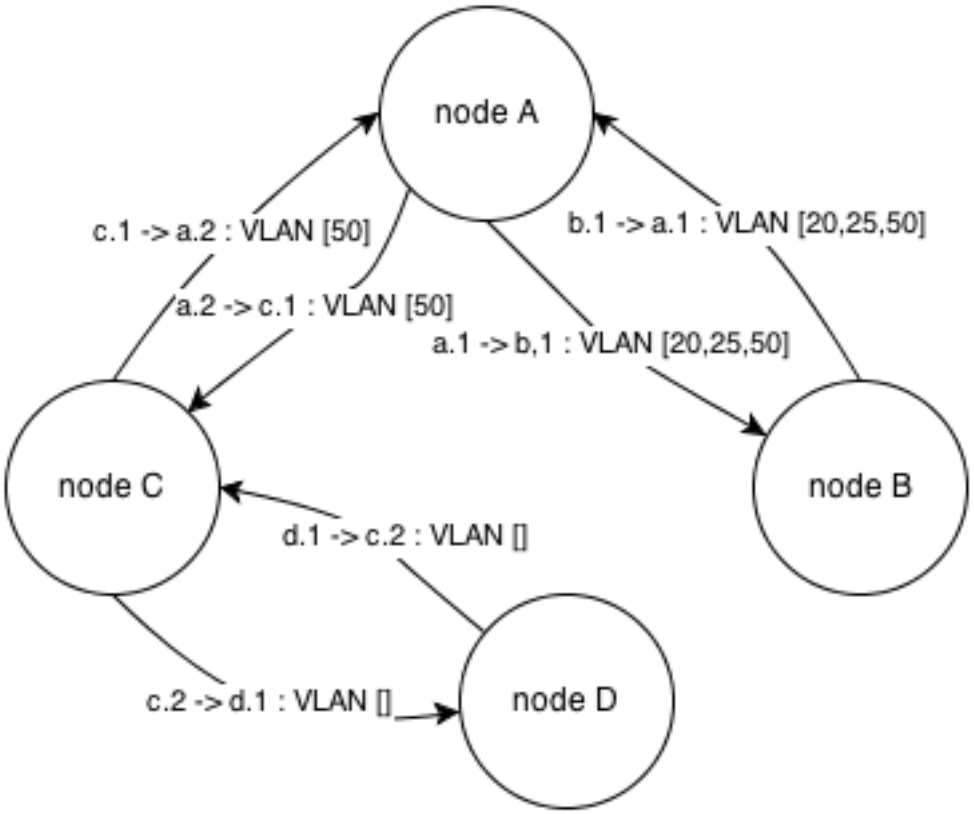

Figure 16: Topology for NML demo

tion/Deadaption service containing logical *ports* who shows this. Relations are added for the ports by using *isSink* and *isSource* relations to show the unidirectional connections between the *ports*.

5.4 SOURCE CODE

The latest netmap prototype you find in the zip-file/cd under *netmap/*[15] and the NML export demo is in *nml_export/*

Netmap lives in *netmap/media/js/src/netmap/* which have the following folder structure[16]:

COLLECTIONS holds a set of models.

MODELS contains application data and logic around the data

TEMPLATES this is where handlebars templates stored and loaded from

VIEWS this is where all rendered views are. Views have a few sub folders as widgets/ which holds all Netmaps widgets. There is also a sub folder for modal/ where modals are stored and info/ contains views that is used in multiple widgets.

15 Make sure the netmap_cleanup bookmark is checked out

widget_mix.js adds shared functionaility to widgets, but currently does the work of making widgets collapsible if the widget is enabled to be collapsible.

router.js is handles request management since netmap is a single page JavaScript application and uses hashbangs & HTML5 History

resource.js is the shared application storage when the application gets initiated. It loads the bootstrap data that is made available from when django renders the *backbone.html* template which resides in *netmap/templates/netmap/*.

views/draw_map.js takes care of using D3JS and does data binding towards the topology graph stored in a backbone model and have multiple updateRender methods for rendering all the visualizations.

main.js adds a global error callback and sets a few external properties before launching the Netmap application.

app.js bootstraps the Netmap application and kicks it off by starting the router

At the server side code resides in the packages *nav.netmap* and *nav.web.netmap* and uses topology data from *nav.topology*. In *nav.topology* it uses *vlan.build_layer#_graph* functions as a base before *nav.netmap.topology* takes care of adding additional meta data and re-adjusts the graph to a multigraph. *nav.topology.d3_js* has exporting code to take an networkx graph and export it as D3JS format, this was mainly used in the first version of the prototype, is still in use for the the second version but the graph gets rebuilt in JavaScript application using the the *D3Force* and *SetEquality* plugins residing in *netmap/media/src/plugins/* . These plugins are useful for when reloading topology data and will also be useful for further work when implenting NML parser in Netmap.

NML export demo you find the test file in *nml_export/tests/unittests/netmap/nml_test.py*[16] using *nml_export/test/unittests/netmap/netmapgraph_testcase.py.*

16 Only outputs XML you can validate manually against the XML base schema.

Part V

CONCLUSION AND FEATURE WORK

6 CONCLUSION

In this theses project, I have searched for a generic and abstract model to describe network topologies supporting both multi-layer networks and multi-domain networks. I've also searched for a protocol for network topology exchange and a protocol for dealing with dynamically creation of circuit connections inter domain. I've also been developing on a prototype for network topology visualization with monitoring features.

Feedback from NAV Internet Relay Chat room tells us Netmap is something networking operators is interested in and find useful to use for getting a overview over their network topology. Users seems to agree with the chosen technologies in the new prototype of Netmap, certainly getting rid of the dependency of requiring a Java applet runtime in your browsers which a lot of users had issues with from the old version of Netmap. There was also some feedback with issues of rendering SVG in their browser, but this was simply solved by telling the user to update their browser. In some rare cases tech administrators could set certain policies in Internet Explorer which disallowed rendering of SVG, this was fixed in a later version of the prototype to inform the end-user about the problem which we also received good feedback on from affected users.

I also spent some time evaluating the technology chosen, and I seem to have picked good tools for the job even though it was a steep learning curve to handle all the frameworks and tools which makes it easier for developing JavaScript applications and Netmap been through a few development iterations, but the job would be a lot harder if they weren't available.

NML I recommend for describing network topologies, given the examples in the NML base schema[10] and my own experimentation of exporting NML (appendix A) managing to deal with multi-layer (VLAN). NML also support extensions so it easy for monitoring tools to include meta data for related link load and traffic data and like vise for network provisioning tools can use NML as a base for sharing topologies across domains using NSI-CS version 2. NML is also a result of multiple other projects supports and agrees that the world need one standard of describing network topologies in an abstract and generic way. NSI which is a part of NSF complements NML as they plan to integrate NML as the standard for exchanging network topologies across domains, and there is paper already been produced about path finding[5] for NDL which NML is inspired from & path-finding

experimentation for NML is on the todo list by the NML & NSI working groups.

NSI architecture does not specify transport technologies used within each domain, since that is the task of the NRM who handles provisioning internally which makes NSI highly capable of handling multi-layer, multi-domain and multi-service data transport environments with NSI-CS and it's other components in NSF.

7 FURTHER WORK

- Netmap currently doesn't use the NML topology model, but since it holds it's own graph format it should hopefully be trivial to write the import parsing function in Netmap. The challenge here is how to deal with displaying a multi directional graph, as there is a challenge of how to visualize a topology without drawing an arc for every VLAN. Imagine a link between two nodes carrying >=5 VLAN, it would quickly turn into a mess and it will be hard for the force directed layout algorithm to position the graph neatly. Netmap deals with this by having it's own data backend extensions in NAV and convert the NAV topology data (multi directional graph) to a Netmap topology data (multi graph). So one of the tasks for implementing NML import in Netmap is to ensure you export NML with *Bidirectional Port/Bidirectional Link* data as well which Netmap can use for importing NML to it's own internal format that is used together with D3JS.

  Using D3Force and SetEquality plugins (available since April 2013 rewrite) should make this easier to implement with newest prototype of Netmap and make it less dependent on NAV

  Since time didn't allow it I didn't have time to make sure it exported data with *BiDirectionalPort/BiDirectionalLink* data in the NML export, but a huge amount of time was spent on making the JavaScript application less dependent on NAV for further work.

  This is one of the primary goals further work should solve so Netmap ends up being a self contained visualization application.

- NAV should fully support importing and exporting av NML, this can come in quite handy when users has questions about their network topology in NAV. We could then import an anonymous snapshot for debugging purposes.

- Add functionality to support network provisioning where you allowed the NOC administrators to pick which STPs you would like to export as SDPs and have this exported topology as NML available by a web service.

  The web service could then again be tested against two instances of *OpenNSA*[1] where it tries to do connection reservations between two topologies, as of this writing the hacker(s) at NORDUnet is working on implementing NML support to support

1 http://git.nordu.net/?p=opennsa.git;a=shortlog;h=refs/heads/nsi2

NSI-CS version 2. Suggestion for why to use *OpenNSA* is because it is written in python and should be familiar for any one who have been hacking at NAV.

- Explore possibilities on how to export Netmap topology view to an zip file containing the SVG and all the graphical resources you can download and view it a SVG capable image viewer which is not the browser. Example: add the visualized topology in a presentation

This would be useful for investigating how the network traffic flows under a distributed denial of service attack on your network.

- Add playback support in NAVs Netmap version for network monitoring so you can record the topology changes and rewind and forward as the topology changes for a given period. Look into possibilities of exporting the playback as an animated image or video.

Part VI

APPENDIX

A

NAV TOPOLOGY TO NML EXAMPLE

```
<nml:Topology xmlns:nml="http://schemas.ogf.org/nml/2013/03/base
    #" id="urn:ogf:network:uninett.no" version="2013-05-31T
    01:17:38.693849Z">
  <nml:Link id="urn:ogf:network:uninett.no:link:unittest.c.nav:
      out:::unittest.a.nav:in">
    <nml:Name>Link from unittest.c.nav to unittest.a.nav</nml:
        Name>
  </nml:Link>
  <nml:Link id="urn:ogf:network:uninett.no:link:unittest.b.nav:
      out:::unittest.a.nav:in">
    <nml:Name>Link from unittest.b.nav to unittest.a.nav</nml:
        Name>
  </nml:Link>
  <nml:Link id="urn:ogf:network:uninett.no:link:unittest.a.nav:
      out:::unittest.b.nav:in">
    <nml:Name>Link from unittest.a.nav to unittest.b.nav</nml:
        Name>
  </nml:Link>
  <nml:Link id="urn:ogf:network:uninett.no:link:unittest.a.nav:
      out:::unittest.c.nav:in">
    <nml:Name>Link from unittest.a.nav to unittest.c.nav</nml:
        Name>
  </nml:Link>
  <nml:Link id="urn:ogf:network:uninett.no:link:unittest.d.nav:
      out:::unittest.c.nav:in">
    <nml:Name>Link from unittest.d.nav to unittest.c.nav</nml:
        Name>
  </nml:Link>
  <nml:Link id="urn:ogf:network:uninett.no:link:unittest.c.nav:
      out:::unittest.d.nav:in">
    <nml:Name>Link from unittest.c.nav to unittest.d.nav</nml:
        Name>
  </nml:Link>
  <nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
      .nav:2:in">
    <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
        isSink">
      <nml:Link id="urn:ogf:network:uninett.no:link:unittest.c.
          nav:out:::unittest.a.nav:in"/>
    </nml:Relation>
  </nml:Port>
  <nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.c
      .nav:1:out">
    <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
        hasService">
```

```
    <nml:AdaptationService id="urn:ogf:network:uninett.no:port:
        layer2:unittest.c.nav:1:out:adaptationService">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#canProvidePort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.c.nav:1:out:25"/>
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.c.nav:1:out:50"/>
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.c.nav:1:out:20"/>
      </nml:Relation>
    </nml:AdaptationService>
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      isSource">
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.c.
        nav:out:::unittest.a.nav:in"/>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.c
    .nav:1:out:25">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">25</nml:Label>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.c
    .nav:1:out:50">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">50</nml:Label>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.c
    .nav:1:out:20">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">20</nml:Label>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:2:in:25">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">25</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.a.nav:2:in:25:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.a.nav:2:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
```

```
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:2:in:50">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">50</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.a.nav:2:in:50:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.a.nav:2:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:2:in:20">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">20</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.a.nav:2:in:20:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.a.nav:2:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:1:in">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      isSink">
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.b.
        nav:out:::unittest.a.nav:in"/>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.b
    .nav:1:out">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:AdaptationService id="urn:ogf:network:uninett.no:port:
        layer2:unittest.b.nav:1:out:adaptationService">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#canProvidePort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.b.nav:1:out:25"/>
```

```
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.b.nav:1:out:50"/>
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.b.nav:1:out:20"/>
      </nml:Relation>
    </nml:AdaptationService>
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      isSource">
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.b.
        nav:out:::unittest.a.nav:in"/>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.b
    .nav:1:out:25">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">25</nml:Label>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.b
    .nav:1:out:50">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">50</nml:Label>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.b
    .nav:1:out:20">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">20</nml:Label>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:1:in:25">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">25</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.a.nav:1:in:25:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.a.nav:1:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:1:in:50">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">50</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
```

```
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.a.nav:1:in:50:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.a.nav:1:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:1:in:20">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">20</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.a.nav:1:in:20:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.a.nav:1:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.b
    .nav:1:in">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      isSink">
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.a.
        nav:out:::unittest.b.nav:in"/>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:1:out">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:AdaptationService id="urn:ogf:network:uninett.no:port:
        layer2:unittest.a.nav:1:out:adaptationService">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#canProvidePort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.a.nav:1:out:25"/>
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.a.nav:1:out:50"/>
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.a.nav:1:out:20"/>
      </nml:Relation>
    </nml:AdaptationService>
```

```
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      isSource">
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.a.
        nav:out:::unittest.b.nav:in"/>
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.a.
        nav:out:::unittest.c.nav:in"/>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:1:out:25">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">25</nml:Label>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:1:out:50">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">50</nml:Label>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.a
    .nav:1:out:20">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">20</nml:Label>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.b
    .nav:1:in:25">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">25</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.b.nav:1:in:25:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.b.nav:1:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.b
    .nav:1:in:50">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">50</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.b.nav:1:in:50:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
```

```
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.b.nav:1:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.b
    .nav:1:in:20">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">20</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.b.nav:1:in:20:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.b.nav:1:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.c
    .nav:1:in">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      isSink">
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.a.
        nav:out:::unittest.c.nav:in"/>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.c
    .nav:1:in:25">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">25</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.c.nav:1:in:25:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.c.nav:1:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.c
    .nav:1:in:50">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">50</nml:Label>
```

```
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.c.nav:1:in:50:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.c.nav:1:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.c
    .nav:1:in:20">
  <nml:Label labeltype="http://schemas.ogf.org/nml/2012/10/
      ethernet/vlan">20</nml:Label>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasService">
    <nml:DeadaptationService id="urn:ogf:network:uninett.no:
        port:layer2:unittest.c.nav:1:in:20:deadaptationService
        ">
      <nml:Relation type="http://schemas.ogf.org/nml/2013/03/
          base#providesPort">
        <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
            unittest.c.nav:1:in"/>
      </nml:Relation>
    </nml:DeadaptationService>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.c
    .nav:2:in">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      isSink">
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.d.
        nav:out:::unittest.c.nav:in"/>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.d
    .nav:1:out">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      isSource">
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.d.
        nav:out:::unittest.c.nav:in"/>
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.d
    .nav:1:in">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      isSink">
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.c.
        nav:out:::unittest.d.nav:in"/>
```

```
  </nml:Relation>
</nml:Port>
<nml:Port id="urn:ogf:network:uninett.no:port:layer2:unittest.c
    .nav:2:out">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      isSource">
    <nml:Link id="urn:ogf:network:uninett.no:link:unittest.c.
        nav:out:::unittest.d.nav:in"/>
  </nml:Relation>
</nml:Port>
<nml:Node id="urn:ogf:network:uninett.no:node:unittest.a.nav">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasOutboundPort">
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.c.nav:1:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.c.nav:1:25:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.c.nav:1:50:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.c.nav:1:20:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.b.nav:1:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.b.nav:1:25:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.b.nav:1:50:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.b.nav:1:20:out"/>
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasInboundPort">
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.b.nav:1:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.b.nav:1:25:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.b.nav:1:50:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.b.nav:1:20:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.c.nav:1:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.c.nav:1:25:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.c.nav:1:50:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.c.nav:1:20:in"/>
  </nml:Relation>
</nml:Node>
<nml:Node id="urn:ogf:network:uninett.no:node:unittest.b.nav">
```

```
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasOutboundPort">
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:25:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:50:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:20:out"/>
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasInboundPort">
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:25:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:50:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:20:in"/>
  </nml:Relation>
</nml:Node>
<nml:Node id="urn:ogf:network:uninett.no:node:unittest.c.nav">
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasOutboundPort">
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:25:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:50:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:1:20:out"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.d.nav:1:out"/>
  </nml:Relation>
  <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
      hasInboundPort">
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:2:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:2:25:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:2:50:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.a.nav:2:20:in"/>
    <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
        unittest.d.nav:1:in"/>
  </nml:Relation>
</nml:Node>
<nml:Node id="urn:ogf:network:uninett.no:node:unittest.d.nav">
```

```
    <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
        hasOutboundPort">
      <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
          unittest.c.nav:2:out"/>
    </nml:Relation>
    <nml:Relation type="http://schemas.ogf.org/nml/2013/03/base#
        hasInboundPort">
      <nml:Port id="urn:ogf:network:uninett.no:port:layer2:
          unittest.c.nav:2:in"/>
    </nml:Relation>
  </nml:Node>
</nml:Topology>
```

BIBLIOGRAPHY

[1] *NAV wiki and documentation*. URL https://nav.uninett.no. (Cited on pages ix and 35.)

[2] Stanislav Shalunov Ben Teitelbaum. Why premium ip service has not deployed (and probably never will). 5 2002. URL http://qos.internet2.edu/wg/documents-informational/20020503-premium-problems-non-architectural.html. (Cited on page 29.)

[3] Mattis Daae. Kikkhullsoperasjon i hd. *UNINYTT*, 4, 2009. URL https://www.uninett.no/sites/drupal.uninett.no.uninett/files/webfm/Uninytt_nr_4_09_LR.pdf. (Cited on page 4.)

[4] Freek Dijkstra. A urn namespace for network resources. draft, 5 2011. (Cited on page 19.)

[5] Freek Dijkstra, Jeroen Van Der Ham, Paola Grosso, and Cees De Laat. Path finding using the multi-layer network description language, 2007. (Cited on page 43.)

[6] P. Grosso, A. Brown, A. Cedeyn, F. Dijkstra, J. van der Ham, A. Patil, P. Primet, M. Swany, and J. Zurawski. Network topology descriptions in hybrid networks. Open Grid Forum, 2010. NDL. (Cited on pages 3, 15, 16, and 17.)

[7] Inder Monga Jerry Sobieski John MacAuley & Chin Guok Guy Roberts, Tomohiro Kudoh. Nsi connection service protocol 2.0. v7, 5 2013. Group Working Draft. (Cited on pages ix, 26, 27, and 28.)

[8] Inder Monga Jerry Sobieski John Vollbrecht Guy Roberts, Tomohiro Kudoh. Network services framework. *Open Grid Forum Document GFD.173, http://www.gridforum.org/documents/GFD.173.pdf .*, 12 2010. (Cited on pages 4, 25, 26, and 28.)

[9] Tomohiro Kudoh & Inder Monga Guy Roberts. Presentation: Nsi v2.0: what can it do for me?, 09 2012. (Cited on page 25.)

[10] Freek Dijkstra Roman Lapacz Jeroen J.van der Ham, J. V. D. and Jason Zurawski. Network markup language base schema. 05 2013. (Cited on pages ix, 18, 19, 20, 21, and 43.)

[11] Roman Lapacz & Aaron Brown Jeroen van der Ham, Freek Dijkstra. The terena networking conference paper: The network

markup language (nml) a standardized network topology abstraction for inter-domain and cross-layer network applications. 2013. (Cited on pages 15, 16, and 17.)

[12] Donald E. Knuth. Computer Programming as an Art. *Communications of the ACM*, 17(12):667–673, December 1974. (Cited on page v.)

[13] Stephen G. Kobourov. Spring embedders and force directed graph drawing algorithms. *CoRR*, abs/1201.3011, 2012. (Cited on page 34.)

[14] James Kurose. *Computer networking : a top-down approach*. Pearson, Boston, Mass, fifth edition, 2010. ISBN 978-0-13-136548-3. (Cited on pages ix, 7, 8, 9, 10, 11, and 12.)

[15] John MacAuley. Presentation: Ogf 37, charlottesville - network services interface: Additional use cases. forum, 3 2013. (Cited on page 29.)

[16] Addy Osmani. Developing backbone.js applications. 2013. URL `http://addyosmani.github.io/backbone-fundamentals/`. (Cited on pages ix, 34, and 38.)

[17] Jerry Sobieski. An overview and demonstration of: Nsi framework, nsi version 2.0 implementations, glif automated gole, 2012. (Cited on pages 25, 26, 27, and 28.)

[18] Inder Monga Tomohiro Kudoh, Guy Roberts. Network services interface: An interface for requesting dynamic inter-datacenter networks. 2013. (Cited on pages ix, 25, and 26.)

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured at:

http://postcards.miede.de/

Final Version as of June 1, 2013 (`classicthesis` version 1.0).