

THE NETWORK AS A DATA TRANSPORTING AND DATA STORING APPLICATION

In partial fulfillment of the requirements for the Master of Science degree

Section Computational Science

University of Amsterdam

Bui Hung Viet

Supervised by Prof. Dr. Robert Meijer

August 1, 2004

Table of Content:

Table of Content:	3
Acknowledgement	5
Abstract	6
Chapter 1: Introduction	8
1.1. Motivation	8
1.1.1. The Concept of Virtual Network Resources	8
1.1.2. Store and Forward in Lambda-Switching Networks	9
1.1.3. Creating Sophisticated Network Aware Applications	11
1.2. Thesis Outline	12
Chapter 2: Web Services and Grid Technologies	14
2.1. Introduction	14
2.2. Web Services	14
2.2.1 Introduction to Web Services	14
2.2.2 Web Service Behavior	14
2.2.3 Web Service Standards	15
2.3. Grid Computing and Grid Technologies	16
2.3.1. Layered Grid Architecture	17
2.3.2 The Open Grid Services Architecture	19
2.3.3 OGSF and WSRF	20
2.3.4 Globus Toolkit	20
2.3.5 Commodity Tool Kits	21
2.3.6 GridFTP and Reliable File Transfer Service	22
2.4. The Potential of Web and Grid Services	23
2.4.1. What Can One Do If One Has These Technologies?	23
2.4.2. Why Is It Very Intriguing If One Has Things Online?	24
Chapter 3: Lambda Networking	27
3.1 Introduction	27
3.2. Lambda Networking	28
3.3. Authentication Authorization Accounting Concept and Implementation	30
3.4. The Current State of Lambda Networking	33
Chapter 4: Storage Service and Network Service Model	36
4.1. Introduction	36
4.2. Virtual Network Resources in Internetworking and Telecommunication Field	36
4.3. Storage Service Model	37
4.4. Store and Forward Network Service Model	39
4.4.1. Services Provided by the Broker	40
4.4.2 Central Controlling Broker Model	42
4.4.3. Using Storage Service	44
4.4.4 Other Controlling Broker Models	47
Chapter 5: Network Service Design and Operation	48
5.1. Introduction	48
5.2. Storage Service Design	48

5.2.1. Storage Service Development.....	48
5.2.2. Grid Service Approach	49
5.2.3 Services Provided by Storage Services.....	50
5.2.4 Web Service Approach	52
5.3. Service Provided by AAA	54
5.4. Network Service Implementation	55
5.4.1. Experiment Scenario.....	56
5.4.2. What does experimental result say?.....	60
5.4.3. Grid Serviced vs. Web Services.....	60
5.4.4. Mathematics Perspective of Store-Forward Lambda-switching Network.....	61
Chapter 6: Conclusions and Future Directions	63
6.1. Conclusions.....	63
6.1.1. The Concept of Virtual Network Resources Opens an Easy Way to Create to Meaningful New Networking Facilities and Applications	63
6.1.2. The Store and Forward Network Service Is a Meaningful Extension to an Optical Network.....	63
6.3. Future Directions	63
6.3.1. Technology Improvements	64
6.3.2. New Virtualization Concepts.....	64
6.3.3. The Virtualization of Resources Outside the Field of Information and Communication Technologies	65
Appendix.....	69
References:.....	71

Acknowledgement

Time has gone so fast. It is coming to the end of the master program course, which started two years ago. In October of last year, Peter Sloot introduced me to Robert Meijer to see if I was willing to do the research with him. Then my master-thesis-project story began since.

First, I would thank to my supervisor for his constant supervision and inspiration as well throughout the time. I also would like to sincerely thank to my collaborator Bas van Oudenaarde, who supports and provides for me with technical aids as well as useful scientific contribution for my work.

Many thanks to Peter Sloot for his permission and encourage so that I was able to carry out a thesis topic albeit that is in many ways beyond the scope of Computational Science.

Besides, I got a plenty of ideas and help from my friends, people in both SCS and AIR groups. No words suffice to express my respect to them.

Finally, a great deal of thanks to my parents for giving me a life, fostering and for my happiness to live my life passionately and completely.

Abstract

In this report, we introduce the concept of virtual network resources and then apply this concept to deal with the real and specific problems in the field of lambda switching. Lambda switching – the switching of the light in one optical fiber to another – is from a networking point of view at a rather primitive technological stage. The creation of an optical path through a series of switches is either borrowing ill-fitting technologies from the Internet protocol suite or done by hand. Better suited technologies from telecommunications, especially SS7 signaling, are unavailable yet. On the other side, conceptual and technological advances in Computer Science may pose an alternative and intriguing way to deal with the path-setup issues in lambda networking. The concept of virtual network resources applied to lambda networking means that one controls an optical network element with a computer and that one exposes the functionalities of the switch as a Web service on the Internet. Apparently, one should secure the network resources from malicious use through the Internet. As this report shows, the virtual network resources concept does allow the setup of complex optical paths and, most promisingly, allows creating the new networking applications.

This report investigates and ultimately demonstrates the power of this approach by integrating a network service to a storage service into a store and forward network—something that is with neither the Internet protocol concept nor the telecommunication-signaling concepts are foreseen and easy to realize.

Conceptually, a virtual resource can be thought of as a control part of the real resource. This control part consists of a control of the functionalities of the real resource and an interface to connect with resource users, which enables the safe Internet access to the resource.

Virtual resources facilitate the use of real resources. It is possible to use them easily without knowing in detail the (physical, computational, topological ...) nature of the real resources. Furthermore, user is able to combine the resources to create new applications. In principle, some of these applications govern the interactions of distributed physical apparatus – creating a kind of a new machine. In this report, we apply such ideas to network elements and storage devices.

We focus on studying the concept of virtual network resources (resources of telecom or computer networks). We show that by applying this concept, some, if not most, problems (e.g. bandwidth capacity management) of integrating and controlling network resources can be tackled. Often, it is by allowing/supporting a multitude of specific solutions.

In practice, the concept of virtual network resources applied to lambda networking means that one controls an optical network element (switches, routers) with a computer and that some or all of the functionalities of those elements are exposed as a Web (or Grid) service on the Internet. Clearly ample attention must be paid to Authentication, Authorization, and Administrative issues (AAA), for more details of AAA see [22]. Specifically, this report investigates and ultimately demonstrates the power of the virtual

network resources by an application of integrating a network service and a storage service into a store and forward network. In the virtual network resource approach, we will demonstrate a quite effortless undertaking. Yet, it is proven for the concept. In some more traditional approaches, if one would follow, one has either to embark on the Internet protocols or to extend the telecommunication signaling concepts. In the application mentioned above, those approaches are much more difficult to implement than our approach.

Adding storage to a high capacity optical network may solve some practical problems too. An optical switch has a rather limited switching capacity as compared to a traditional telephone switch. Huge switches can at most accommodate about 1000 optical fibers. This means that in practice only handful fibers at most are connected to inter-connect optical switches. Fibers are assigned temporarily to users and are loosely detached from interconnect optical switches as soon users no longer use those fibers. Therefore, it is expected that in practice the reservation of optical networking capacity to popular destinations will result in a better use of the original optical network.

By adding storage elements to the network, the network is able transport some data even if an end-to-end connection is not possible at a certain point in time. If data were temporarily stored at some location, it would be eventually transported to the final destination once the necessary connection is available.

In this report, a store-forward network service model is introduced, which applies the concept of virtual network resources to make it conceptual and technically easy to integrate and coordinate the complex interactions between applications, optical networks, and storage devices. The infrastructure of the store and forward network service model are based on the current (hardware and software) infrastructure of lambda-switching network one the one hand and Web and Grid service technologies on the other hand.

Besides, in spite of not giving a complete proof that the performance of the store-forward lambda-switching network is higher than the current lambda-switching with respect to both experiment and mathematics it is suggested that using this new network is something better than the original network.

Ultimately, we also point out that the concept of virtual network resources can be extended many other areas.

Chapter 1: Introduction

1.1. Motivation

1.1.1. The Concept of Virtual Network Resources

A software object can represent for a network device. By accessing that surrogate object, network user can make use of the corresponding network element. The idea of viewing a network element as a software object is not new. New is that with the arrival of Web (and Grid) service technologies it becomes viable to do so. It is in fact a quite natural approach that a designer wants to abstract away hardware by covering software as its interface, which exposes some functionalities of the hardware.

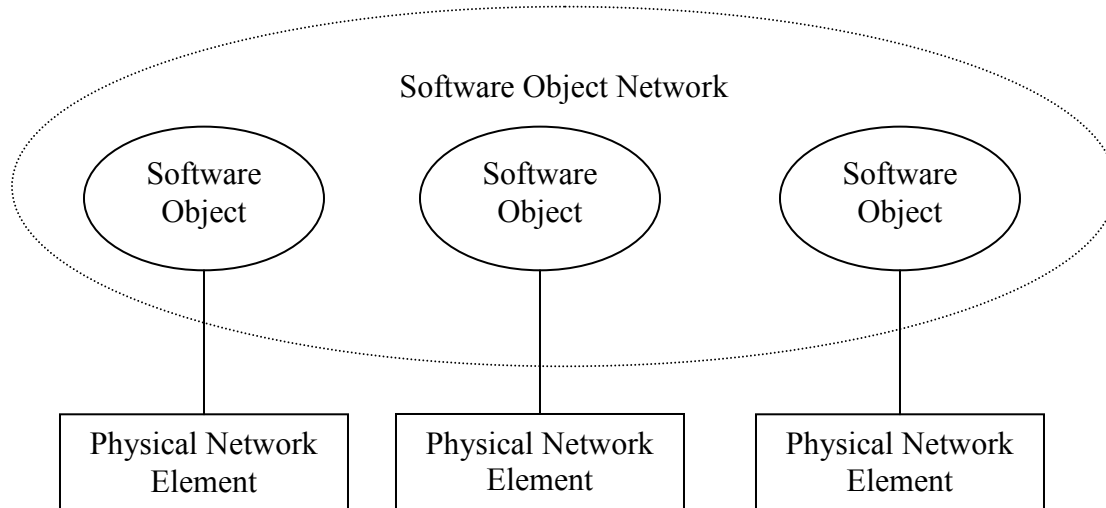


Figure 1.1: Network Element as Software Object

To some extent, a software object can be seen as a virtual network resource because it represents for some physical network element, which is certainly considered as network resource. However, the concept of virtual network resources is somewhat more abstract and formal. In the scope of internetworking and telecommunication, software object is preferably used while the concept of virtual network resources refers to general sorts of network resource in many different areas. It also means that the concept of virtual network resources can be observed widely. This approach came up naturally, especially as a resource itself is so complex that it should be controlled by the corresponding virtual resource and thus abstracting away the complexity of direct controls of that resource. In design, virtual resource should be constructed such a way that the use of a virtual resource is much simpler than the use of the real resource.

Figure 1.1 suggests that software objects, which represent for network elements, can be grouped by some software object network to do jobs.

1.1.2. Store and Forward in Lambda-Switching Networks

Lambda-switching technologies have been developing actively at AIR group in collaboration with several different organizations and universities in the world. In that group, many researches have been conducted in many aspects such as the concept of generic AAA, optical networking for Grid applications. Here AAA is in fact a technology that also exposes the functionalities of a switch as a set of Web services. At the time, this representation of a switch was seen as a mere technical issue; AAA was used in combination with a switch abstraction object to deal with authorization of switch usage – not yet as a new concept. To some extent a real exploration and exploitations of the virtual network resources concept was never conducted thoroughly. Moreover, Web and Grid services are merely used as an easy way to control light paths. The integration of those technologies into software programs used the network were not foreseen. It is hard to get accustomed to the insight that a network elements and even entire network can be treated via their software representations as the software objects so that one is able to integrate into applications.

Lambda switching is the state-of-the-art technology developed on optical infrastructure. The optical network used lambda-switching technology is called lambda-switching network. One of the advantages of this network is that it enables data to be transferred between endpoint applications at gigabit speed over very long distance that is quite difficult to do on the same optical infrastructure with different technologies. For this reason, this network is becoming more and more important especially to server for high bandwidth applications. Developing lambda-switching technologies is also a part of DataTag project. In essence, a switched lambda-switching network is similar to the typical connection oriented networks like ATM network or telephone network. A lambda-switching network is formed by optical switches (Diamondwave developed by Calient Network). However, the differences of this network in comparison with other networks can be attributed to their signaling protocols.

To make use of the network, user has to be able to setup a light path connected two endpoint applications. For example, the Client at UvA has 64 ports, meaning that it allows at most 64 fibers pairs (bidirectional) or 128 single fibers (unidirectional) to plug in. Hence, it can switch signals between at most 64 fiber pairs or 128 single fibers. In the current lambda-switching network (the one we experiment on), there are two Clients (one at Amsterdam with 64 connections and one at Chicago with 128 connections). In principle, when a user is accommodated by the network (authenticated and authorized), the connection is a physical connection between the endpoint applications.

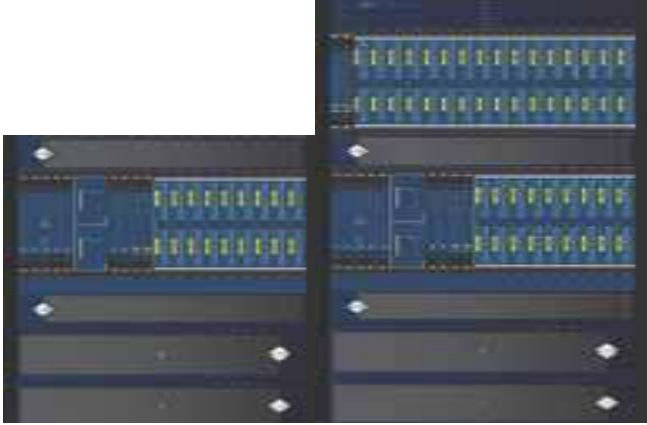


Figure 1.2a (left): DiamondWave™ PCX128
 Figure 1.2b (right): DiamondWave™ PCX256
 Pictures from Calient Network (www.calient.net)

Obviously, the number of few fibers that interconnect switches in a typical optical network will result in a finite blocking probability (see figure 1.3). By introducing storage facilities one may not need an end-to-end connection between the endpoints but only a part of it at the time. In some cases, storage may improve the availability of the network. The details of the storage service are explained in chapter three and four.

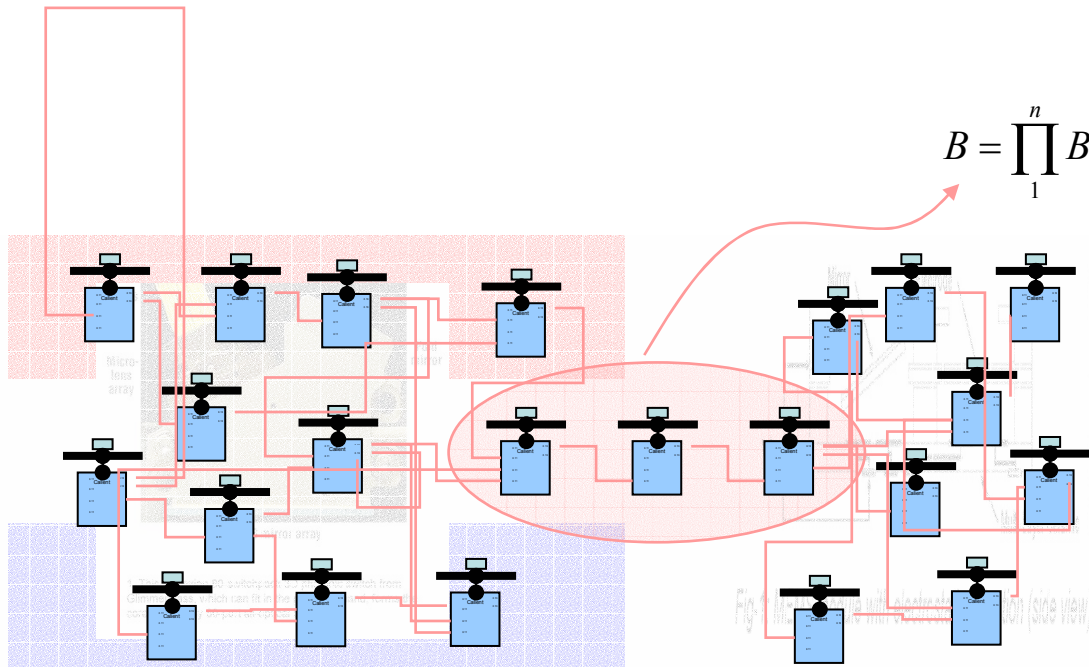


Figure 1.3: Lambda-switching network is deemed to serve as a backbone patch panel with almost unlimited bandwidth but with finite blocking probabilities.

1.1.3. Creating Sophisticated Network Aware Applications

Conceptually, one can represent a network device by a software object. Furthermore, one can create conditions such that those objects are connected to the Internet. Hence, it is possible to create complex interactions between devices by creating interaction between their proxy objects. Coming up with that idea, we apply both Web service and Grid technologies to implement a network aware application, supporting for the concept, that is, by enabling the network with service environment, switches, and storages are integrated. Moreover, it opens possibility that all required network resources can be integrated, providing a unique service for users. In our scenario, the network service model is a combination of different network resources to serve users to transfer data conveniently and efficiently.

Figure 1.4 shows an idea of how network elements like storage and switch can be integrated intuitively. They are all represented by software objects, which are connected by a network. More details of integration will be discussed in chapter 4 and 5.

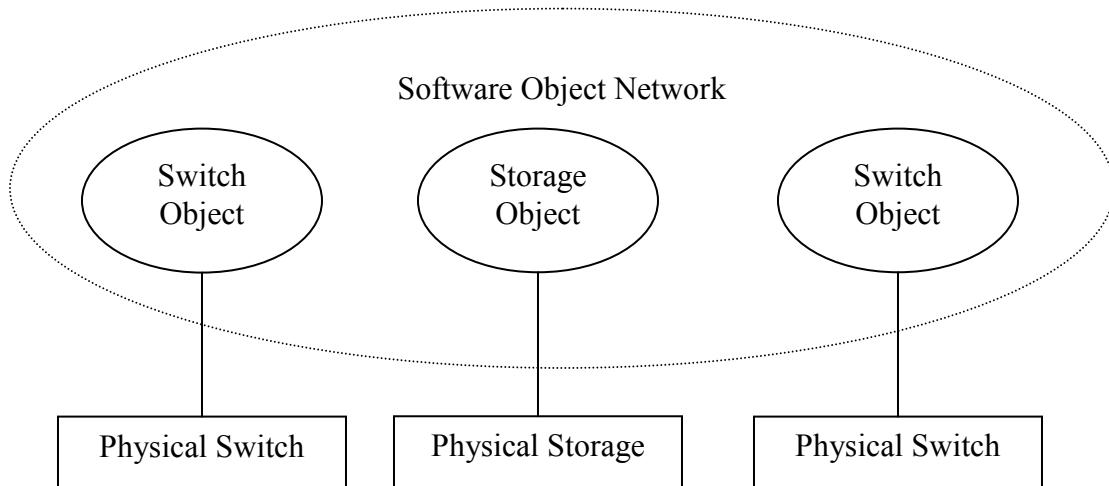


Figure 1.4: Application Integration

Next, an example of network device seen as software object is conducted to illustrate for the concept of virtual network resources. The figure 1.5 depicts the Calient (optical switch) controlled by software (PIN/PDC and PPBAC, see more in 3.4). The details are described in chapter three, however one can see how Calient fits with the concept of virtual network resources.

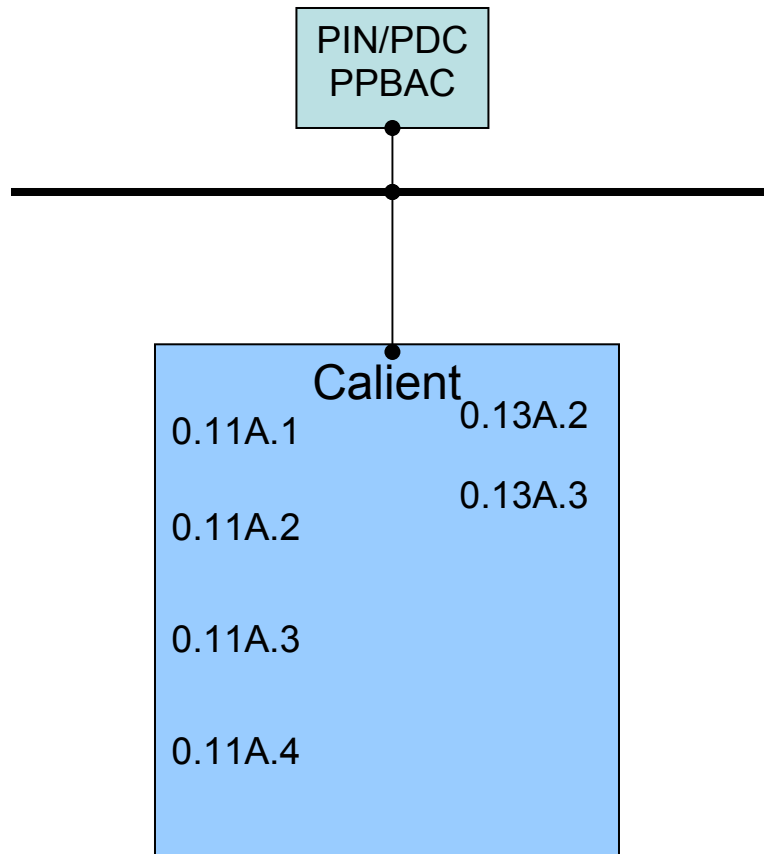


Figure 1.5: Calient Switch and Control Software Fit with the Concept of Virtual Network Resources

Legend: PIN: Photonic Inter-domain Negotiator, PDC: Photonic Domain Controller, PPBAC: Photonic Policy Based Access Controller

1.2. Thesis Outline

- In Chapter 2, Web and Grid service technologies are described with emphasis on architecture, Java Web services, GridFTP and PFTP that are used in the proof of concept implementations. More importantly, the relation between these service technologies and the concept of virtual network resources are given.
- Chapter 3 introduces about lambda networking. It focus on describing the network topology and technologies using on the current lambda-switching network such as AAA and PIN.

- Chapter 4 is elaborates on the network service model, covering the layered architecture, storage service model, broker model, the integration, and operation of network service as well.
- In chapter 5, the implementation issues of the network service model on basis of the storage service are elaborated. Furthermore, proofs of concepts implementations are evaluated.
- Chapter 6 first summarizes the work and suggests applications of the new virtual network resources concept. Last but not least, possibilities for future R&D are discussed

Chapter 2: Web Services and Grid Technologies

2.1. Introduction

This chapter discusses about two cutting-edge distributed technologies, namely Web service technologies and Grid technology. In this thesis work, both technologies are deployed in order to prove our concept. In fact, both technologies have the same conceptual background. However, Grid technologies seem to be the most advanced. As it turns out that was not a guarantee for a successful deployment and in some cases, we needed to fall back to Web service technologies. Although Web service technologies is a good candidate in deploying the experiment, as can be seen later, Grid services technologies is ultimately the best choice, particularly for Grid applications thus being described quite intensively. In addition to that, this chapter also mentions about the convergence of these technologies in near future.

2.2. Web Services

2.2.1 Introduction to Web Services

"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards." [6]

Intuitively, a Web service can be thought of as a way of accessing resources via the Internet (identified by URI). Distributed technologies and systems have been received a lot of attention from computer scientists in the both scientific and commercial areas. Along with Web services, other distributed technologies such as DCOM, CORBA, and DCE...have been developing rapidly and continuously to the demand for distributed applications. Lately, Web services appear to the dominant technology in developing heterogeneous and large-scale distributed systems. However, surprisingly few are available via the Internet. This means either that most Web services are not meant for general usage or that the technology is somewhere near the top of its hype curve.

2.2.2 Web Service Behavior

Figure 2.1 draws a typical operation of Web service. In essence, a Web service representing for one or more sorts of resource does two main operations: advertising itself and serving requester. In turn, requester can find information about Web services and then use them by sending and receiving messages. All operations performed are based on standards, making it easier to decentralize, independent among all parties in distributed system. All service providers, requester, and registry can be developed on different technologies however, using the same standards, which will be introduced later.

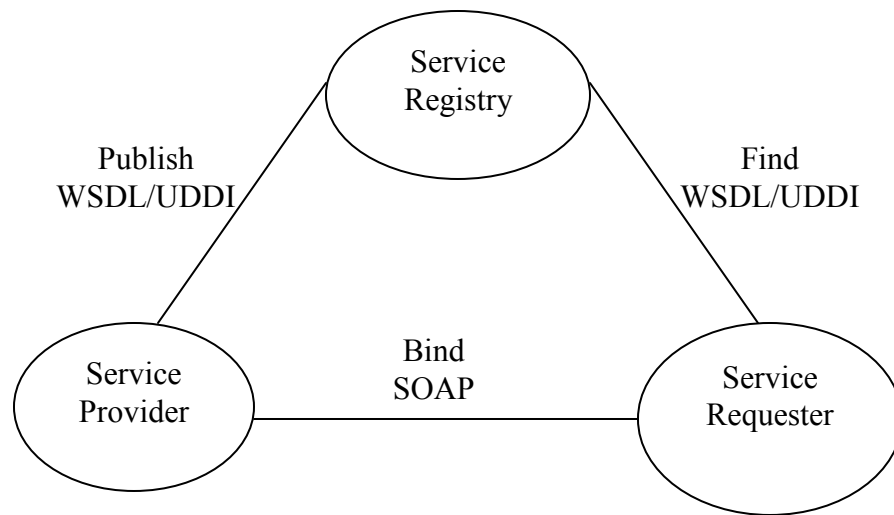


Figure 2.1: Web Services Model

2.2.3 Web Service Standards

The crucial power of Web service technologies is that it relates directly to Web-server technologies. A Web-server can be used in the transfer and execution of a service request. Since Web-server technologies are safe, reliable, sophisticated, cheap, and easy to integrate with specific back-office technologies, Web services are the technology of choice to integrate applications over the borders of corporative domains.

- **eXtensible Markup Language**

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Unlike HTML emphasizing on visualizing information, XML emphasizes on defining structure to describe information. Derived from Standard Generalized Markup Language (SGML), XML is plaintext, readable, and has a straightforward syntax. A part or the whole content of an XML document can be encrypted. XML syntax is simple because that endpoint communications find it easy and fast to pack and parse information by XML. XML is extendable; it gives a software engineering handle to create XML documents specific for a given application. To do so one deploys the XML Schema Definition Language, a language for describing and constraining the content of XML documents. For each sort of application, based on XML schema, XML exchanging form on network service called XML instance would be generated strictly following the defined syntax.

- **WSDL-Web Service Description Language**

As mentioned above, WSDL defines a XML schema specifically designed to describe Web services. Each Web service processes its own WSDL file, enabling all requesters using this service to understand semantic of that Web service.

- **SOAP-Simple Object Access Protocol**

SOAP is a (not so) simple protocol using transport protocols such as HTTP, FTP, or RPC to transport XML messages to object. Its main task is to wrap XML messages (generating based on WSDL) and then bind them to transport protocols. SOAP is the mean for requester access Web services and retrieves information. Both questions sent from requester and answer issuing by service provider are done by SOAP.

- **UDDI-Universal Discovery Description and Integration**

This standard is a set of mechanisms allowing service providers to advertise the services through APIs and requesters to find the services they need through key words.

- **Java Technology and Web Services**

Java Web Service Developer Package (JWS DP) in The Java 2 Platform, Enterprise Edition (J2EE) can be used to develop Web service applications. It supports developer in designing, deploying, and accessing Web services. It also aids the developer by handling most if not all interactions with XML documents. Additionally, the package supports features such as security, distributed transaction management, and connection pool management as well. Because of such features and its mature implementation, JWSD turned out to be a reliable technology to create the software for our experiments with service oriented network concepts.

2.3. Grid Computing and Grid Technologies

"Grid computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative application, and, in some cases, high-performance orientation." [11] In other words, by using Grid technologies, distributed sites are able to coordinate, collaborate and sharing their resources. Grid Technologies are also considered to extend the Web services technologies. The concept of the Grid first appeared more than ten year ago coming from the demand for building distributed systems with large-scale resource sharing and innovative applications. Recently, Grid technologies that are implementing extended Web service concepts being regarded by the scientific community as the technology to integrated applications over the internet. There are, however, a variety of Grid systems and Grid technologies. The Open Grid Services Architecture (OSGA) and WS-Resource Framework, which has replaced for the Open Grid Services Infrastructure (OSGI), are seen as the de facto of the Grid. The Globus Toolkit is the closest implementation of those specifications; *"a community-based, open architecture, open source set of services and software libraries that support Grids and Grid application."* [8] As an inevitable trend, Grid systems and Grid technologies tend to conform and converge to OGSA and WSRF (formerly was OGS I).

Virtual Organization is defined as follow: *"Virtual Organization (VO) is a set of individuals and/or institutions defined by such sharing rules form what we call a virtual organization."* [11] VO motivated for the development of Grid technologies. In deed, a framework which is able to integrate all VOs to form a single distributed system which make VOs possible not only to access to computers, software, data and resources but

also to coordinate multi resource for the use of computation. In order to achieve that shared goal, that framework ought to fulfill the concern and requirements such as the authentication, authorization, resource access, resource discovery and so forth.

Here, some examples are discussed to clarify the fact that current distributed computing technologies do not address the requirements that need for establishing VOs. The Internet is designed to communicate enormous sites so that they are easily able to exchange information but it is lack of a sufficient mean to coordinate multiple resources distributed over many sites for the use of computation. The second example is about CORBA and EJB. Those object based distributed systems merely serve for a single organization in terms of sharing resources but for multiple VOs, they have to encounter with lots of drawbacks such as the firewalls or the various access resource policies. Such technologies are used to meet those concerns, requirements are named Grid technologies, and at the same time, the concept of Grid system is formulated. Despite the fact that there are many forms of Grid system and each of them have its own properties; the Grid commodity believes that there are many features in common. Hence, open Grid architecture has been adopted. It is worth noting that the Grid and Grid technologies are the complementary concepts for current technologies rather than replace them. Moreover, the Grid and Grid technologies reuse many current technologies and, they in return contribute to the development of those current distributed technologies.

In practice, we experienced rather many problems with the implementations of the Grid technologies. Partly these problems originated from the fact that the implementations were new and the extensions to the standard Web services are complicated.

2.3.1. Layered Grid Architecture

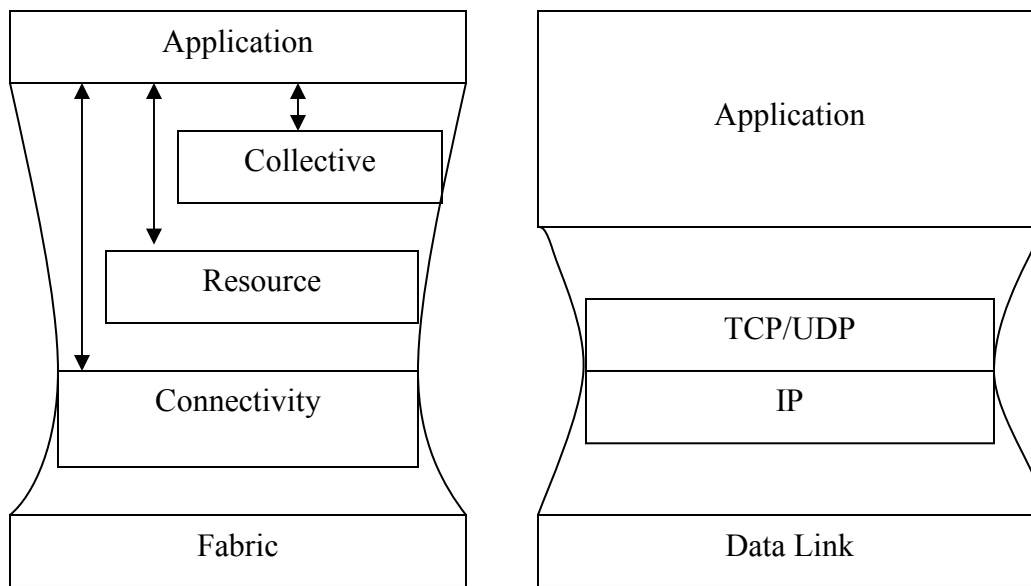


Figure 2.2: The Layered Grid Architecture borrows the bottleneck concept from TCP/IP

In order to reduce the complexity and to be able to inter-operate among Grid systems, the architecture of the Grid is layered into five parts that have different functions and properties. Figure 2.2 shows the correspondence between Grid Protocol Architecture and Internet Protocol Architecture.

The open Grid Architecture can be defined as a set of open standards (inter-Grid protocols), following the hourglass model, which means the Resource and Connectivity protocols act a role as the neck in hourglass model. The aim of this design is to point out that the Application, Fabric layers can be a diversity of behaviors, resource types, and underlying technologies but protocols at neck should be small and compact to facilitate the sharing of individual resources. All layers are briefly outlined as following:

- **Fabric**

Like the data link, this layer provides the Grid with resource to connect with local resources, making use of them. Resources in that layer could be computational resources, storage systems, catalogs, network resources and sensors while a local resource could be either physical or logical such as a distributed file system, computer cluster etc. In principle, the layer resources have to implement the inquiry mechanisms. In short, the fabric layer can be thought of as a bridge between local resources and the Grid.

- **Connectivity**

The main role of this layer is to make communication and secure smooth. Currently, the communication mainly relies on the set of protocols TCP/IP but may change in future. As

with communication, authentication also bases on set of standards, which are commonly used in the Internet.

- **Resource**

This layer defines a set of protocols (including APIs and SDKs) for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on individual resources. By using these protocols, Resource layer is able to invoke Fabric layer functions to access and control local resources.

- **Collective**

The Collective layer coordinates multiple resources regarding a variety of sharing behaviors without placing new requirements on the resource being shared.

- **Application**

An application is formed by the sequence of calling services over layers. An application is capable of communicating with lower layers (not necessary the right lower) by invoking to the APIs and SDKs of those layers. Then, APIs and SDKs would themselves call the services of the corresponding layer.

2.3.2 The Open Grid Services Architecture

As mentioned above, the emergence of VOs gives the birth for the Grid. Building VOs based on the concept of the Grid vary from place to place depending on the various demands like small or large scale, short-live or long-live, single institution or multi-institution, homogeneous or heterogeneous system. Over all, many ensembles still exist, constructing the so-called Open Grid Service Architecture (OGSA). In deed, developers of applications for VOs encounter with common issues. For instance, every Grid needs to be equipped with services (a service is a network-enabled entity that provides some capability), there are certainly many services used in common. The Grid (or VOs structure) needs mechanisms to make entities interoperable. Mechanisms used here are services and the Grid using those mechanisms is therefore called service orientation environment. Services in the Grid have to satisfy two criteria, namely definition of service interfaces and the identification of the protocols. One more feature of the architecture is the virtualization; make it essential to be transparent, adaptable with local resources, across multiple platforms. Web Service Definition Language (WSDL) has been chosen to characterize for service interface, which is platform independence and thus virtualization is easier to implement on local resources. According to OGSA “A Grid service: a Web service that conforms to a set of convention and supports standard interfaces for such purposes as lifetime management” [8]. In this manner, Grid service is considered as a more semantically richer Web service.

Depending on the specific requirement, a Web service can be designed to be either stateful or stateless. In contrast, a Grid service is always stateful. A part from that, a Grid service has much more advanced features that can not find in a Web service such as lifetime management, notification, creating transient service (factory). Here, it is not intended to describe or compare the behaviors of Grid service and Web service, however,

we stress again that the choice of Web service and Grid service as interface technologies for some resource and the design of them can lead to the differences of using that resource. Obviously, if Web service is chosen to make the interface stateful, the design of that Web service has to contain some external mechanism that support state feature. With the latest Grid service interface technologies, one could benefit from using directly the state support mechanisms (such as lifetime management etc...) embedded in the interface. This point will be illustrated in our storage service design in chapter 4.

2.3.3 OGSi and WSRF

“Building on both Grid and Web services technologies, the Open Grid Services Infrastructure (OGSI) defines mechanisms for creating, managing, and exchanging information among entities called Grid services.” [9] The design of storage service heavily relies on Globus Toolkit, which is considered as the de facto implementation of OGSi. Hence, we start with a short discussion about Globus Toolkit. Besides, Java CogKit and Reliable File Transfer Service are the main components in storage architecture we therefore describe about their architectures and functionalities.

“In January 2004, the WS-Resource Framework was proposed as a refactoring and evolution of OGSi aimed at exploiting new Web services standards, specifically WS-Addressing, and at evolving OGSi based on early implementation and application experiences”[30] *“The WS-Resource construct has been proposed as a means of expressing the relationship between stateful resources and Web services.”* [31]

However, WSRF-based GT4.0 will be released in GT4 in near future. Here, only GT3.0 (the implementation of OGSi) is used for our experiment. Even in the latest release GT3.2, some WSRF are added but most of them are only minor changes.

2.3.4 Globus Toolkit

At the time of working on the thesis, Globus Toolkit version 3.0 is the newest release. At the first sight, Globus Toolkit 3.0 (GT3) strictly follows OGSi and uses Web Services protocols as an indispensable part of its architecture. As drawn in figure 2.3, GT3 is used Java as hosting environment and Web server to publish its services. Behind Web server is the Grid service container that contains a set of types of services with different levels such as system services, base services and user-defined services. In Grid service container, security and OSGi are two important components embedded, making GT3 fundamentally different from typical Web services architectures.

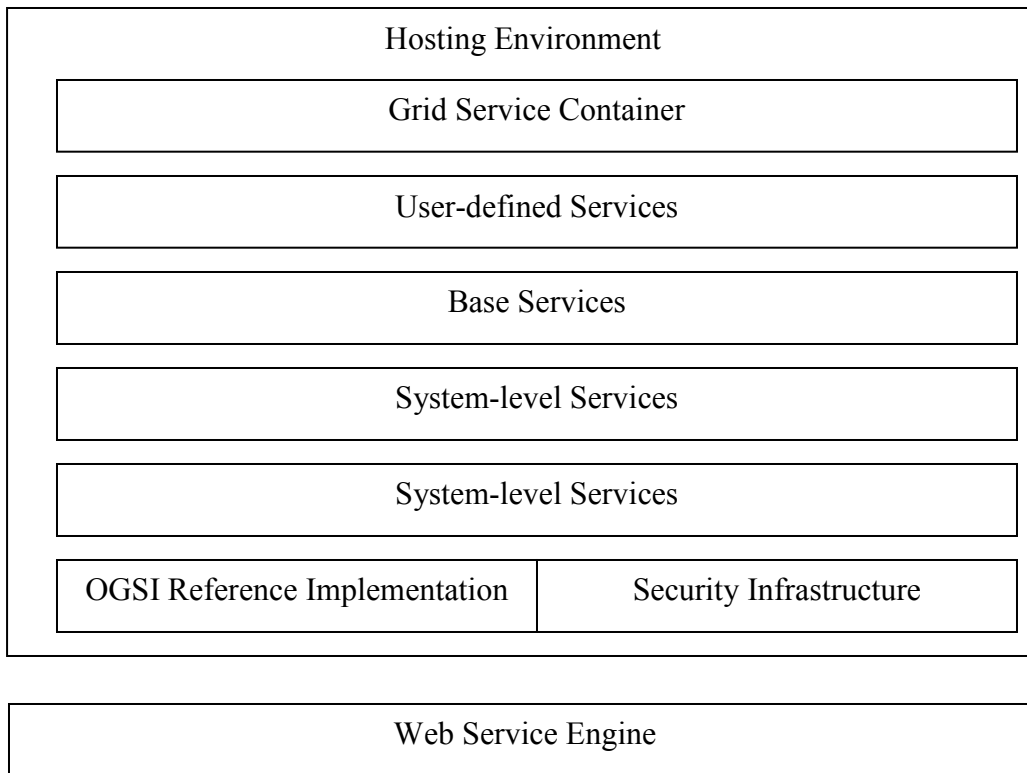


Figure 2.3: Globus Toolkit 3.0 Layered Architecture

2.3.5 Commodity Tool Kits

Programmers often find it difficult to work directly with GT3, thus CogKit as a programming library is developed to facilitate the use of GT3. CogKit supports for a variety of languages such as Java, Perl, Python, and C/C++. On the one hand, CogKit is capable of directly accessing to existing services and being capable of developing new Grid services on the other hand. In short, CogKit is a bridge between Grid developer and Grid system, providing high-level tools to develop Grid services instead of working directly with SOAP/XML. Amongst a number of languages supported by CogKit, Java is preferable due to its obviously advanced features like platform independent, OOP etc...as well as easy to use GT3 library (most of GT3 component built on Java). This is also the reason for us to choose Java in developing storage service.

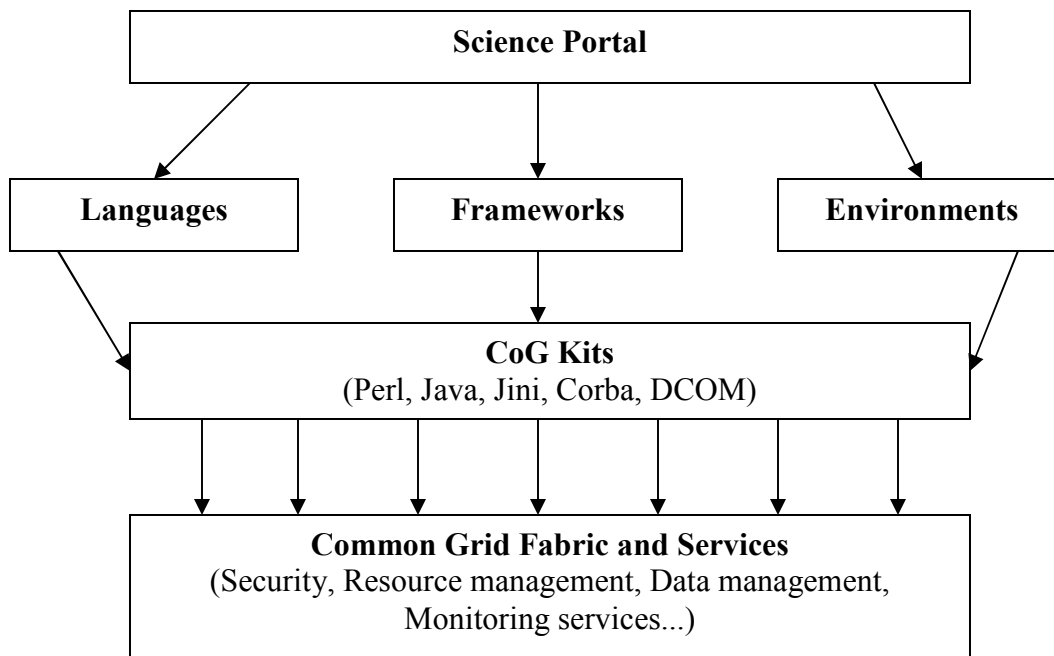


Figure 2.4: Commodity Tool Kit Architecture

2.3.6 GridFTP and Reliable File Transfer Service

Initially, we aimed to use GridFTP to serve as a data transport component in storage service design. However, we have decided to use Reliable File Transfer (RFT) due to its advantages in comparison with GridFTP. The GridFTP is a tool developed to meet the demand for a fast, secure, efficient, and reliable transport mechanism in the Grid. To some extent, it is an extension of FTP in TCP/IP protocols. It complements some advanced feature that makes GridFTP adapt to the Grid such as Grid Security Infrastructure and Kerberos support, third-party control of data transfer, parallel data transfer, striped data transfer and partial file transfer etc... Reliable Transfer Service (RFT) is an OGSA based service. RFT is capable of providing Grid users with controlling and monitoring third party file transfers using GridFTP servers. In fact, GridFTP performs the actual file transfer. RFT uses PostgreSQL to store the state of the transfer to be able to restart after failures. Obviously, programmers would benefit from using RFT instead of GridFTP because they do not have to encounter with hundred lines code, avoiding unexpected errors. Moreover, in combination with Java CogKit to access RFT also facilitates the use of GridFTP because programmers just have to work with Java class instead of working with SOAP/XML directly. Figure 2.5 illustrates how two GridFTP at different sites are instantiated by RFT. This mechanism (third-party transferring) is now new, as can be seen in FTP protocol, however, RFT has a Grid service interface is a new and advanced feature and of course one should use it as standard transport mean in building Grid system by GT3.

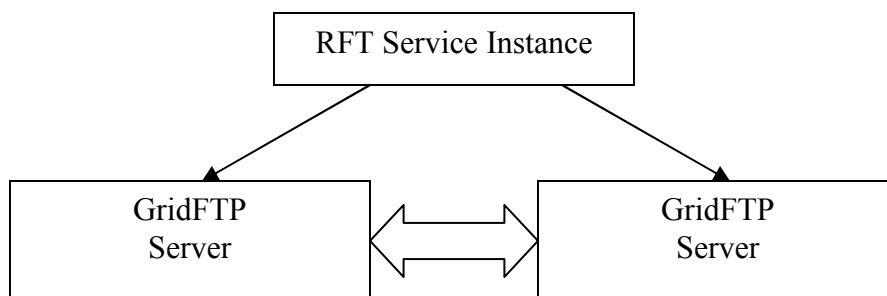


Figure 2.5: Third-party Data Transferring by RFT

2.4. The Potential of Web and Grid Services

2.4.1. What Can One Do If One Has These Technologies?

Web service and Grid technologies, if correctly applied to software that inter-works with (physical) resources, will bring then the ability to integrate these resources into applications. Application controls the inter-working of the resources. Furthermore, if possible for resources, the software might allow multiple concurrent users to access resources. As such, virtual resources have to be available in the form of Web (or Grid) services, wrapping all resource functionalities, which virtual resources normally should have. Alternatively, Web (or Grid) services environments play as the bridge between resource providers (virtual resources) and resource consumers (applications). Choosing Web services or Grid technologies depends upon the context of the application. For instance, Grid technologies are obviously suitable for Grid-applications that are currently popular in scientific environments while Web services technologies are more popular in corporate software. Figure 2.6 suggests that Web service or Grid technologies are the good choices to create interface for network resources. This created interface brings resource online for the accessibility from Internet based applications.

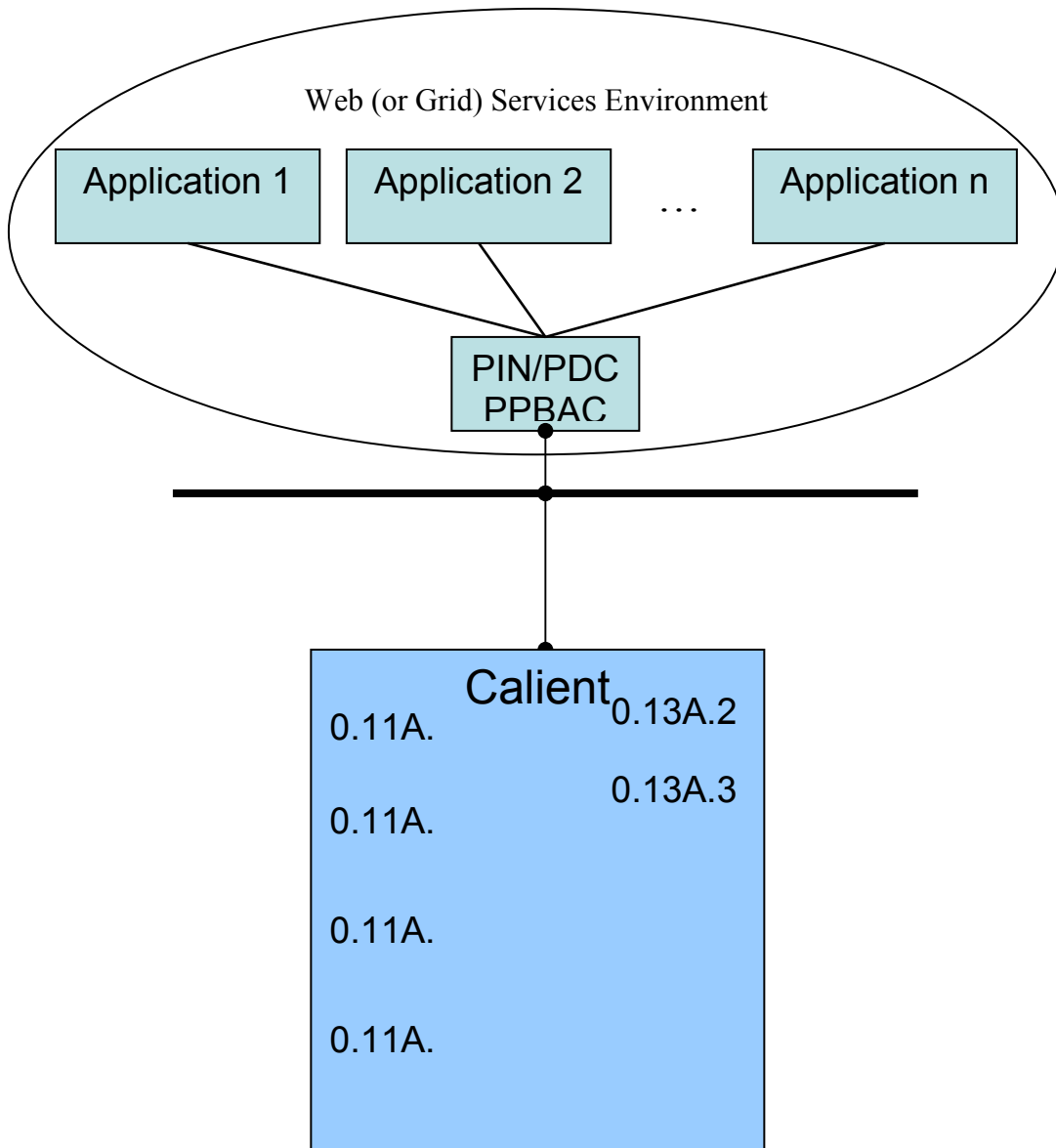


Figure 2.6: Web (or Grid) services technologies create interfaces to resources that are accessible via the Internet for other applications

2.4.2. Why Is It Very Intriguing If One Has Things Online?

Online virtual network resources can be exploited by one or more other software programs. In such control program, virtual resources are represented by software (proxy) objects, exposing relevant properties, methods, and events. This program in turn can be used by other software program as a single (integrated) service. The controls in the software program are trivial because those resources are considered as integral software objects of that program, which can be created and deleted easily. Furthermore, one

physical network resource can be used in some different software programs with similar or different functionalities if its control software would allow that. That means for each control program, virtual network resource itself can limit its functionality depending on the agreement between resource users and providers.

Figure 2.7 shows an example of how network elements are integrated and controlled by online software objects. It is assumed that two optical switches and storage have created a network path to a layer-three router. This router, also controlled by software, behaves specifically for this application and, for reasons known to the programmer of the application, multicasts all its IP traffic. Conceptually, each network device is represented by online software object. Of course, they function differently but they are placed on the sharing line (e.g. the Internet). In this manner, those objects can be regarded as online objects. Another application (called Super Mario here) in turn controls all online objects. Thereby, instead of invoking directly to network elements, Super Mario only needs to deal with online objects, which is obviously easier. Super Mario pulls the "ropes" that control the network elements: the network becomes a marionet.

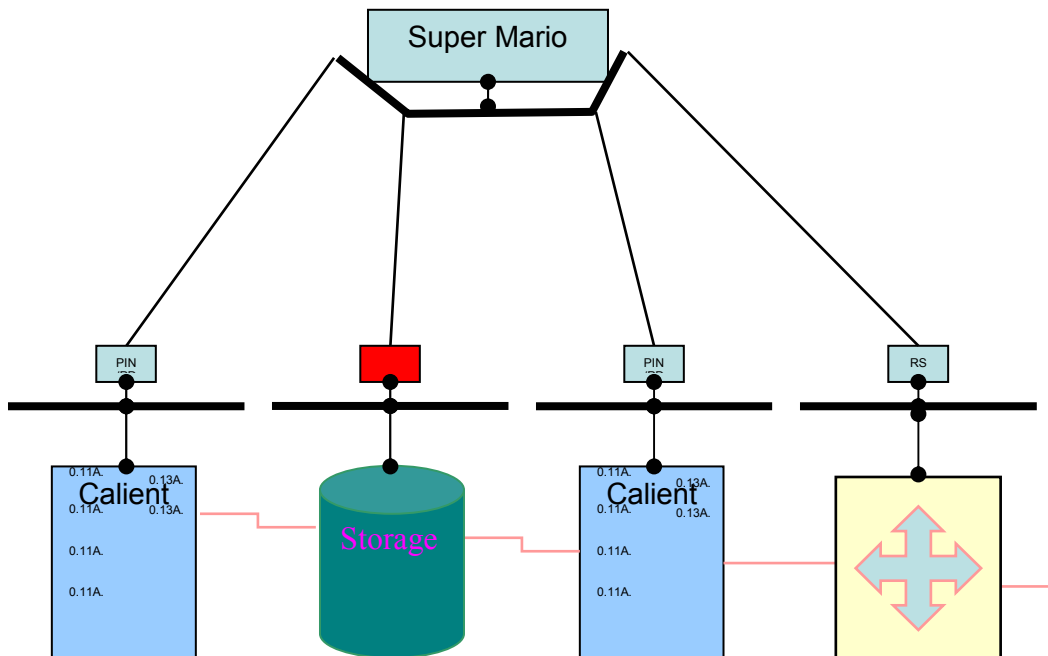


Figure 2.7: Applications integrate various network elements to provide a single application specific network service.

In figure 2.8, to be more specific, we suggest three methods to control software objects, as they are online.

- The application itself does all by possibly creating a very specific usage of the network (or a specific dynamic network topology).
- A broker integrates the object into a more general-purpose network service, providing a single access service point for users.

- Using a workflow paradigm a set of brokers the network services are used by a variety of applications customers.

Depending on certain goals, those methods can be chosen or even combined. In our implementation, the broker method is used (see more details in chapter 4).

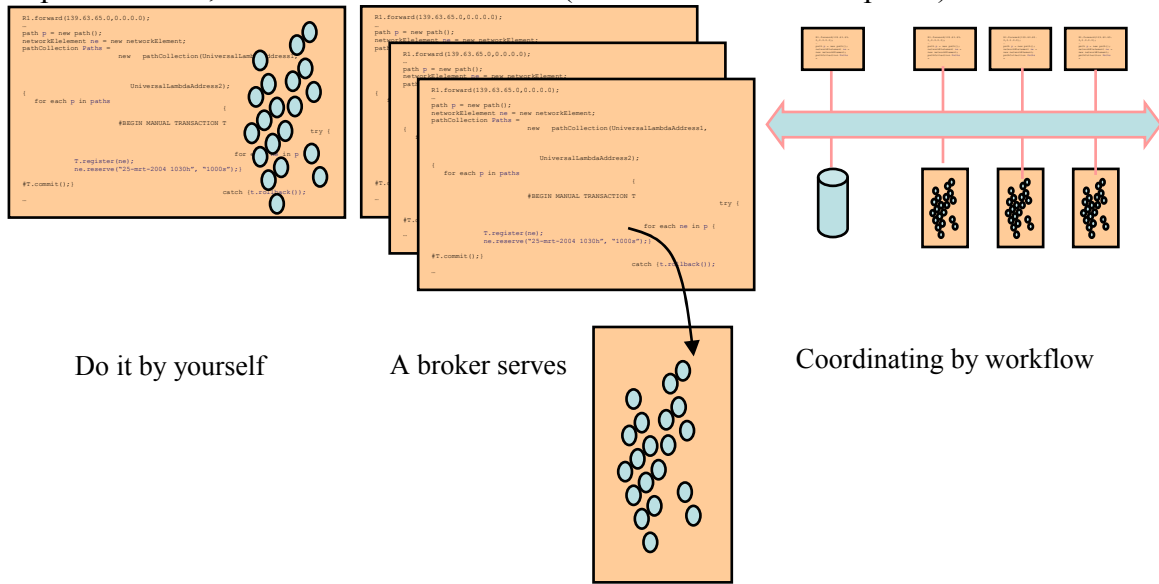


Figure 2.8: Three patterns to integrate network element objects with other software.

Chapter 3: Lambda Networking

3.1 Introduction

The ultimate task of any network is to set up and control the connections between network users (applications) as illustrated in figure 3.1. Consequently, a network has to provide for applications with some mechanism (or signaling protocol) to set up connections. In general, applications have two mechanisms with which they set up connections. One mechanism is to use a software entity - mostly called dialup adaptor - with which one can setup connection oriented paths: e.g. a dialup, Multi-protocol Label Switching (or Multi-protocol Lambda Switching-MPlambdaS) or Asynchronous Transfer Node (ATM) connection. Here, in fact, the dialup adaptor sets up the path with the aid of the User Network Interface (UNI). In the case of a dialup connection it consists of the telephone signaling system and a PPP server. To address an MPLS service, the dial-up adaptor interacts with the MPLS server. In the case of ATM the dialup adaptor interacts with an ATM-UNI server that is accessed via the edge ATM switch of the operator. In these cases, the application cannot access or address network domain issues or network elements behind the edge network element of the access operator.

Another common facility to setup layer 3 connections is the socket interface. Most frequently socket protocol used for the TCP/IP family of protocols is operated on a connectionless routing network. In most implementations, the socket interface is capable of handling other protocol families and connection oriented paths. Nevertheless the socket protocol is implicitly designed to address an abstract UNI and thus not individual networks or network elements.

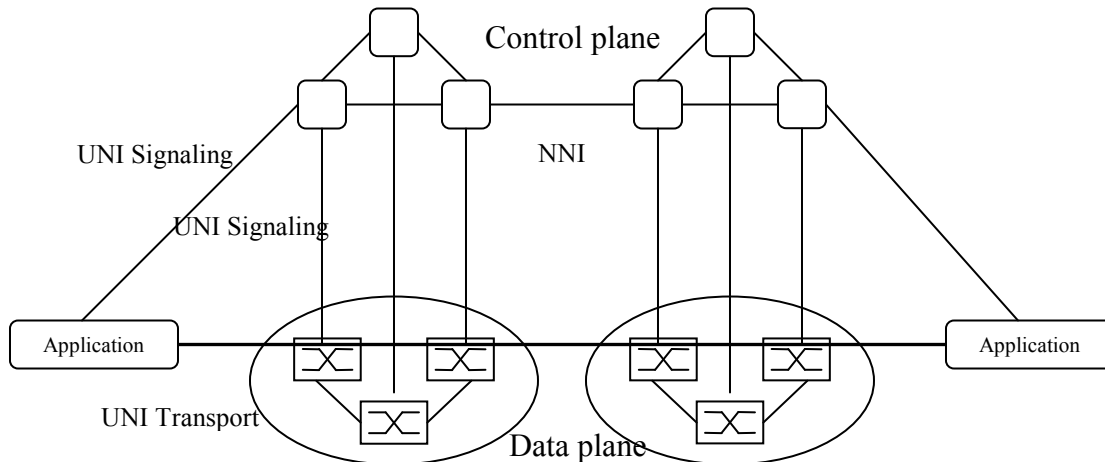


Figure 3.1: An Example of Signaling Over Multiple Domains

In conclusion, the most common path setup mechanism has to interact with a single edge device. This edge device obscures most details of network elements and domains through which the path goes. Sometimes this is not a problem but a benefit. In any case, in the case of optical networking a commonly used signaling system to setup a path (or sub-path) is lacking. Indeed instead of developing a system that probably ends up as complex as SS7 for the telephony infrastructure, the optical networking community has, probably due to a lack of human resources, embarked on another path and then it turned to a service oriented approach. Web service and Grid service technologies have been opted as UNI signaling of lambda-switching network.

3.2. Lambda Networking

As indicated in figure 3.2, Photonic Cross Connected (PXC) is the major element in a lambda-switching network. Its main purpose is to direct IP package to other switch following correctly the path that is set as package being created.

Dense Wavelength Division Multiplexing (DWDM) is a technique that allows multiple signals of different wavelengths to tump onto a single fiber. Such single fiber contains a number of virtual fibers that have different wavelengths (often called lambdas). This technology is preferably used in PXC but not limited to other technologies such as GigabitEthernet, SDH and SONET.

The figure 3.5 shows a typical architecture of PXC in compliance with an IP router.

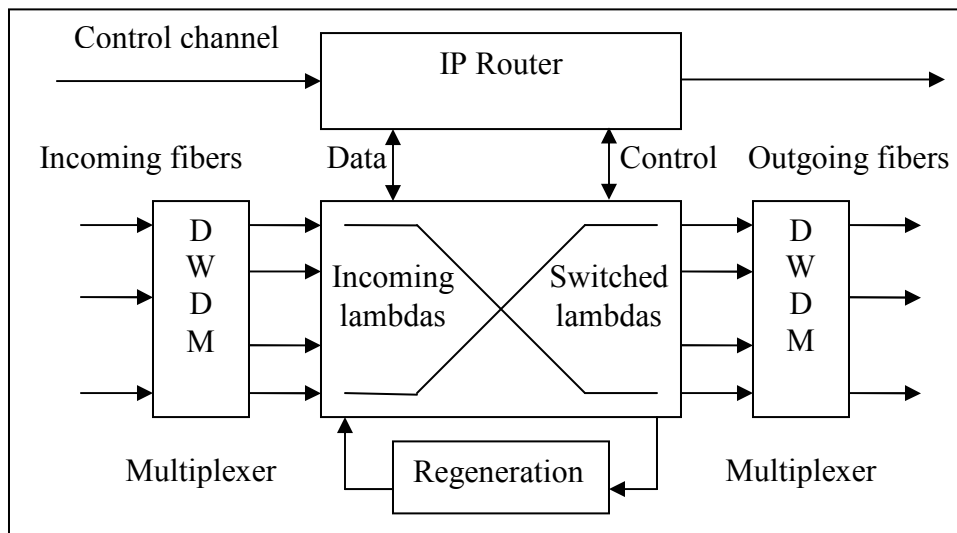
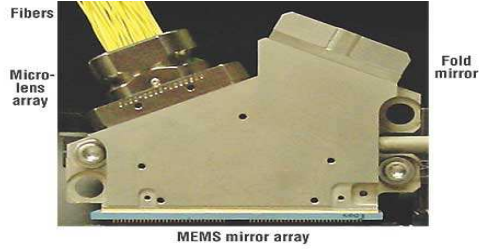


Figure 3.2: a typical PXC Architecture

The current lambda-switching network aims to provide for applications with optical paths connecting applications on each pair of nodes. To assure the bandwidth and QoS, a fair of fibers is dedicated to connect each pair of nodes. Thereby, the switching job at PXC is rather simple. PXC only needs to switch physical fibers coming one PXC to one or more outgoing fibers by using a brilliant mirror array as shown in figure 3.3.



1. This Reflexion 80-switchpack 3D photonic switch from Glimmerglass, which can fit in the palm of a hand, forms the core of a 80- by 80-port all-optical switch.

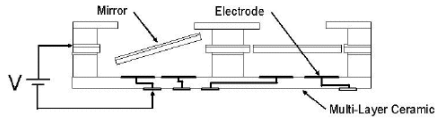


Fig-1: MEMS module with electrostatic actuation (side view)

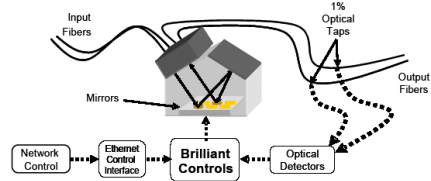


Fig-2: Brilliant Controls closed-loop feedback system

Figure 3.3: A λ -switch switches light from one fiber to one or more other fibers

Typically, PXC's of a lambda-switching network are distributed across multiple domains as drawn in figure 3.3. It raises a lot of issues that signaling protocols of this network have to deal with. For example, domains are authorized by different authorizations. Therefore, in order to set up the optical path crossing over domains, application (or user) needs to be authorized by all of those domains. Currently, PIN (see more details in 3.4) and AAA (see more details in 3.3) solved completely such issues. In our experiment, we do not actually use PIN software and do use AAA service a lot. Hence, PIN is only short described while the AAA concept and the implementation are discussed in detail.

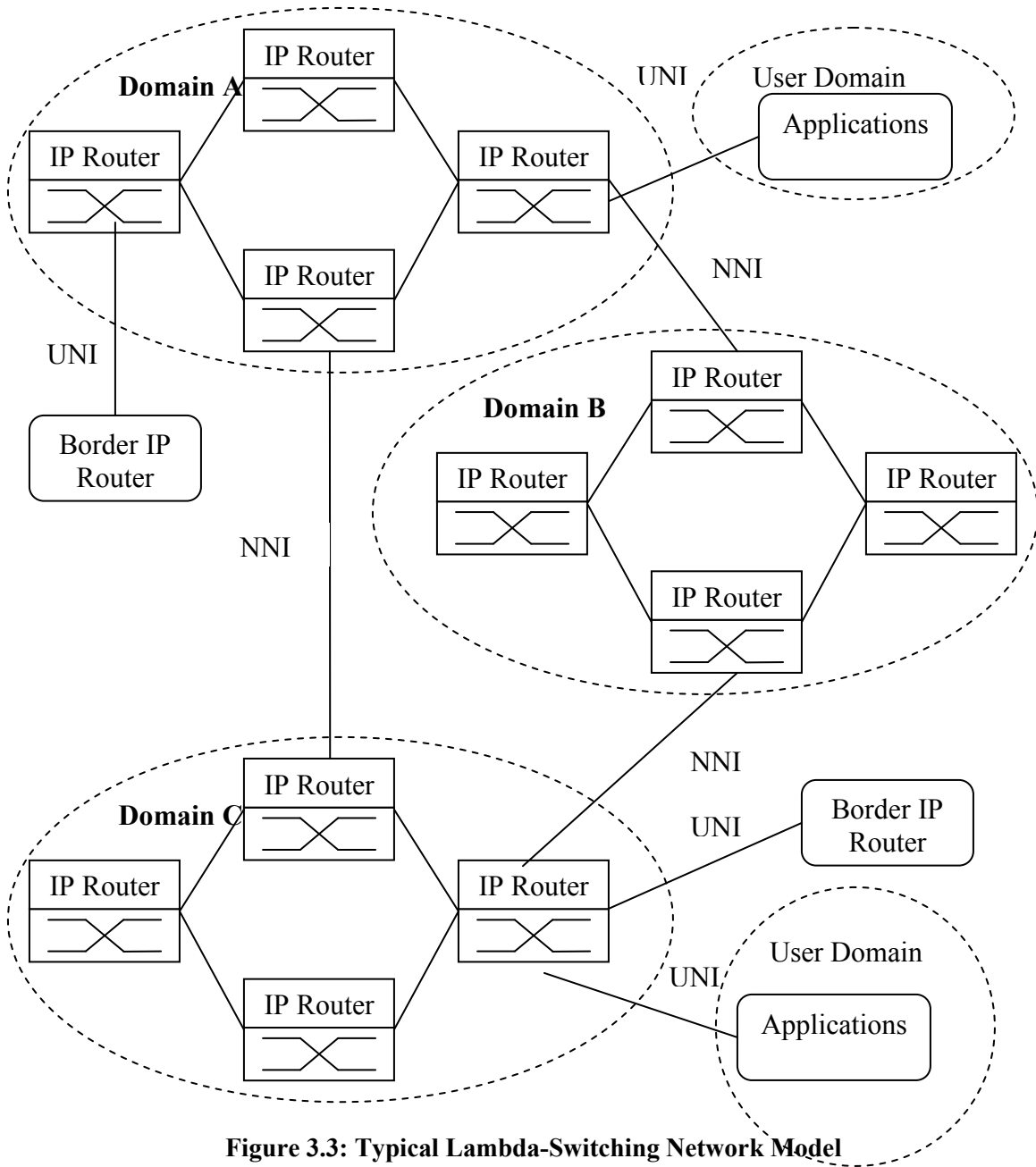


Figure 3.3: Typical Lambda-Switching Network Model

3.3. Authentication Authorization Accounting Concept and Implementation

The discussion here just covers some basis points of AAA that serves the specific purpose of the report. Figure 3.4 sketches a number of network resources distributed across multiple domains. How does one create a network path for a user in a situation

where access to and transfer through a domain is in matter of authentication, authorization and administration (in case somebody has to pay for the service usage)?

The AAA research group investigates this situation, in the past this led to a number of RFCs. Furthermore, this group has developed an AAA software program that performs the AAA job and does, in the case of lambda networks, the optical switching as well. A complete and independent AAA software program is placed at every domain. Each AAA software program itself consists of three independent components, being in charge of doing several main tasks as follows:

- AAA server is in charge of receiving and processing requests coming from clients (resource users). It provides network resources at its domain if required by invoking to two other components. Moreover, it negotiates with other AAAs to satisfy the request that asks network resources across domains. The communication between AAAs is a peer-to-peer protocol.
- ASM (application specific module) is a function library, being capable of directly manage network resources. Those functionalities are provided in the form of API or protocol based interface.
- PR (policy rules) stores set of policies that determine the behaviors of AAA server on requests.

Those components are independent in the sense that either of them can be changed without any effect on others.

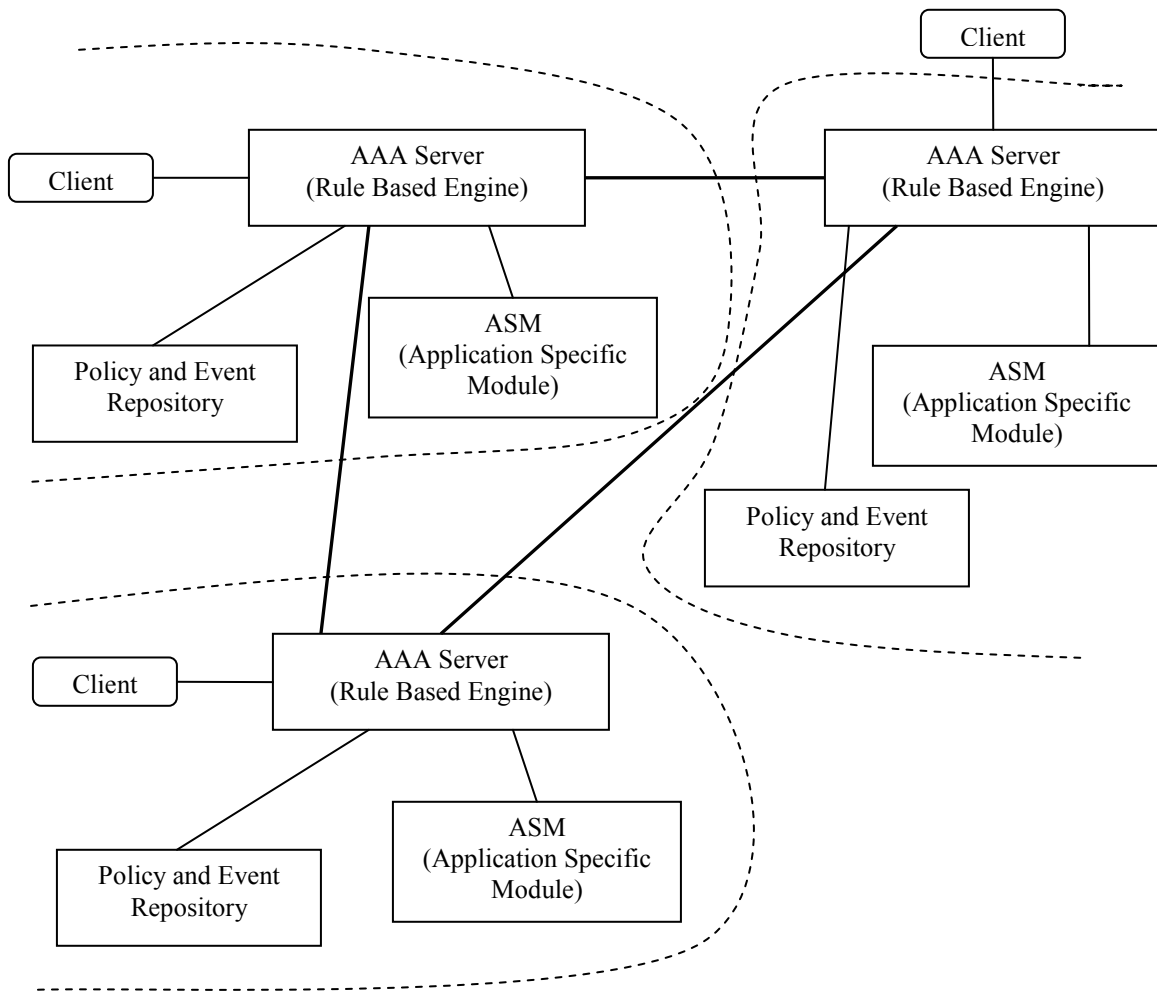


Figure 3.4: Generic AAA Architecture Cross Multiple Domains

In the lambda-switching network, the lambda switches are administered by many organizations. AAA technologies are suitable to manage those lambda-switching resources. An implementation of AAA designed specifically for lambda-switching network has been developing in AIR group [26]. It provides for network users with ability of opening some optical path over domains in the form of Web services (or Grid services)

3.4. The Current State of Lambda Networking

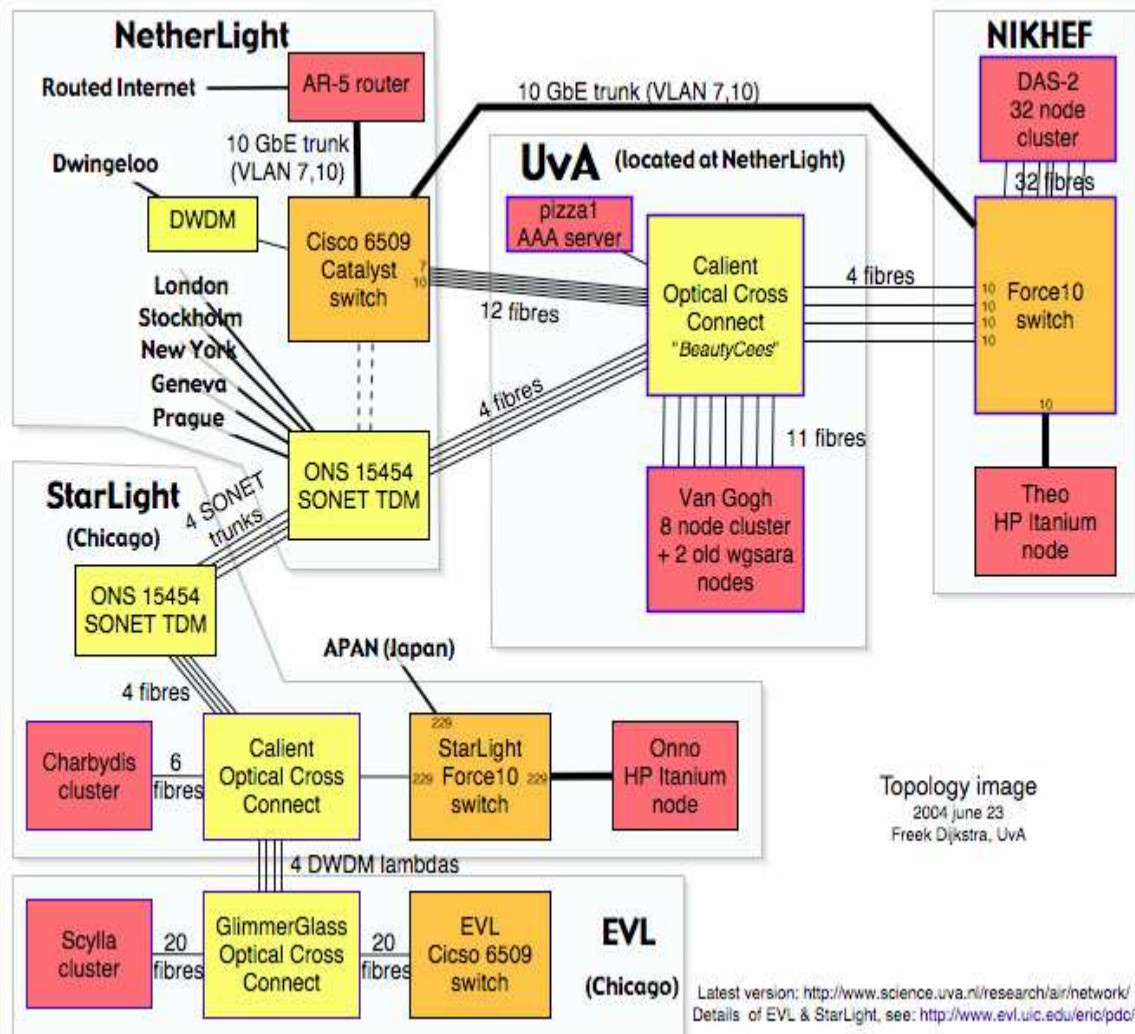
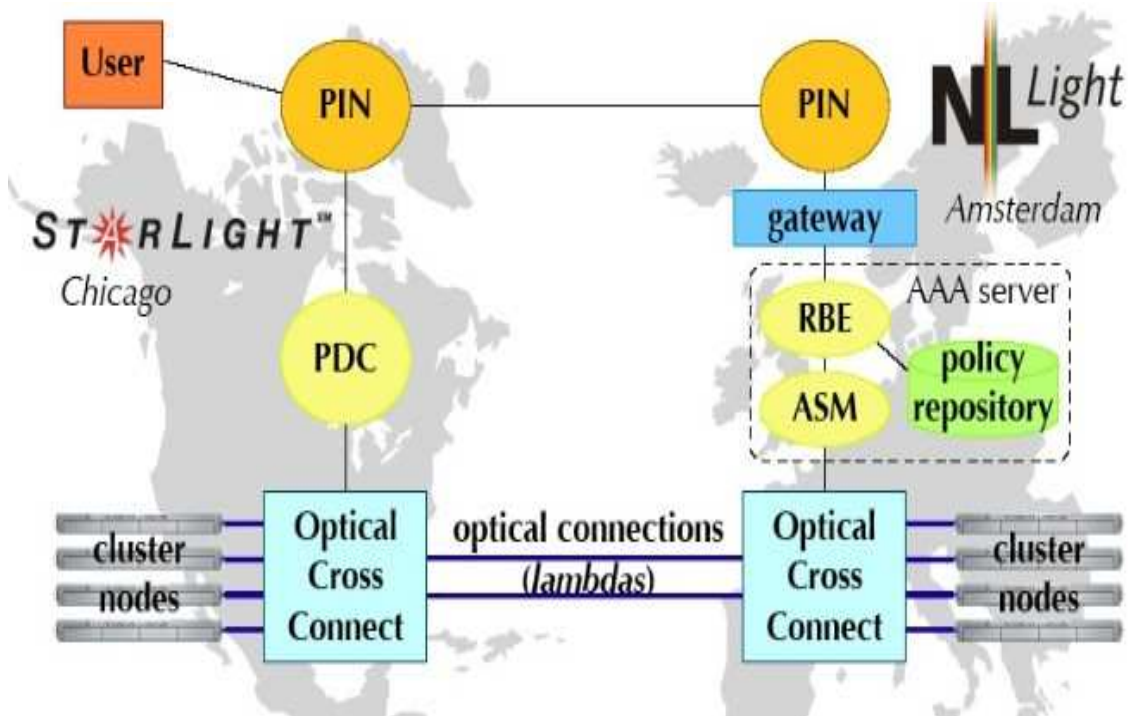


Figure 3.5: The Current Topology of Lambda-switching Network

At the first look, the network topology is quite complex, operating at three layers. As can be seen in figure 3.5, depending on each goal of communicating, many technologies at link-layer can be deployed such as DWDM or GigaEthernet.



PIN	Photonic Inter-domain Negotiator
PDC	Photonic Domain Controller
AAA	Authentication, Authorization and Accounting
RBE	Rule Based Engine
ASM	Application Specific Module

Figure 3.6: Signaling Technologies in Lambda-switching Network

Figure 3.6 is a part of the entire network focusing on lambda-switching part and the software for signaling. Being constructed across multiple domains is one of the most difficulties of lambda-switching network in provisioning lambdas (or lambda signaling).

Although Calient switch does support Multi-protocol Label Switching (MPLS) or Multi-protocol Lambda Switching (MPLambdaS), it needs some pre-configurations at two end-point applications (or two nodes). In addition, too much effort for arranging connection between two parties with the current specs can be considered as another disadvantage. Still, switches enabled MPLS from different vendors might find them incompatible to co-operate. Moreover, MPLS in fact does not provide any mechanism to solve domain issues yet. A lightweight protocol called Photonic Inter-domain Negotiator (PIN) has been developed to cope with multi-domain issues. The main goal of PIN is to provision lambdas through multi administrative domains to setup an end-to-end optical pipe for high bandwidth demanding applications. PIN is still being developed at EVL ([33]).

In collaboration with PIN, Photonic Data Controller (PDC) can be used to control switches in a single administrative domain and Photonic Policy Based Access Controller

(PPBAC) to provide lambdas. The Air group of the UvA has experience gained experience with this software gained in the Supercomputing 2003 event. A comparison with AA shows that all functionalities of PDC and PPBAC are integrated into AAA. AAA can, on contrary to PIN, solve inter-domain issues. AAA supports Web service interface and will support Grid environment in near future.

Despite the progress of PIN and AAA, for the time being, the lambda-switching network remains a work in progress.

Chapter 4: Storage Service and Network Service Model

4.1. Introduction

In this chapter we explore the concepts of network services. Until now the Web service approach to interface to the network infrastructure was merely seen as a technical easy way to control networking equipment and not as a new concept for networking. In this chapter, however, we explore what it means if a network is regarded as an entity that provides a service.

In the network service concept, the AAA service discloses the lambda-switching network to potential users on the Internet. As such, it is available for integration with other Web and Grid services. Clearly such integration must be meaningful, i.e. the service to be integrated to the network through software must be connected to the network or the network management system. Therefore, as a manner to study the network service concept we have chosen to study the integration storage facilities to the network. This integration can be done in such a way that in turn it creates a new network service (called store and forward network service). As we will show, with a proper constructed network and storage service, this integration is just a matter of a few lines of code. Although the creation of a “proper” online service that represent a network or storage is not an easy task. Furthermore, the actual performance of the storage service is of minor interest for this study.

The objective of exploring the network service concept is to explore its new possibilities. One of them is a new way of creating personalized networked applications and we show that by creating an application that combines a network service with a storage service. This is also a specific solution to the congestion problem of lambda-switching network as described previously. The concept of network service can be seen elsewhere too (e.g. Jini [27] or UPnP [28]). Yet, in our approach, Grid services and Web services technologies are used to build up network service model. To this end, we focus on describing the storage service, the new network service model.

4.2. Virtual Network Resources in Internetworking and Telecommunication Field

The fact that one uses Web or Grid service technologies to interface to the networking equipment opens a new way of looking at a network. For software engineers, Web services are manifested as (software) objects in their code. Consequently, a network is represented as a collection of objects in a computer program¹. The physical network becomes a virtual network resource to the programmer: an entity, which somewhere

¹ Note that by the Web service and Grid service concepts one automatically chooses objects to represent for physical resources. Other representations are also thinkable, e.g. one can project a switching action in a network onto an instruction in a virtual machine.

exists, performs some functions but exists for him only as an object in his code. Since dealing with a network means then dealing with a set of objects, one easily imagines that one can aggregate those object into an agent, a broker that relieves the programmer from the burden to deal with individual networking elements and other resources. An example for a Broker program that controls networking related resources through software objects is shown in figure 4.1:

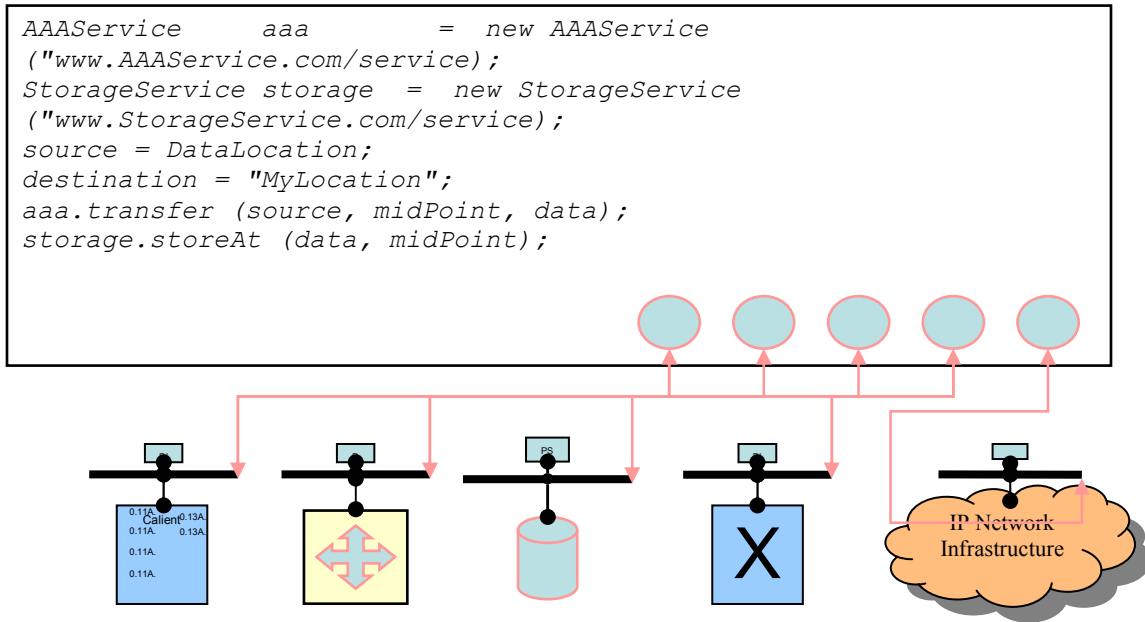


Figure 4.1: Resources Controlled by Software Objects in Java Program

4.3. Storage Service Model

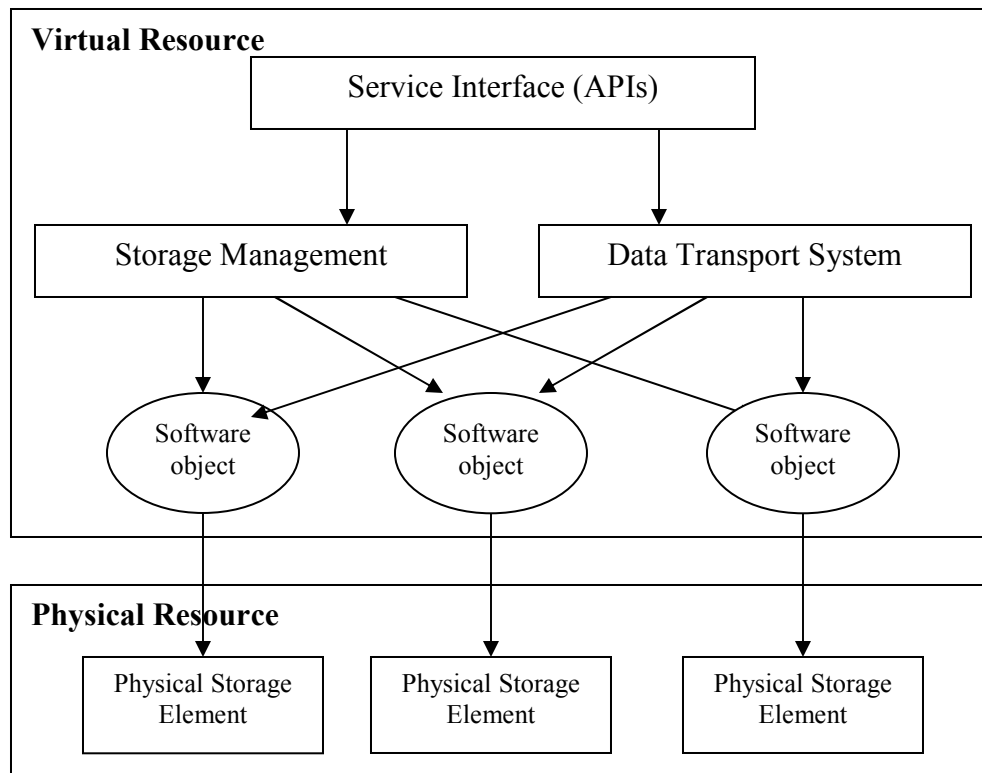


Figure 4.2: Logical Storage Service Architecture

As a means to study the network service concept we decided to integrate it with another service: the storage service. This was somewhat an arbitrary choice motivated only by the facts that it was a straightforward and useful service that was easy to implement with a personal computer.

In the lambda-switching network, there is no intermediate storage of data. The network elements operate at (OSI) layer 1: thereby establishing an optical path. Consequently, there is no sense to transmit data if a part of the dedicated path is unavailable. For some applications, it might be desirable to transmit data to a storage, which is physically closer to the destination than the source. The storage on its turn should then transmit the data to its destination at another time when the remainder of the path is available. In our case, the storage is a computer with an optical interface, a proper protocol stack, and a storage service. Here, it is suggested that the service interface component speak an as much independent language as possible (e.g. Web service)

As mentioned, for specific situations, application programmers want to store data along the transmission path if the remainder of the path will be unavailable for some time. This storing action can improve the performance of the network.

Figure 4.2 represents the model of the Storage Service that has been designed and implemented. According to virtual network resources concept, the architecture should be separated into two parts: virtual network resource and physical network resource. The virtual part of Storage Service contains three basis components. The service interface ought to handle all kinds of incoming requests such as asking for storage, transferring or retrieving data or checking status of stored data etc... This component is the unique bridge connecting users and the storage system.

Another component called Storage Management is in charge of managing and controlling physical storage elements locally. For instance, upon request from Service Interface, it can call primitives to reserve space on disk or create and delete physical storage elements. Nevertheless, from the usage point of view, users should be shielded as much as possible from details of operation of the service and that the use of the interface itself involves as simple as possible software constructs.

Since we are studying the networking service concept, we want to integrate it with a storage service. The storage service itself is created by adding a Web or Grid service interface to the storage facilities. Consequently, each storage facility will be represented by a software object in a computer program that integrates the network and the storage services.

Since data has to move to the storage facilities, one needs an additional service to transport data. In our model a Web service called Data Transport System is in charge of moving data from user location to storage and from store to user destination or another storage system. This component in fact on the one hand, accesses to physical storage (e.g. disk) through the software objects and initiate some transport engine (e.g. File Transfer Protocol software) to do the real data movement.

The physical storage resource can be an external disk or some other storage technology as long as an API is available from which we can control that resource and attach a Web service to.

4.4. Store and Forward Network Service Model

Figure 4.3 shows the architecture of network service, which is created by the inspiration of the Grid, and TCP/IP architecture: hourglass model. The neck represents a limited set of protocols, APIs, or other interfaces that have been made available to the service consumers. Similar to the concept of reduced instruction set computers (RISC), one can construct functions that are more complex from the ones presented in the neck. This is advantageous also for the network resource providers – they do not have to support a rich set of interface functions. Therefore, instead of letting application programmers to deal with network and storage facilities, we have designed a broker that integrates these to a storage network service. Clearly the benefit is, just with any other “bottleneck architectures” an ease of use which is obtained at the cost of allowing programmers to create more optimal, more specific solutions.

A broker can be thought of as a bridge to cross from users to multiple network resources. In principle, one wants to use network resources then one has to use it through the broker of those network resources. Broker is uniquely responsible for the use of network resources. Broker has its own interface allowing users to access and its own APIs to control network resources.

In our model, the neck of this hourglass is formed by the Broker service that allows a variety of applications to access a variety of network resources. In fact, in our experiment, SOAP/XML is used because of many its advantages. For instance, user just knows about the XML schema of the broker service, one can access to that service regardless of whatever technology used at user side as long as this technology supports SOAP/XML.

4.4.1. Services Provided by the Broker

From the Broker point of view, all network elements are viewed as online software objects. For all invocations to network element services, (all) the Broker has to activate software objects inside its code. Web (or Grid) service technologies connect these software objects to network elements. Thus, all sorts of rather elementary services like the storage service and the AAA service are network resources that are used by the Broker through software objects. By introducing a broker, the user does not need to access directly these resources. Instead, user has to access through a single service point, the Broker, of the network. As such, the Broker should be of help to the user to perform a sequence of tasks on the primitive services. Moreover, the Broker would ease the software engineering efforts of its users if it were able to combine and abstract the service primitives of its network resources to the essence required by its applications. Therefore, in the design of the interface one has to make choices like “this broker should support that kind of applications” and “the most common access mechanism to my broker services will be ...”

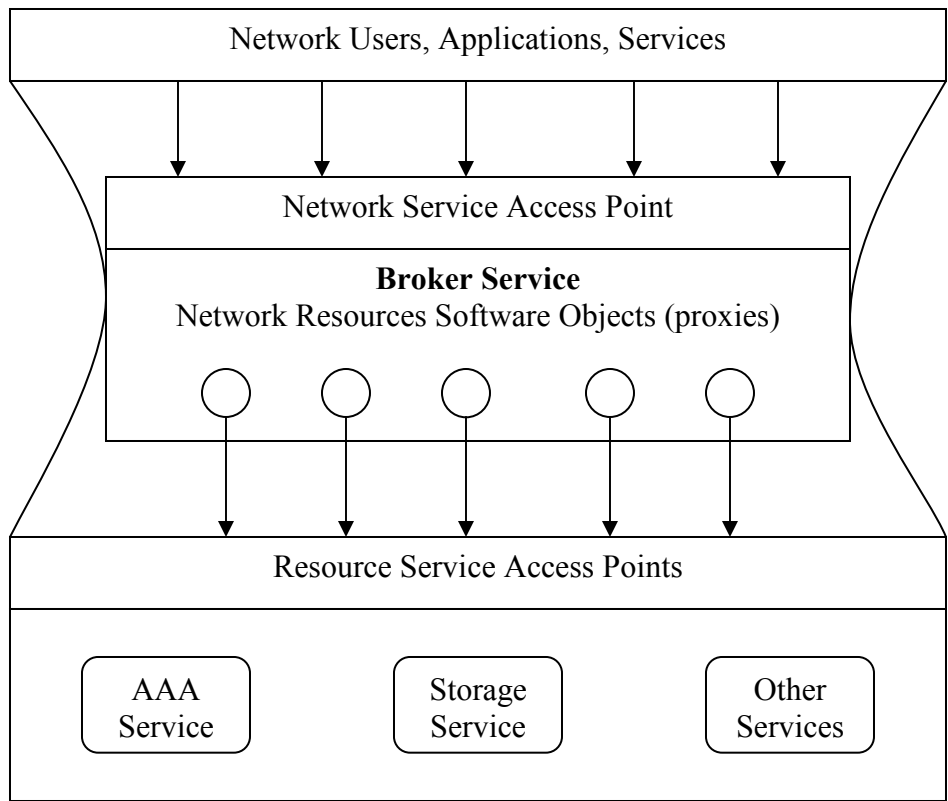


Figure 4.3: Store and Forward Network Service Model, Glass Hour Model Architecture

4.4.2 Central Controlling Broker Model

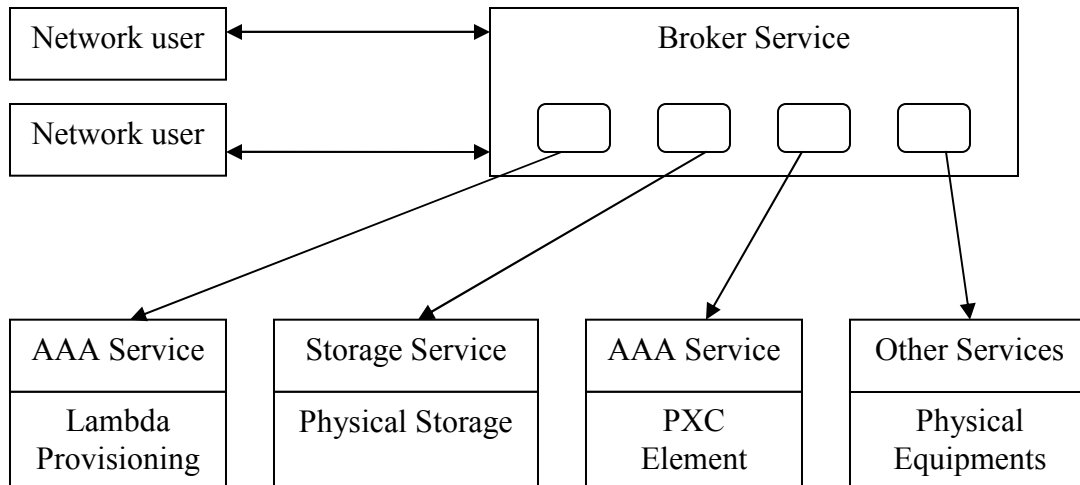


Figure 4.4: Centralized Controlling Model

As depicted in figure 4.4, in the central broker model the entire lambda network, the storage service and other services are viewed as software objects with respect to management and control. In this model, the Broker controls all network equipments. The control here is a matter of straightforward software engineering since the AAA interface and the Storage Interface are designed to make the exploitation of the network element as easy as possible. This is achieved by abstracting away details of the physical (OSI) layer and the switching protocols etc...

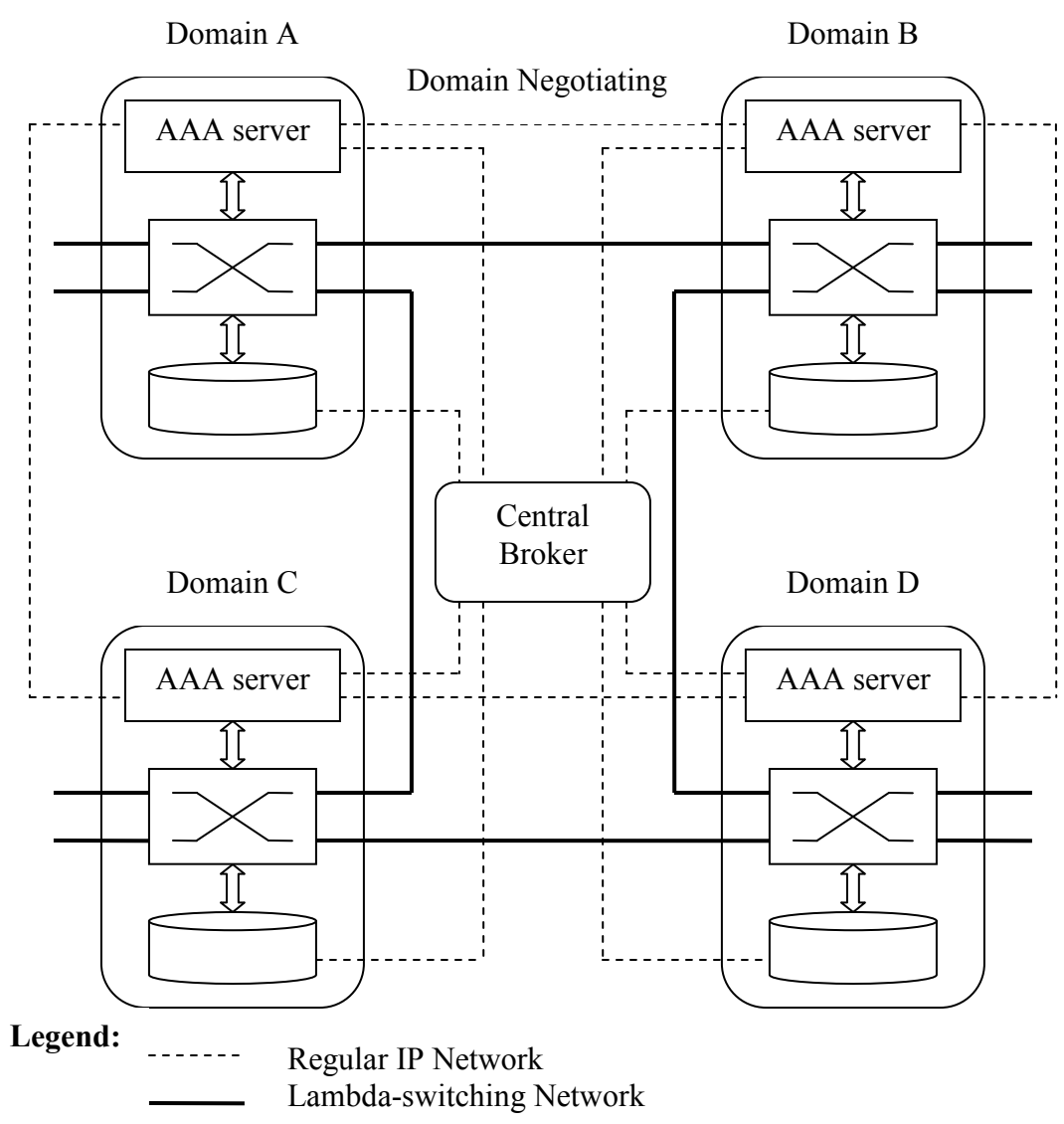


Figure 4.5: Centralized Controlling Model across Domains

Figure 4.5 describes a way of controlling network elements by the Broker. In this way, the Broker controls all local storages and all AAA servers located at all domains. On the one hand, the Broker is responsible for receiving data-transferring requests from users and on the other hand controlling AAA servers and storages through the corresponding software objects. The Broker is supposed to know all knowledge about AAA and storage as well as network topology. This model appears to be suitable for the lambda-switching network with a small number of switches.

4.4.3. Using Storage Service

This part describes the possible usages of storage service from the aspect of users as well as of the Broker. Algorithm 1 shows how the Broker works from the user point of view. The operations on user side are rather simple. For example, user asks the network service to transfer data from one location to another location.

Algorithm 1: Network Service Usage

```
1  NetworkService service = new NetworkService (“www.networkservice.com/service”);
2  try {
3      broker.transfer(“DataLocation”, “MyLocation”, data);
4  } catch (Exception e) {
5      message(“Fail to transfer data”);
6  }
```

Next, how the Broker manages its network resources to serve the request coming from user is illustrated in Algorithm 2.

Algorithm 2: Simple Broker Operation

```
1  AAAService aaa = new AAAService (“www.AAAService.com/service”);
2  StorageService storage = new StorageService (“www.StorageService.com/service”);
3  source = DataLocation;
4  destination = “MyLocation”;
5  try {
6      aaa.transfer (source, midPoint, data);
7      storage.storeAt(data, midPoint);
8      aaa.transfer(midPoint, destination, data);
9  } catch (Exception e) {
10     message(“Fail to transfer data”);
11 }
```

Moreover, one can imagine of a more intriguing and robust way of controlling the Broker’s network resources by using transaction technology. Algorithm 3 is an example of the use transaction technology in a networking environment. At the time of writing this thesis, we have never seen such a path setup reported in literature. The transferring data job is done if and only if the Broker succeeds in reserving enough network resources for it. The network resources are in this case a lambda-switch and storage.

Algorithm 3: Transactional Broker Operation

```
1 Path path = new Path( source, dest );  
2 NetworkElementList list = path.getConnection();  
3 NetworkElement ne = new NetworkElement;  
4 TransactionManager tm = new TransactionManager;  
5 tm.begin():  
6 try {  
7     for each ne in list {  
8         tm.register(ne);  
9         ne.reserve();  
10    }  
11    tm.commit();  
12    transfer(data, source, dest);  
13 } catch ( tm.rollback() ) {  
14    message("Fail to transfer data");  
15 } finally {  
16    tm.end()  
17 }
```

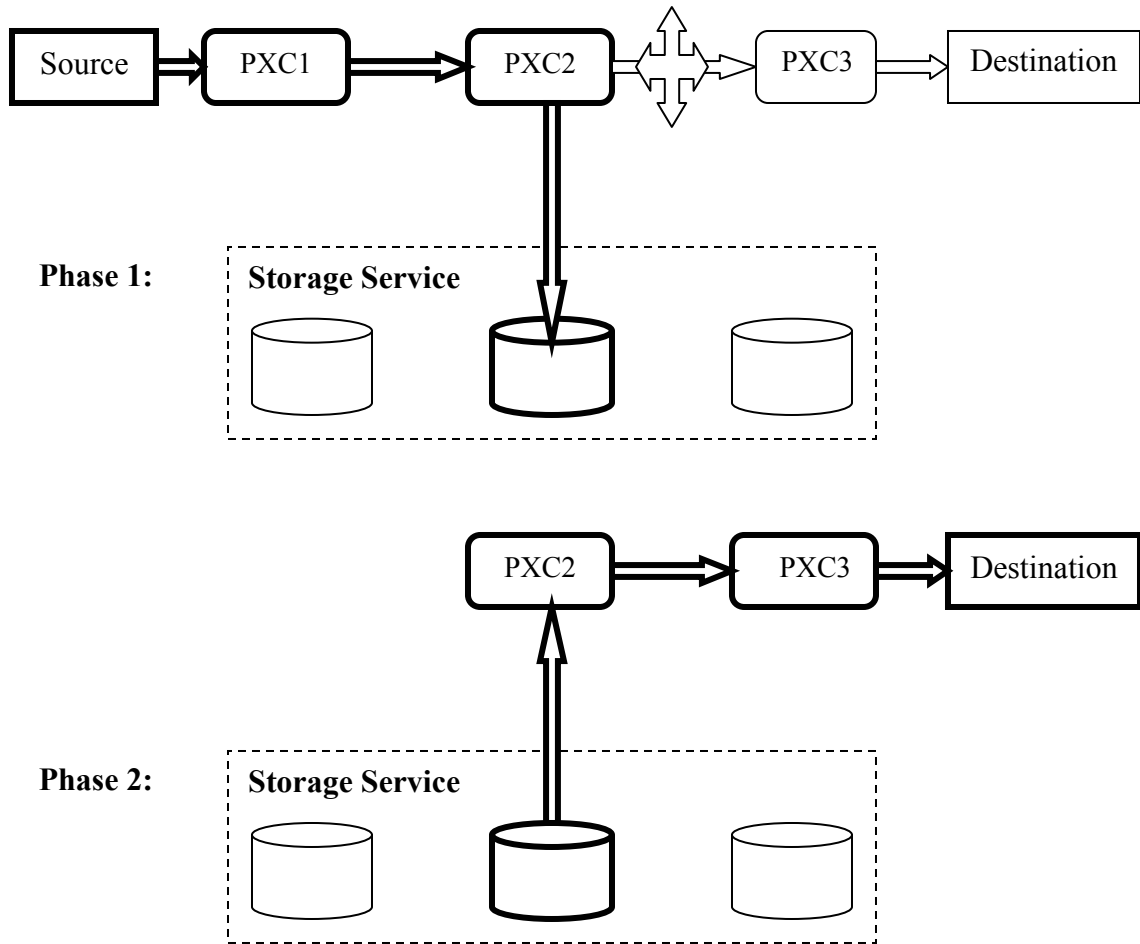


Figure 4.6: Storage Usage

Figure 4.6 illustrates an example in which a storage and lambda services are controlled by a broker in order to cope with the problem of not obtaining an end-to-end connection between source and destination at the same time. It should be reminded that Photonic Cross Connect (PXC), i.e. the optical switch, is the basis unit forming lambda-switching network. Information about connections between PXCs is not exposed via the AAA interface. Indeed, how to present information to AAA users about network topology is still a question without definitive answer. For now, it is simply information that is collected and passed outside the services framework.

In the first phase, the Broker fails to reserve connection between PXC2 and PXC3. Hence, data is stored temporarily in the storage located in switch 2. In the second phase, when the PXC2 and PXC3 connection is available, data from the storage is transferred to the destination.

4.4.4 Other Controlling Broker Models

In fact, apart from the centralized model, there are some other ways of controlling the network. However, they are not used in our implementation and design, they are shortly introduced here.

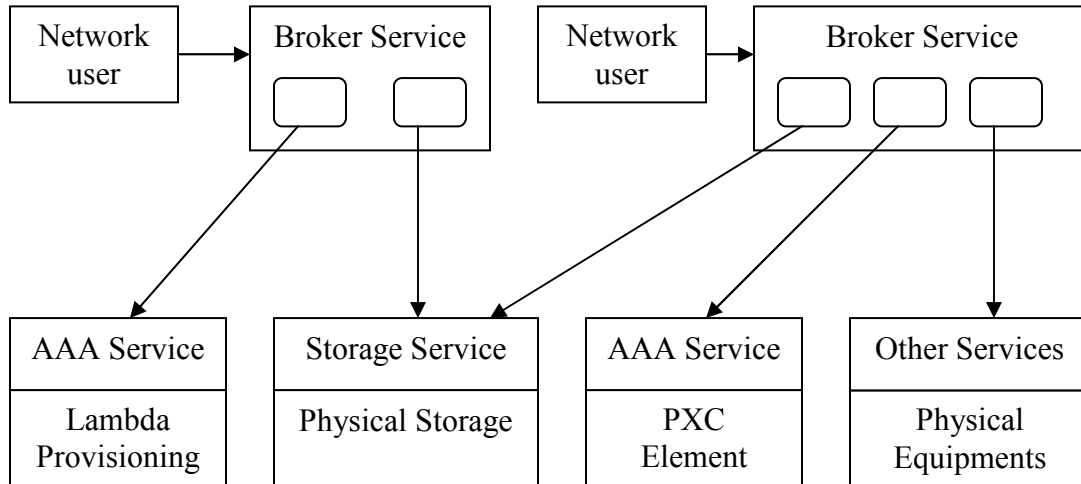


Figure 4.7: Multi-Broker Controlling Model

As indicated in figure 4.7, when network users have very different purposes of using those resources, there are several broker services are made to serve for those specific purposes. Figure 4.8 shows another possible model. That is a broker service is in charge of managing all resources via a workflow. All network users have to access via this workflow to make use of the resources. This model might be advantageous for avoiding congestion when several users access simultaneously to one resource.

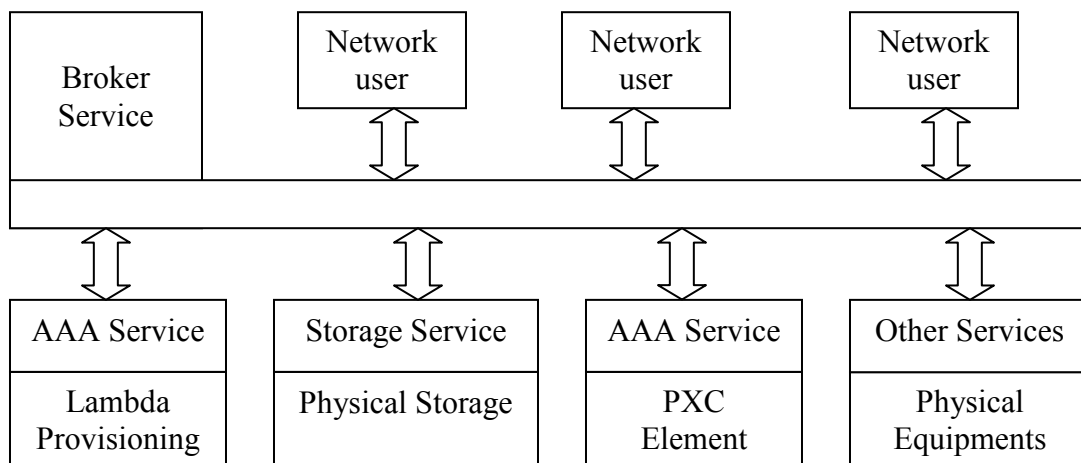


Figure 4.8: Workflow Controlling Model

Chapter 5: Network Service Design and Operation

5.1. Introduction

In this chapter, we describe the design and implementation of the store and forward network service model described previously. For this, also described previously, we use Web services and Grid services. For AAA [see more details in 26], we aim to describe its services rather than its detail design. This is a benefit from using the network service model: one needs only to know how to invoke the service, not how it is constructed. Another advantage is that we are less concerned how those details change as long as the interface remains stable. As a matter of fact the construction of the AAA service is an ongoing effort of the AIR group. Therefore, more importantly, we focus on describing the technical designs and implementations of storage service and network model. All our experiments are carried out on the lambda-switching network in Universiteit van Amsterdam.

5.2. Storage Service Design

From the view of storage service user, a storage service is simply a service in which one can invoke by means of software. The service can store and retrieve data safely and reliably. Furthermore, managing the data stored in the storage is a necessity, but it should not surpass the abilities of very elementary file systems².

In this thesis report, we describe straightforward but functional sufficient storage service architecture. Despite the simplicity of our storage service, it fulfills the basic requirements concerning security, reliability, and ease of use that is mentioned above. For Grid applications, the Globus Toolkit is the de facto interface technology to the storage service. However, as we shall see, we found Web service technologies to be more mature and suitable for a production system. With respect to architecture, the storage service interfaced by Web or Grid service technologies remains the same. Although the storage service can benefit much from applying Grid service technology to the storage service interface, it does not fundamentally change the behavior of storage service to its user (The Broker).

Yet, in terms of technology, one has of course to change some of the code if one changes the interface technology. One of the more noticeable consequences, however, is that in the Grid environment one can use GridFTP as a data transport mean, whereas one has to fall back to a less sophisticated parallel FTP program (see more details in 1.4) in the Web services environment.

5.2.1. Storage Service Development

Initially, in line with the growing popularity of Grid technologies, we used Globus Toolkit and GridFTP to develop our storage service software. The Globus toolkit was used to create the interface to the storage facilities and GridFTP for the Data Transfer

² In some documents, it is called metadata management.

Component. The software was tested to on a normal IP network with a dummy AAA object. However, as it turned out, due to a missing IP number plan on the lambda-switching network, we were not able to obtain the proper security certificates for the Grid services. Hence, Grid just did not run! Consequently, we were forced to use the Java Web Service Developer Package (JWS DP) in The Java 2 Platform, Enterprise Edition (J2EE). A simple parallel file transfer program was developed to replace the GridFTP application. Indeed, for our purposes, this parallel file transfer program became also the tool of choice on networks where GridFTP was available. We describe in the following sections first the Grid service implementations and conclude by a description of those of the Web service.

5.2.2. Grid Service Approach

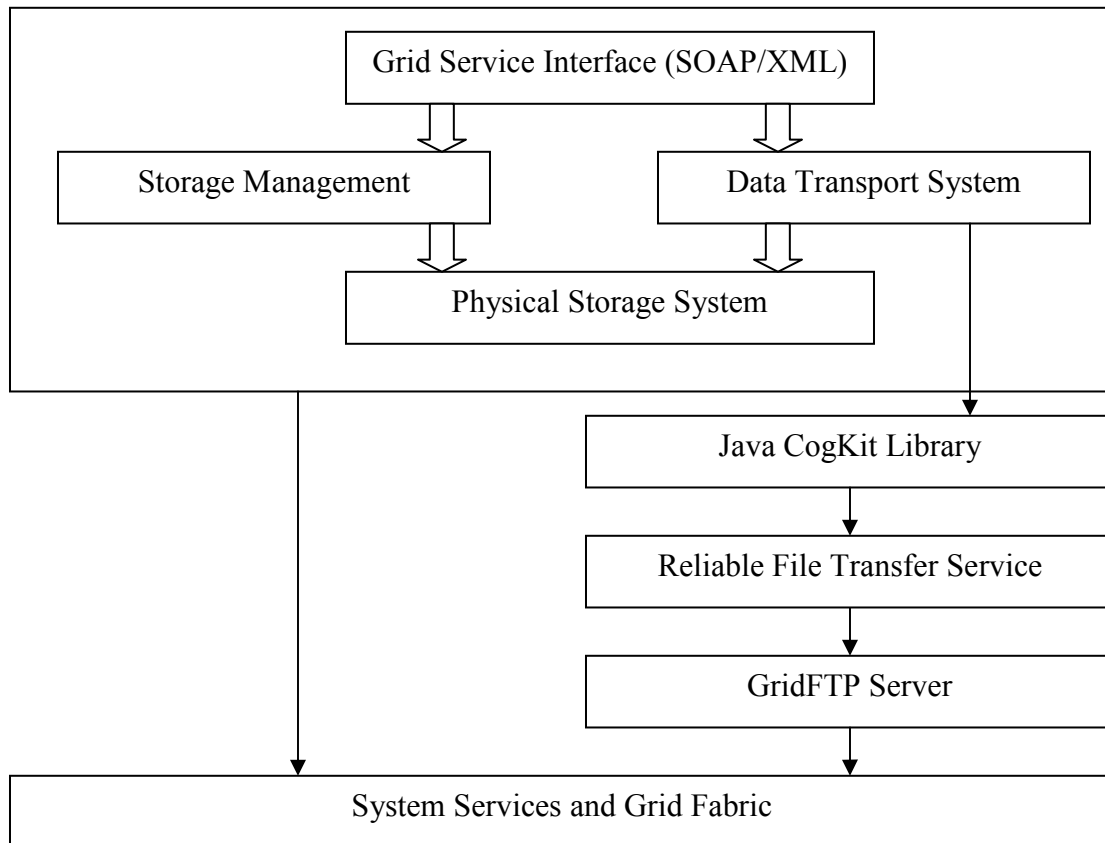


Figure 5.1: Grid Storage Service Layered Architecture

Fundamentally, the design is forced to follow the well known design patterns of GT3. The reason for this is that the storage service uses the software components of GT3. The architecture as indicated in figure 5.1 follows the logical architecture in the previous chapter. However, there is one point being slightly different with original logical design in order to adapt to GT3. That is because directly controlling GridFTP or even RFT is

complex and inconvenient; Java CogKit Library is used to play as an immediate between the Data Transport System and the actual data transport engine. More specifically, the Data Transport System component accesses through two layered services namely Java CogKit and RFT before reaching to the actual data transport engine: GridFTP. The physical storage system is disk system on the computer installed GT3.

5.2.3 Services Provided by Storage Services

There are many ways of constructing storage service primitives. However, inspired on the “bottleneck principle” of protocol design we have tried to restrict our selves to construct the most basic ones. We do not prove here that they are indeed the most basic ones, or that the set of primitives is complete. If we lack some functionality, we would simply implement one or more additional primitives. The primitive used are sufficient for the requirements of our experiment. Some care was taken to enable (not to disable) a possible extension or improvement. As seen from table 5.1, storage services primarily have to deal with the single storage elements of a storage system. In our design, each storage element is regarded as the smallest unit of data with respect to management. For instance, one storage element manages one directory on disk, which contains a set of data files. Besides, storage services have to concern with the general information of the storage system such as its capacity, the number of storage elements. It should be realized that our design reflects the logical architecture of storage service as indicated in chapter 4 (fig 4.2).

In this design, each physical storage element is represented by a unique identification. This identification is maintained by the storage service from the corresponding physical storage element created till destroyed. Through this identification, user can access to its physical storage element and control all possible actions such as transferring or retrieve data. User created identification should destroy when it no longer uses that corresponding physical element.

In some more sophisticated requirements, it is also suggested that the identification can be extended to an object that might contain more information about physical storage element. For example, it should contain the information of user created its physical storage element.

Using Grid technologies makes storage service interface naturally stateful. If we would rely too much on this feature, it would be very hard to support other interfaces at the same time. In our situation, we also have to integrate a stateless Web service (AAA service). The design of the Broker would become complicated.

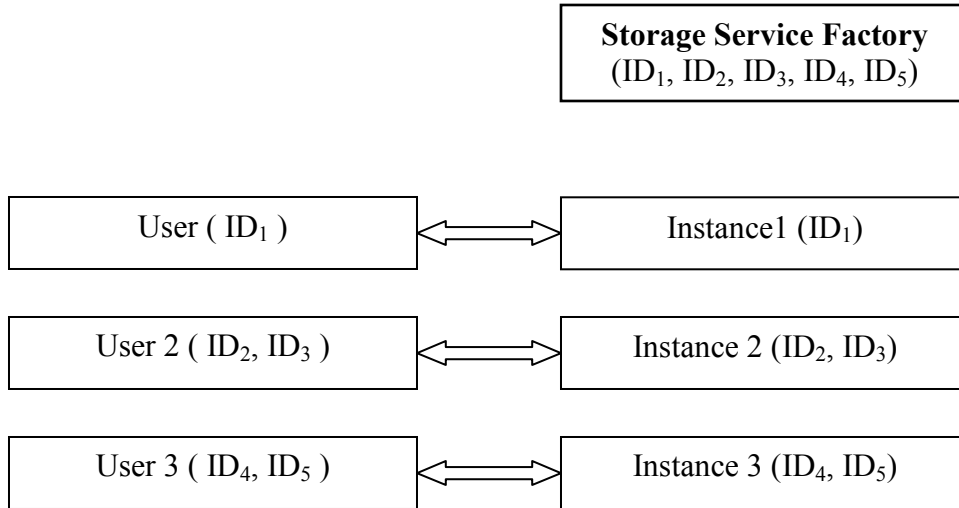


Figure 5.2: The Stateful Storage Service Using Grid Service Technologies

Figure 5.2 describes the mechanism of control of physical storage elements with Grid service technologies. Each user has its own storage service instance and holds a number of identifications that represent for the physical storage element it controls.

Table 5.1: Storage Service Primitives Description

Primitive	Name	Short Description
1	string getInfo()	<ul style="list-style-type: none"> Returning all information about the storage service.
2	string create(string ID, int size)	<ul style="list-style-type: none"> Create a storage element. Returns some related information.
3	string delete(string ID)	<ul style="list-style-type: none"> Delete a storage element. Returns some related information.
4	string getInfoOf(string ID)	<ul style="list-style-type: none"> Returns information about a specific single storage element
5	string store(string ID, string list, string params)	<ul style="list-style-type: none"> Transfer files to the storage element. List contains the addresses in which files locate. Params contain values to init GridFTP such as the number of parallel streams, TCP buffer size etc...
6	string retrieve(string ID, string RetrievePath, string params)	<ul style="list-style-type: none"> Retrieve files back. RetrievePath indicates the location in which files will be stored. Params contain parameters similar to the one in primitive fifth.

Table 5.2 describes the parameters used for the data transport mean: GridFTP. Those parameters are used by the Data Transport System component to control GridFTP.

Table 5.2: Parameter Description

Number	Name	Value	Description
1	Type of transfer	Boolean	<ul style="list-style-type: none"> true = binary, false = ascii
2	Block size	Integer	<ul style="list-style-type: none"> The size (in bytes) of the buffer to be used by the underlying transfer methods.
3	TCP buffer size	Integer	<ul style="list-style-type: none"> The size (in bytes) of the buffer to be used by the underlying FPT data channels
4	Notpt	Boolean	<ul style="list-style-type: none"> stands for: No-third-party-transfers. Notpt = true (by default), switch third-party transfers off. Notpt = false, enable third-party transfers
5	Parallel streams	Integer	<ul style="list-style-type: none"> Specify the number of parallel data connections used
6	DCAU	Boolean	<ul style="list-style-type: none"> Enable data-channel authentication (true – on, false = off).
7	Concurrency	Integer	<ul style="list-style-type: none"> Number of files transferred.

5.2.4 Web Service Approach

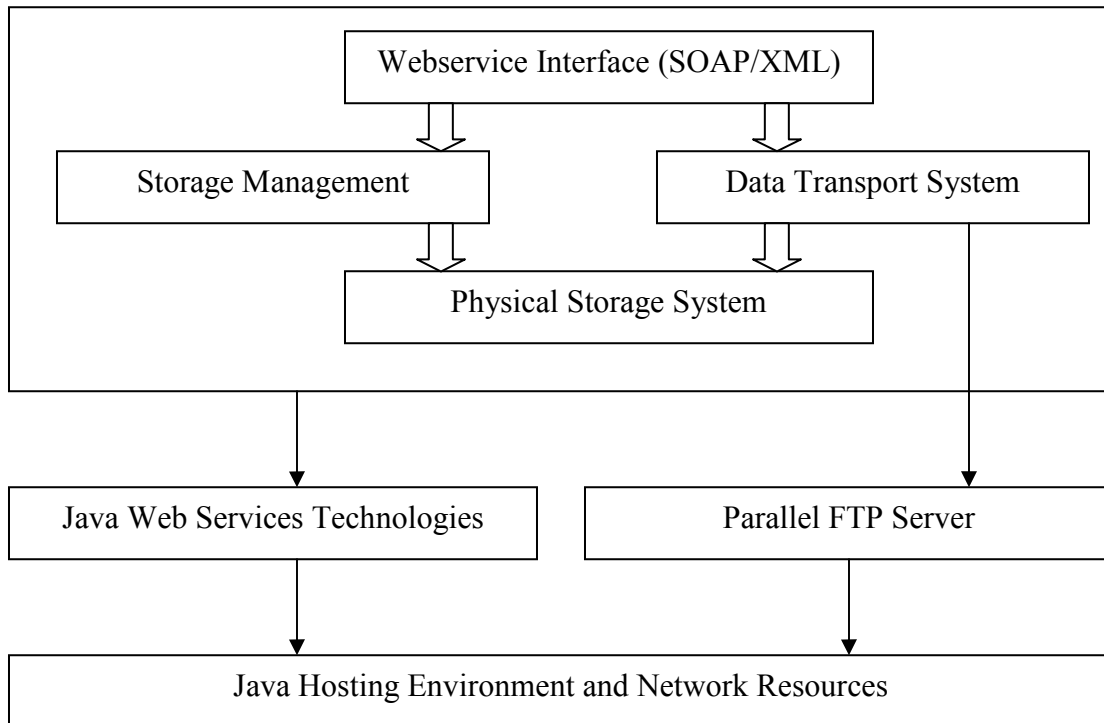


Figure 5.3: Web Storage Service Layered Architecture

Basically, the storage service architecture is not fundamentally changed if one deploys on Web instead of Grid service technologies to create a service interface. In the Web service approach, the mechanism of controlling physical storage element through its identification remains the same. The storage service interface is stateless here and the Broker, as well as the resource related software must maintain state. Indeed, the Figure 5.2 shows that storage service interface has to serve for more than one user at the same time. It is worth noting that, because the storage service interface here is stateless, it does not distinguish between users yet the physical storage elements belonging to each user can be recognized by the corresponding identifications.

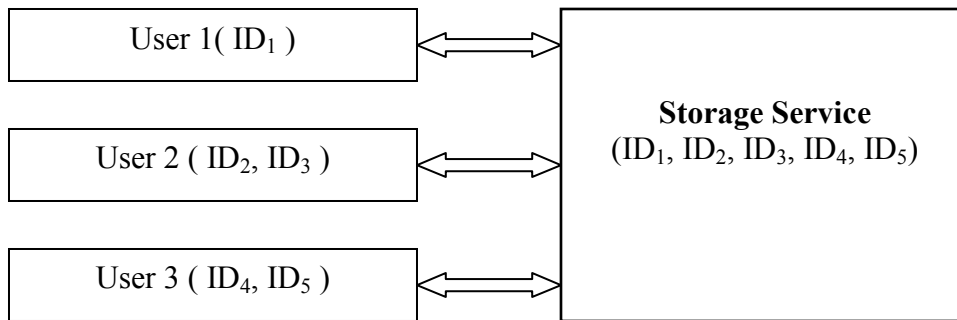


Figure 5.4: The Stateless Storage Service Using Web Service Technologies

In practice, the Web service interface is easier and a certainly good interface for the storage resources. Web service users do not have to deal with Grid security and the complexities of installing and running the Globus Toolkit. Indeed, hindsight comes with clarity: the extra functionalities offered by the Globus Toolkit did not compensate for the extra effort in dealing with this toolkit. The more mature and well-supported Web service technologies were, also in matters of security and functionality, sufficient for our operational purposes.

Figure 5.5 reflect the use of the Parallel FTP component. Parallel FTP is capable of transferring data simultaneously by stripping files into equal pieces and then sending the parts onto multiple of TCP connections, (it is worth noting that GridFTP shares this aspect with parallel FTP). Indeed, as the bandwidth dedicated for applications of physical transmission line is high (MTU is high), using only one single TCP or UDP connection results in a less efficient usage of transport capacity. One single physical transmission line is in fact only capable of sending frames consequently. However, this transmission line is able to perform efficiently if a sufficient number of frames are always available for being sent. More than one TCP connection can obviously create more available frame than one TCP connection can. Parallel FTP is designed in this manner to exploit the available bandwidth. The design of the parallel FTP program is rather simple. Despite this, it supports many useful features for our storage service. Along with the parallelism, to achieve the highest transfer rates the key parameters (such as TCP window size, buffer size for each TCP connection etc...) should be adjusted automatically or even by hand. Yet, this master thesis is centered on the subject of service integration and thus tuning

TCP parameters is out of scope. In short, architecture of parallel FTP is good enough to meet the demand for transferring huge data in high-bandwidth applications.

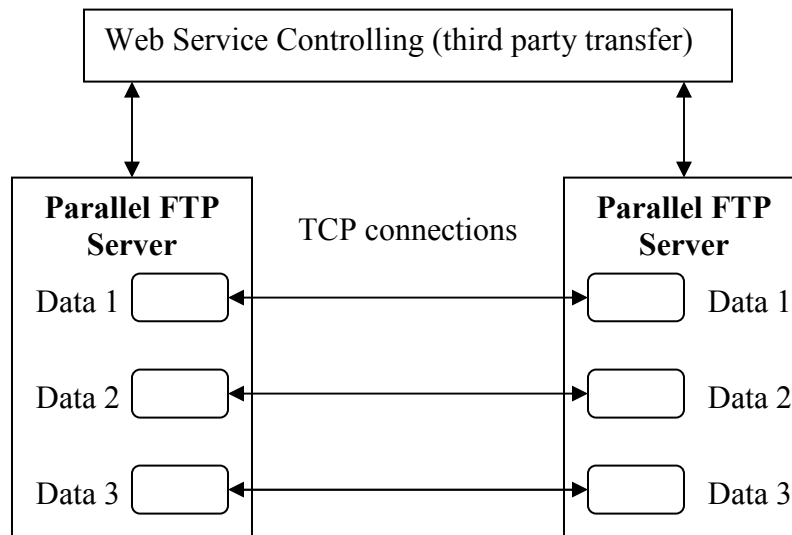


Figure 5.5: Parallel FTP Architecture

5.3. Service Provided by AAA

“Implement a simple Bandwidth on Demand (BoD) service based on Generic AAA concepts. In this project, we could proof the power of our policy language. We defined an AAA request message for a typical BoD request (XML) and constructed a policy for this purpose.” [26] Here, some details of the AAA service interface are briefly introduced to explain how one can deploy the AAA service. The AAA service is available as a stateless Web service. For now, it is only accessed by sending XML/SOAP-messages over HTTP. Users of the AAA service can setup an end-to-end connection. The AAA server (providing the AAA service) is unique at each lambda domain. Those servers are able to inter-operate to satisfy user requests. These inter-operations are transparent to users. For instance, for being able to provision lambda-bandwidth in which there are connections crossing more than one domain, the AAA server has to negotiate with its peers at different domains in order to obtain some connections. The figure below describes the current AAA service interface in XML schema:

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns="http://www.AAA.org/ns/AAA_BoD"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.AAA.org/ns/AAA_BoD">

    <xsd:element name="AAARequest" type="LambdaType"/>

    <xsd:complexType name="LambdaType">
        <xsd:sequence>
            <xsd:element name="Authorization"
type="AuthorizationType"/>
            <xsd:element name="LambdaRequest"
type="LambdaRequestType"/>
        </xsd:sequence>
        <xsd:attribute name="version" use="required"
type="xsd:string"/>
        <xsd:attribute name="type" use="required"
type="xsd:string"/>
    </xsd:complexType>

    <xsd:complexType name="AuthorizationType">
        <xsd:sequence>
            <xsd:element name="credentialID" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="LambdaRequestType">
        <xsd:sequence>
            <xsd:element name="srcPort" type="xsd:positiveInteger"/>
            <xsd:element name="dstPort" type="xsd:positiveInteger"/>
            <xsd:element name="Bandwidth" minOccurs="0"
type="xsd:positiveInteger"/>
            <xsd:element name="StartTime" minOccurs="0"
type="xsd:string"/>
            <xsd:element name="Duration" minOccurs="0"
type="xsd:decimal"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>

```

5.4. Network Service Implementation

This part describes in detail the uses of the network service in practice.. It covers the network configuration, the software design of the Broker program as well as the storage service for experiments with the new software.

5.4.1. Experiment Scenario

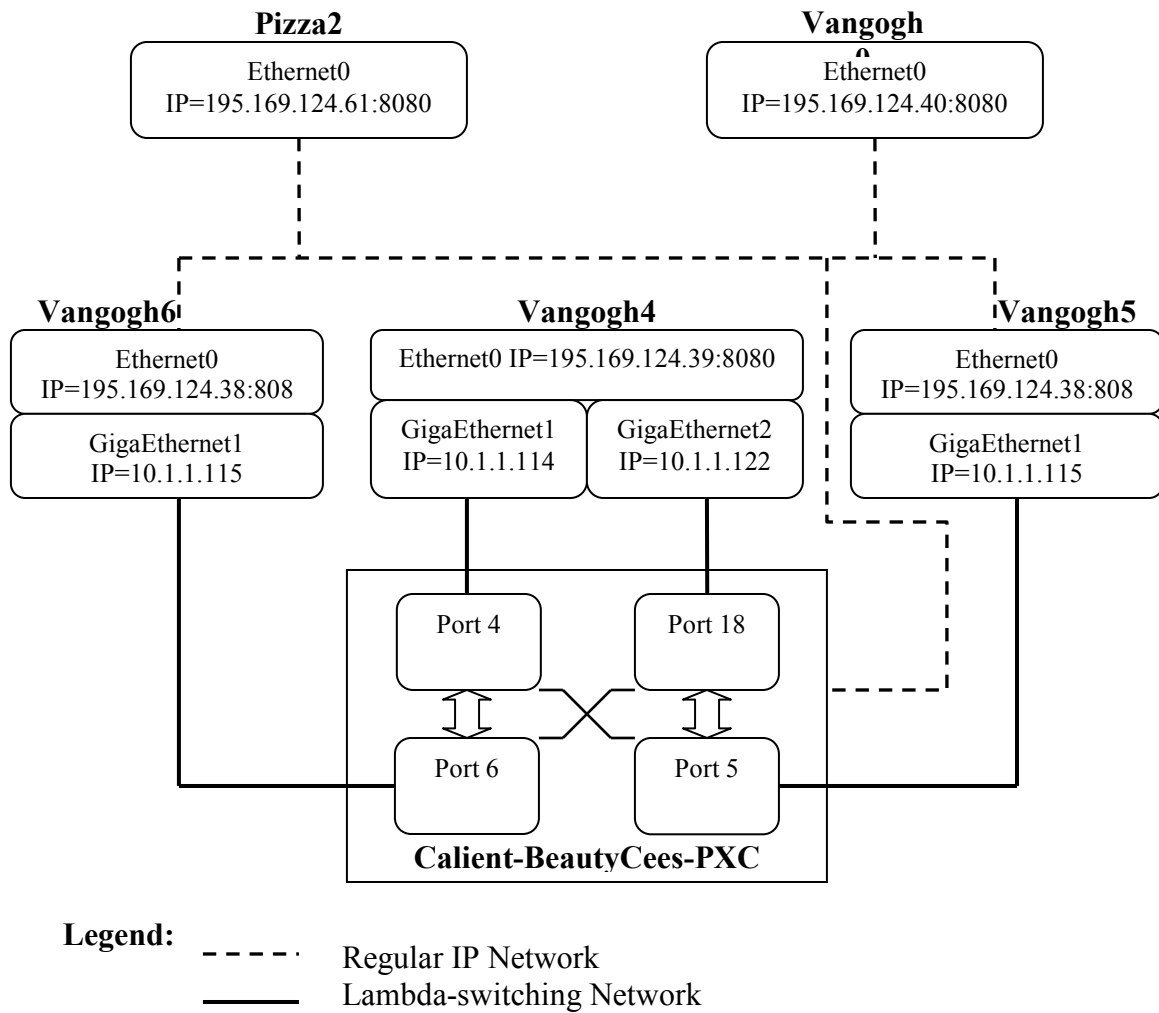


Figure 5.6: Testbed Hardware Configuration

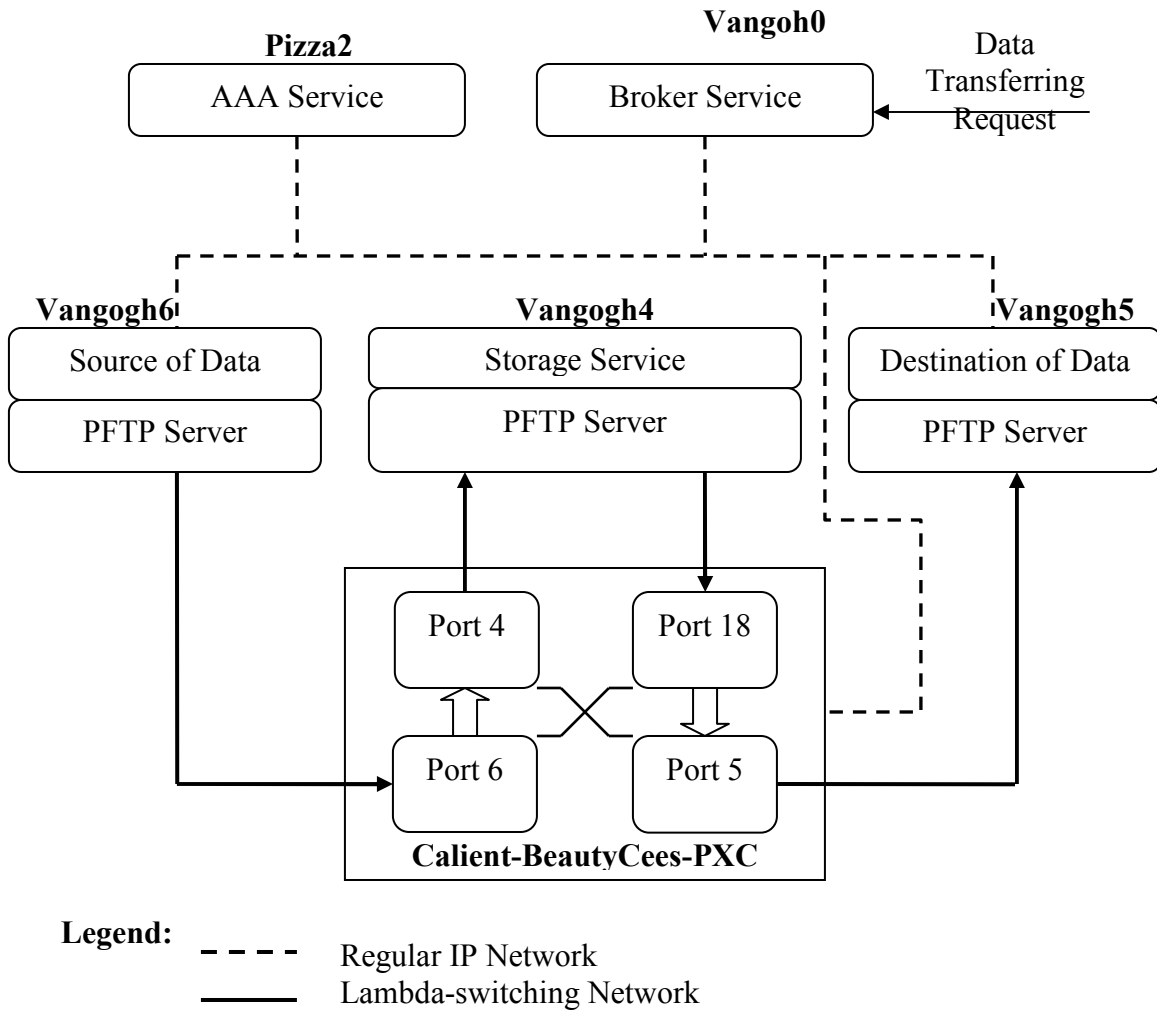


Figure 5.7: Testbed Software Configuration

The experiment is carried out on the lambda-switching network of the University of Amsterdam, Netherlight. Figure 5.6 and 5.7 depict how hardware and software are configured.

For the hardware configuration, one Calient switch, Vangogh cluster and Pizza2 are used. All of them are located at Netherlight domain. There are two sorts of network interfaces: normal Ethernet and gigaEthernet. GigaEthernet interface is used to operate on optical path while normal Ethernet interface is used to operate on regular IP network. All nodes are equipped with normal Ethernet interface. Three extra gigaEthernets are placed on Vangogh3, Vangogh4 and Vangogh5. In particular, Vangogh4 is equipped with one more gigaEthernet interfaces so as to be able to communicate simultaneously with Vangogh3 and Vangogh5. In addition, Calient switch has its own normal Ethernet to connect to AAA server on Pizza2. Obviously, this hardware topology has the control channel and the data channel separated.

For the software configuration, AAA service, Broker service and Storage service are installed respectively on Pizza2, Vangogh0 and Vangogh4. The separated PFTP servers are installed on Vangogh3, Vangogh4 and Vangogh5. Since there is only one domain in this network topology, PIN is therefore not used here. However, one should be aware that PIN does exist to deal with multi-domain issues in lambda-switching network constructed on multiple of domains.

In short, the network topology is rather simple yet good enough for the experiment.

Let us look at the case when a user requests the Broker to transfer data from Vangogh6 to Vangogh5 over lambda-switching network. The task of broker is then to start, stop, and manage actions that results in conveying data to the destination (Vangogh 5). In addition, suppose there is one storage service at the Calient-BeautyCees-PXC (it is Storage Service on Vangogh4) and broker knows precisely the network topology. The schema is as follow:

1. *The broker launches a new data transferring transaction; asking AAA Service to provision lambda from Vangogh6 to Vangogh5*
2. *If the broker fails to obtain a lambda in step 1, the broker continues with asking the storage service to arrange a storage element and the AAA service for a lambda to that storage service. Jump to step five otherwise.*
3. *If broker fails to get storage in step 2, it goes to step five else it starts transferring data to the storage.*
4. *In another transaction, the Broker asks the AAA service for a lambda from the storage location (vangogh4) to the destination (vangogh5). If that succeeds, the broker starts transferring data to vangogh5, jump to step five otherwise.*
5. *Broker reports to user whether or not it has succeeded in performing the request.*

A snapshot of the Broker code is shown in Table 3. The objective is to demonstrate how software objects, which represent storage and switching element, are used in end user applications. Furthermore, it shows how a network is represented as an application part and used as such. The real Java code contains many other classes (e.g. classes to wrap up AAA service, storage service, PFTP etc...) and is therefore much longer and more complex than the one in Table 5.3.

On this network configuration, various experiments are conducted. That is, files with different sizes are transferred through the network and the attached storage. Moreover, those files are split into smaller files sending over separate socket streams. The results are put in the appendix, because the qualitative understanding of them is not related to the subject of this thesis. The experiments were done to test the software, to tune the usability of the interface designs (i.e. for complex data transfer and storage resources) and, ultimately, to experience and demonstrate the power of the virtual resource models in the context of the new networked applications.

Table 5.3: A snippet of the Broker code

```
package broker;

public class Broker {

static final String vangogh6 = "10.1.1.116";
```

```

static final String vangogh5 = "10.1.1.115";

static final String AAAServer = "http://195.169.124.61:8080/AAA/server";

static final String LambdaPlace = "http://www.AAA.org/ns/AAA_BoD
http://195.169.124.61/LambdaRequest.xsd";

public static void main(String[] args) {

    AAAService aaa = new AAAService( AAAServer, LambdaPlace);
    try {
        Stub stub = createProxy();
        stub._setProperty( javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
                           args[0] );
        StorageIF storageService = ( StorageIF )stub;

        storageService.create( "storagel", 10 );
        storageService.create( "storage2", 20 );
        storageService.getInfoOf( "storagel" );
        storageService.getInfoOf( "storage2" );

        String[] transferList = new String[3];
        transferList[0] = "/home/space/hbui/tmp/BrokerData/Source/file1";
        transferList[1] = "/home/space/hbui/tmp/BrokerData/Source/file2";
        transferList[2] = "/home/space/hbui/tmp/BrokerData/Source/file3";

        if ( aaa.getConnection("vangogh6", "vangogh4") == Boolean.TRUE ) {
            storageService.store( "storagel", transferList, vangogh6 );
            if ( aaa.breakConnection("vangogh6", "vangogh4") == Boolean.TRUE ) {
                System.out.println( "Break Sucess 6->4" );
            } else {
                System.out.println( "Break Fail 6->4" );
            }
        }

        } else {
            System.out.println( "Get Fail 6->4" );
        }

        String retrievePath = "/home/space/hbui/tmp/BrokerData/Destination";

        if ( stored == Boolean.TRUE && aaa.getConnection("vangogh4",
"vangogh5") == Boolean.TRUE ) {
            storageService.retrieve( "storagel", retrievePath, vangogh5 );

            if ( aaa.breakConnection("vangogh4", "vangogh5") == Boolean.TRUE ) {
                System.out.println( "Break Sucess 4->5" );
            } else {
                System.out.println( "Break Fail 4->5" );
            }
        } else {
            System.out.println( "Get Fail 4->5" );
        }
    }
}

```

```
    }  
  
    storageService.getInfo() );  
} catch (Exception ex) {  
    ex.printStackTrace();  
}  
  
}
```

5.4.2. What does experimental result say?

Adding a storage service to a state-of-the-art lambda-switching network is very hard if one would employ conventional techniques of network protocol design. The network is still a work in progress. Again, it should be restated that the main goal of experiment is to demonstrate the power of the virtual network resources concept. With respect to this point, the lessons learnt from the experiment are concluded as follows:

- Abstractions to software objects of network elements are possible and worthwhile. In the Broker code, network devices (e.g. storage and switches) are represented as software objects. Those objects expose all necessary functionalities of the real network devices that the Broker used. This illustrates the ability of abstracting and integrating network resources by the corresponding virtual network resources into a useful new service. It demonstrates the power of virtual network resources concept
- Both Grid services and Web services technologies are the good choices to inter-work between network resource and its virtual network resource proxy object in the field of telecommunications and optical networking.
- By turning network elements into software objects gives way to create new networking applications. With a few lines of code a store and forward network was created by integrating individual networking and storage resources. .
- The experiment shows that the store and forward network is something better than the original lambda-switching network.

5.4.3. Grid Serviced vs. Web Services

Ideally, Grid technologies would be the preferred interface technology because the lambda-switching network of the University of Amsterdam is designed and developed for Grid-applications. For other sorts of application, the interface should be enabled with other technologies to adapt to various requirements, especially in serving for commercial applications.

Initially, we did implement Grid technologies and our applications were tested successfully on the regular IP network. Unfortunately, shortly after the moment the lambda-switching network became available we discovered that the Grid services could not run on the lambda-switching network. Grid security policies demand a proper IP number plan in order to create the proper certificates that allow the use of the service. For us, at that time, the number plan problem could not be solved. Consequently, we concentrated on the deployment of Web services technologies. Despite the fact that Web service and Grid service technologies have some fundamental differences, in the scope of

this paper, those differences do not affect our concept; meaning that both approaches are sufficiently good in verifying the concept.

5.4.4. Mathematics Perspective of Store-Forward Lambda-switching Network

At present, it is impossible to measure precisely the performance of the store-forward network with respect to blocking probabilities if one transfers data hop by hop or end to end. We have to rely to calculations to estimate this performance.

In figure 1.3, we already pointed out that the finite blocking probabilities would be able to affect to the overall performance of lambda-switching network to establish an end to end connection. At the time of requesting network service to reserve an optical path from host A to host B created by making cross-connect on n switches each switch has a certain probability of successfully making a cross connect. Hence, the probability of succeeding in reserving the optical path can be calculated as:

$$P_{A \rightarrow B} = \prod_{i=1}^n P_{switch-i}$$

If the same request is served in the store-forward lambda-switching network, assume that the optical path is formed by n switches with k-switches equipped with storages. Storage capacity at each switch can be supposed to be unlimited. The problem is that how many cross-connections can be made to the storage? If increasing the number of fibers connecting to switch can increase the probability of storing but in return at the expense of decreasing the number of connections to other switches. Hence, to some extent, each storage-switch certainly has a probability in accepting storing data. Because of the storage feature of the store-forward lambda-switching network, we can obtain:

$$P_{store-forward:A \rightarrow B} = \prod_{i=1}^n P_{switch-i} + \prod_{i=1}^k P_{storage-switch-i} \times \prod_{i=1}^{n-k} P_{switch-i}$$

$$0 \leq P_{A \rightarrow B} < P_{store-forward:A \rightarrow B} \leq 1$$

Figure 5.5 illustrates an example. It is trivial to calculate the probabilities:

$$P_{A \rightarrow B} = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{8}$$

$$P_{store-forward:A \rightarrow B} = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times 1 \times \frac{1}{2} = \frac{3}{8}$$

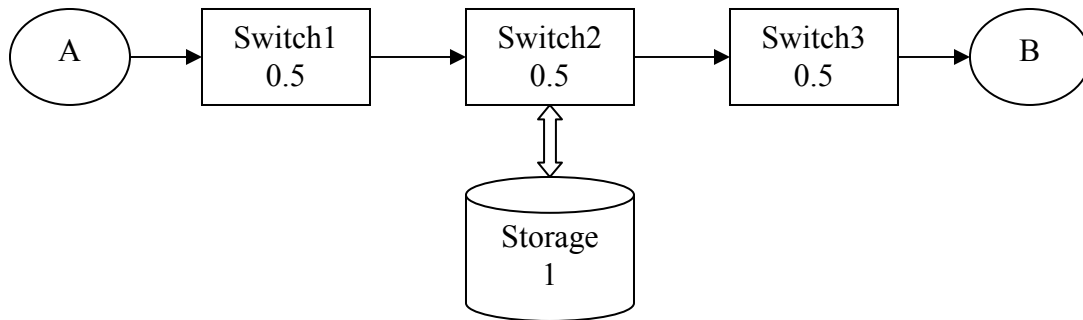


Figure 5.8: Data from host A is transferred through two non-storage-switches and one storage-switch to B.

It is worth noting it is possible for the case that in a store-forward lambda-switching network, if for instance, in figure 5.8, instead of using storage, user keeps waiting for the optical path directly connecting from A to B which might be better in terms of the total time spending for waiting and transferring. However, probabilistically speaking, using storage increases the probabilities of being served and thus improves the overall performance of the network. Therefore, it is advisable for user that it should not wait for the entire connections from A and B once it is possible to arrange connection from A to the storage at switch2. In general, when using the store-forward lambda-switching network user should accept to transfer data as close to the destination as possible if the connection to that temporary location is possible.

Because of the fact that it is difficult to experiment completely our hypothesis on the real lambda-switching network, one can think of simulating the store-forward lambda-switching network to expose completely all features and then being able to compare the performance between two sorts of network. Unfortunately, this direction falls outside the scope of this report.

Chapter 6: Conclusions and Future Directions

6.1. Conclusions

6.1.1. The Concept of Virtual Network Resources Opens an Easy Way to Create Meaningful New Networking Facilities and Applications

The concept of virtual network resources has been described and examined. This was done by designing and implementing the store and forward network service model. In practice this means that the network is augmented with data storage capabilities and that, the inter-working of the storage, the network, and the applications was done with software. This indeed turned out to be a much more flexible and direct approach than the commonly used approaches. In a more traditional approach, a complete set protocol would have to be designed and implemented to accomplish this. This is a major task whilst its outcome, the protocols, is of no direct use for the application programmers. Instead, in our approach, programmers would use software entities as objects to generate the protocols. In consequence, those surrogate objects would abstract the protocols by functions that are used by programmers.

The power of applying the concept is that it opens ways of integrating a variety of network resources from various and traditionally unrelated domains to cope with a single problem or create new services. Grid and Web service technologies have been chosen to implement the concept. These interface technologies create a bridge between the physical network element and its surrogate software object.

6.1.2. The Store and Forward Network Service Is a Meaningful Extension to an Optical Network.

We also succeeded in developing and implementing a store and forward network service by applying the virtual network resources concept.

The advantage of the store and forward network in comparison with normal lambda-switching network is it provides for some applications a faster way to transport data. The reason for this is, as we have illustrated by calculations, that the chance to obtain in a given time period an end-to-end connection is smaller than the chance to obtain a sequence of parts of such a connection.

6.3. Future Directions

There are various directions in which future work could proceed. A straightforward is to improve the technologies that are created for this thesis. Other future research directions are more conceptual.

6.3.1. Technology Improvements

The current store-forward network service software is far from a complete software production. Therefore, the current software needs to be improved in many aspects.

For the short-term purpose, we will continue working with Grid technologies; investigate into the advantages of these technologies when applying to the concept of virtual network resources. We hope that the technical problem on the current lambda-switching network will be solved soon and then we will be able to proceed to deploy GT3 on Vangogh cluster. Note that, there is not much technical difference when deploying Grid technologies although Web service and Grid service have some fundamental differences as detailed in chapter 2. The general framework and the algorithm should be the same. Despite it requires much more practical work than working with Web services technologies, network user benefits from inheriting the advanced featured of Grid technologies such as the dynamic and stateful features of Grid service, the Grid security infrastructure and the base services (e.g. RFT service) as well.

The store and forward network service built here is not yet an operational service due to the time constrains and the main objective of a master thesis project. Currently, there is a very limited set of optical networking applications. Thus, for demonstration purposes it is a good idea to somewhat harden the software.

A long-term project is to implement the algorithm 3 in chapter 4, which is missing in our current experiment. This direction is quite interesting but difficult to implement. The transaction concept assures a set of resources to be reserved and used. We have found that the transaction-support library of the Java language is not sufficient to make the network resources transactional. Instead, it rather focuses on supporting database transactions. Hence, it is essential to develop a general-purpose transaction library or to build a specific transaction library for network resources.

6.3.2. New Virtualization Concepts

In this thesis we have created a Web (and Grid) services for optical switches. A straightforward extension of this idea would be to create for routers a set of Web services (or Grid services). Since (OSI) layer three is rather rich in functions, it is not straightforward (as for example in the case of optical networks) which of them are interesting to virtualize

Researching other ways of virtualizing would probably also lead to new applications of networks. In this report, we have virtualized a network element into the Computer Science concept of software object. There are, however, other ways to represent physical objects in software. For instance, we could virtualize that same network element into a table entry in databases. Here, network elements are used and managed like data elements in the database systems.

Most interestingly, is the virtualization of a network element into an instruction of an abstract computer – the virtual machine as used with the Java and .Net environments?

Such a virtual-machine-instruction set facilitates the incorporation of data communication functions into compilers, opening very intriguing perspectives to routing problems and distributed computing. For instance, a compiler can construct instructions to set up a very optimal end-to-end path for a specific application at running time.

A compiler could generate instructions to replicate automatically the state to online peer programs. Changing the way one virtualizes the resources therefore opens up new ways of exploiting them.

6.3.3. The Virtualization of Resources Outside the Field of Information and Communication Technologies

Furthermore one could attempt to virtualize objects that do not belong to the domain of information and communication. What would happen if we would apply the virtual resource concept to a railway, a house or a teaspoon? The answer might be that one is building an Information Society.

One possible signature for an Information Society is that the interaction over the Internet determines what is going to happen in the real world. We have seen that resources become virtual when they are abstracted by computer software and are available on the Internet. Information Society building then, needs only three steps that should be endlessly repeated. The following example illustrates this. Suppose one has a group of things. The following steps would create an information society technology:

Step 1: Add computational and internet capabilities

- Build in a computer
- Give that computer control over the resource
- Link the computers (via the Internet)

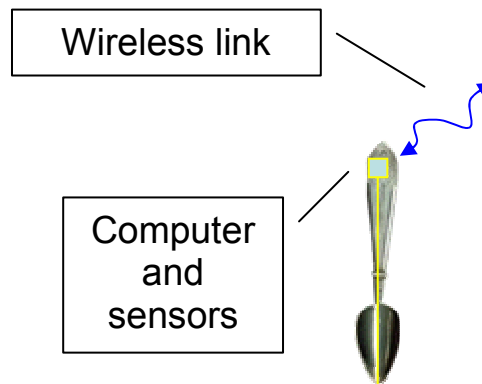


Figure 6.1: Computer software controls resources

Step 2: Create a virtual “broker” for a group of things

- create software that offers easy, controlled, service, to many “customers” via the Internet
- Expose the combined service as a new online resource

- Forget about the individual things

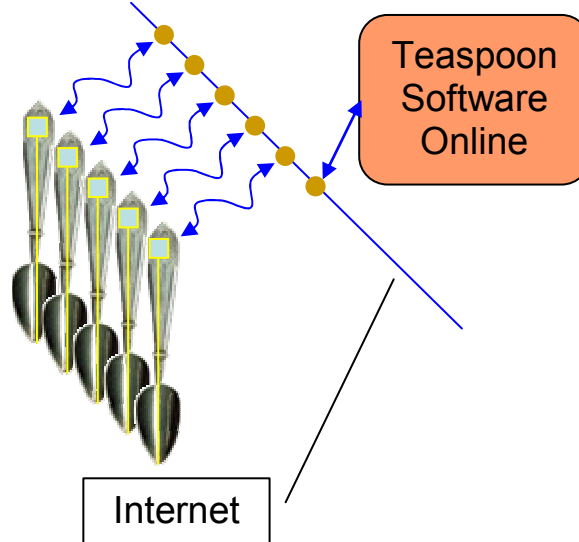


Figure 6.2: Group things by virtual agent

In the third step, the interaction between the resources is coordinated through software, via the Internet

Step 3: Service integration and creation

- Software integrates the services and interactions of various resources into a new service

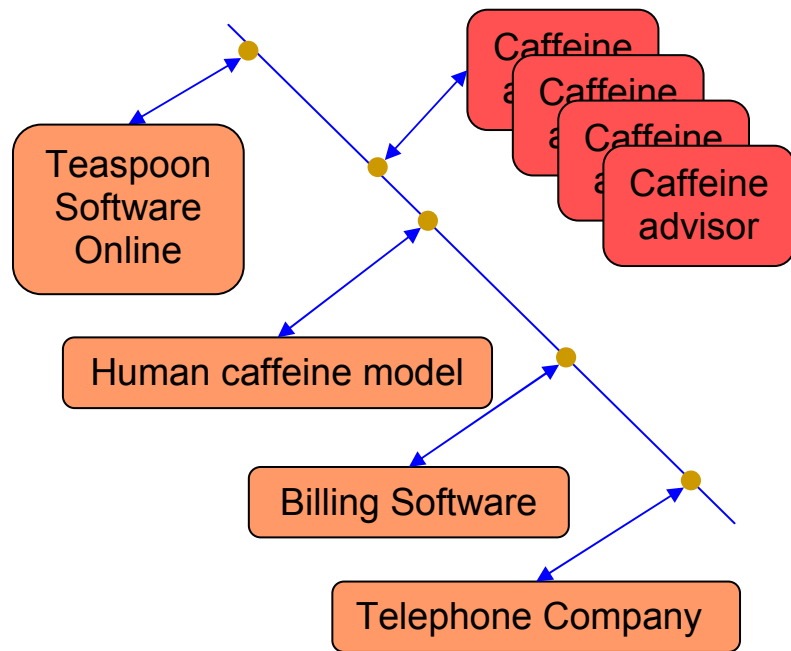


Figure 6.3: Online resources are integrated to create a new resource

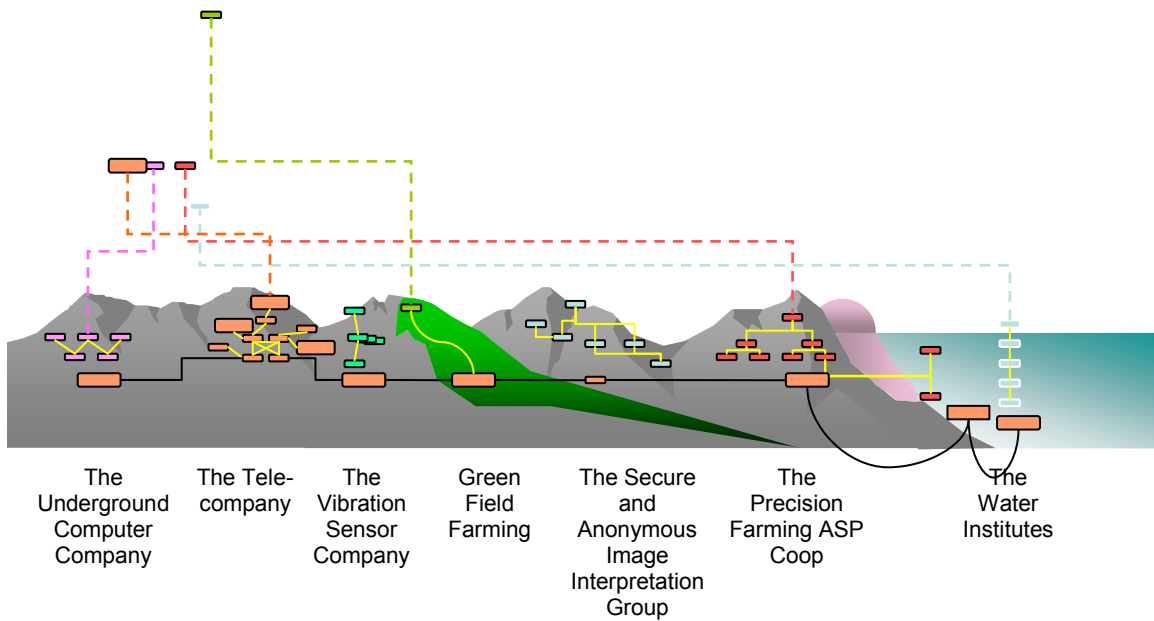


Figure 6.4: Software is the nuts and bolts that holds the Information Society together

Figure 6.4 shows an extreme demonstration of this idea in the context of farming. For every complex resource needed, there is an agent, a broker that exposes their capabilities to on the Internet (e.g. see 4.2, chapter 2). Regardless of the distances and the differences

of those resources, one merely uses the virtual resources to control the real resources by creating an integrated application. The pseudo-code in Table 6.1 illustrates how a broker can use the resources. To do the farming, a series of jobs need to be done by using a number of resources. Interestingly, the farming management itself is also exposed as an object containing the methods to control all necessary resources. In short, the virtual resources concept seems to be a viable way to handle a complex inter-working of complex physical entities. That viable way would lead to the Information Society.

Table 6.1: The Broker does the farming-field-management job

```
import java.FarmingApplication.*;
{
/* Test program for optimal farming field management, trial 10:
see if we can image analyze the field, measure soil composition and get
an irrigation */
public class FarmingIrrigation{
    public void doFarming(){
        // declare and initialize the farming field object
        farmingField field = new farmingField("GreenField", "Lot 7");
        Try{

            //connect the video measurement system for 20 seconds
            field.Image = new VideoImage(field.observationPointSet(19),"20 s");
            //measure soil composition
            field.Soil = new SoilImage(field.observationPointSet(19));
            // Obtain growth analysis, but first reserve computer and network capacity
            // then load the growth program
            field.State.GrowthAdvise = new field.farmingAnalysis.growth("TUCCC",
                "Telecompany", "GreenFieldProgramGrowthTest10");
            // Execute only the irrigation advice
            irrigationResult IR = new field.farmingAnalysis.growth.irrigate
                ("Water Institute");

        } Catch {
            message("Experiment not successful");
        }
    }
}
```

Appendix

In this appendix, these results are only here to give an idea about to performance of the application. Note that the main aim of this report is to investigate into the concept of virtual network resources. This concept is demonstrated by integrating a storage service with a lambda-switching network to form a store-forward network. No attention at all is paid to optimize its performance. However, analyzing the performance, the network scalability could be attributed to future directions.

Table 7.1: Transmission Rate Measurement

Number of Streams	1	2	3	4	5	6
Stream Size (gigabyte)	1	1	1	1	1	1
Time (minute)	0.56	1.334	1.65	2.22	2.61	3.07
Effective Transmission Rate (Gbps)	0.238	0.199	0.242	0.240	0.255	0.261
Data Rate (Gbps)	1	1	1	1	1	1

Table 7.2: Transmission Rate Measurement

Number of Streams	7	8	9	10	1	2
Stream Size (gigabyte)	1	1	1	1	2	2
Time (minute)	3.67	3.96	4.67	5.30	1.09	2.01
Effective Transmission Rate (Gbps)	0.254	0.269	0.257	0.251	0.241	0.265
Data Rate (Gbps)	1	1	1	1	1	1

Table 7.3: Transmission Rate Measurement

Number of Streams	3	4	5	6
Stream Size (gigabyte)	2	2	2	2
Time (minute)	3.08	4.05	4.91	6.26
Effective Transmission Rate (Gbps)	0.260	0.263	0.271	0.256
Data Rate (Gbps)	1	1	1	1

- By convention, Effective Transmission Rate (ETR) is calculated as the measured number of units of data, such as bits, characters, blocks, or frames, transmitted during a significant measurement time interval divided by the measurement time interval.

- For example, the ETR in the first column can be calculated as follows:

$$ETR = (Number\ of\ Streams * Stream\ Size * 8) / (Time * 60)$$

Replacing symbols by numbers, we obtain:

$$ETR = (1 * 1 * 8) / (56. * 60) = 0.238$$

References:

- [1] The Future and Telematics, Robert Meijer, UvA and TNO.
- [2] Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation 04 February 2004.
- [3] XML Schema, W3C Recommendation
- [4] Web Services Description Language (WSDL) 1.1, W3C Recommendation
- [5] SOAP Version 1.2, W3C Recommendation
- [6] Web Services Architecture, Web Services Architecture Requirements, Web Services Architecture Usage Scenarios, Web Service Management: Service Life Cycle, W3C Working Group
- [7] What is the Grid? A Three Point Checklist. I. Foster, GRIDToday, July 20, 2002.
- [8] The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. I. Foster, C. Kesselman, J. Nick, S. Tuecke, Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002. (extended version of Grid Services for Distributed System Integration)
- [9] Open Grid Services Infrastructure (OGSI) Version 1.0. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, D. Snelling; Global Grid Forum Draft Recommendation, 6/27/2003.
- [10] Grid Services for Distributed System Integration. I. Foster, C. Kesselman, J. Nick, S. Tuecke. *Computer*, 35(6), 2002.
- [11] The Anatomy of the Grid: Enabling Scalable Virtual Organizations. I. Foster, C. Kesselman, S. Tuecke. *International J. Supercomputer Applications*, 15(3), 2001. Defines Grid computing and the associated research field, proposes a Grid architecture, and discusses the relationships between Grid technologies and other contemporary technologies.
- [12] The Grid: A New Infrastructure for 21st Century Science. I. Foster. *Physics Today*, 55(2):42-47, 2002.
- [13] The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. *Journal of Network and Computer Applications*, 23:187-200, 2001 (based on conference publication from Proceedings of NetStore Conference 1999).
- [14] Computational Grids, I. Foster, C. Kesselman. *Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure"*, Morgan-Kaufman, 1999.
- [15] Globus: A Metacomputing Infrastructure Toolkit. I. Foster, C. Kesselman. *Intl J. Supercomputer Applications*, 11(2):115-128, 1997. Provides an overview of the Globus project and toolkit.
- [16] GridFTP Protocol Specification (Global Grid Forum Recommendation GFD.20). W. Allcock, editor. March 2003.
- [17] GridFTP Update January 2002. W. Allcock, J. Bresnahan, I. Foster, L. Liming, J. Link, P. Plaszczac. *Technical Report, January 2002*.
- [18] Introduction to DWDM for Metropolitan Networks, CISCO, White Paper,
- [19] The Cisco IP + Optical Unified Control Plane: Accelerating Service Velocity with IP Enabled Provisioning. White Paper.
- [20] RFC 3036, Label Distribution Protocol (LDP)
- [21] RFC 3214, LSP Modification Using CR-LDP

- [22] RFC 2903, Generic AAA Architecture
- [23] RFC 2904, AAA Authorization Framework
- [24] RFC 2905 AAA Authorization Application Examples
- [25] RFC 2906 AAA Authorization Requirements
- [26] <http://www.science.uva.nl/research/air/projects/aaa/>
- [27] <http://wwws.sun.com/software/jini/>
- [28] <http://www.upnp.org/>
- [29] <http://www.google.com/apis/>
- [30] From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution. K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, S. Tuecke, March 5, 2004.
- [31] The WS-Resource Framework. Karl Czajkowski, Donald F Ferguson, Ian Foster, Jeffrey Frey, Steve Graham, Igor Sedukhin David Snelling, Steve Tuecke, William Vambenepe. 03/05/2004
- [32] <http://www.google.com/apis/>
- [33] <http://www.evl.uic.edu/>