

MASTER'S THESIS

A Server Based Architecture for Advance Reservations in a Link-State Domain

Kjell Hedström

Civilingenjörsprogrammet Datateknik

Institutionen för Systemteknik
Avdelningen för Datorkommunikation

A Server Based Architecture for Advance Reservations in a Link-State Domain

Kjell Hedström
kjehed-6@sm.luth.se

22nd March 2001

Abstract

Reservations of resources in networks become important when applications, users or service classes require that their specific demand will be honored. In the Internet of today resource allocation through reservations is possible even though current techniques in many ways are unsatisfactory. Contemporary reservations they demand that routers are active in the reservation process. A known desire for other ways of making reservations, whether for the current or for a future time, has resulted in research that shows that reservations are feasible even when routers only have a passive role in the scheme. This method commands that at least one other network component, but the router, to have an active role.

This Master's Thesis aims at investigating how to design a server for advance reservations, by designing and suggesting the inner workings of the server and its interactions with other network components, with a goal to have a system for advance reservations that easily can be extended with new functionality. Since the server is to be used in future experiments and research, it is especially important to allow changes in an uncomplicated manner.

Routing software from the GateD package was examined to investigate how data was handled in OSPF routers and how to interact with them, with minimal or no changes to the original software.

The results of this work are a suggested design, and a prototype implementation, of a server that implements acquisition and maintenance of OSPF topology information. The server is constructed with the purpose of being the starting ground for future extensions and research.

Future work involves further design and construction of an infrastructure for advance reservations. This involves computation of routes in an advance perspective, communication with a variety of routing and Internet protocols, design and implementation of path selection algorithms that will run upon the server's data structures.

Preface

This work was part of on-going research at the Electrical Engineering Department's Multimedia and Networking lab at the University of Pennsylvania. The majority of the project was done during the fall and early winter of 2000, 2001.

This work was realized with the help and support of many people. Their guiding efforts were indispensable.

First I would like to thank Professor Roch Guerin, for giving me the opportunity to be part of his research group and for his expertise, suggestions and guidance which were encouraging throughout the project. My advisor in Sweden, Olov Schelén, who inspired me to do the thesis at the University of Pennsylvania.

I would also like to give thanks to all friends and colleagues at the Multimedia and Networking lab for their ideas and aid.

To Pär, Per, Robert and Sabina who all encouraged me and prompted me to take this chance at the University of Pennsylvania. Other friends in Sweden as well as in Philadelphia, USA, who have helped me in small, but significant ways all this time. You know who you are.

I am grateful to my brothers and parents, Jan-Olof, Märta, Hedas and Simon for supporting me and motivating me during this time.

Finally I wish to thank my wonderful girlfriend, Anita, for having faith in me.

Kjell Hedström
Philadelphia, USA
January 2001

Contents

Abstract	i
Preface	ii
I Introduction and Theory	1
1 Introduction	2
1.1 Thesis Background	2
1.2 Goals	2
1.3 Related Work	3
1.4 Scope	3
1.5 Outline	3
2 Theory	5
2.1 Routing	5
2.1.1 Routing Generics	5
2.2 OSPF	5
2.2.1 Information Sharing	5
2.2.2 Areas and routers	6
2.2.3 Link-State Packets	7
2.3 GateD	7
3 Analysis	8
3.1 A Server with OSPF interaction	8
3.1.1 LSA Eavesdropping	8
3.1.2 OSPF complexity	8
3.2 The Network Picture	8
3.2.1 Intra-Domain Topology	9
3.2.2 Inter-Domain Topology	9
3.3 Link Bandwidth Over Time	9
3.3.1 A structure for link resources over time	9
II Design and Implementation	11
4 Design	12
4.1 Overview	12
4.1.1 Module Interaction	12
4.2 The Server Daemon	14

4.2.1	Program Details	14
5	Server Modules	15
5.1	Overview	15
5.2	Task Scheduling	15
5.3	GateD extensions	15
5.4	Topology Data	15
5.4.1	Link-State Objects	16
5.4.2	LS Object Specifics	16
5.5	Route Tree	16
5.5.1	Path Calculation	16
6	Topology Management	19
6.1	LSA Insertion	19
6.1.1	Data Mapping	19
6.2	Maintenance	19
7	Implementation	20
7.1	GateD Extensions	20
7.1.1	Reason for Server - Router Communication Scheme	20
7.2	Prototype Implementation	20
7.2.1	Proof of Operation	20
7.2.2	Network Interface	21
7.2.3	Topology Information and Database Buildup	21
III	Conclusions and Future Work	22
8	Future Work	23
9	Conclusions	24
10	Work Evaluation	25
10.1	Performed Tasks	25
10.1.1	Installation	25
10.1.2	Literature Study	25
10.1.3	Software and Code Studies	25
10.1.4	Design and Thesis Documentation	26
10.1.5	Server Implementation	26
	Glossary	27

Bibliography	29
---------------------	-----------

Appendix	32
-----------------	-----------

A Router-LSAs	33
----------------------	-----------

List of Figures

1	Shortest Path Tree	6
2	Intra and Inter Area Communication for three areas that together makes an Autonomous System.	7
3	System Overview	13
4	Area Data Structure	17

Part I

Introduction and Theory

1 Introduction

The introduction describes some generic background about Quality of Service (QoS¹) and network resource reservations with a look at some previous work. After follows a discussion of the goals of this thesis and how they have been achieved. Finally the outline of this work is described.

1.1 Thesis Background

The Internet is growing at a very fast pace and has done so for a number of years. As the demand for network performance increase, scarcity of available resources may become an issue. Congestion occurs when the amount of information to be transferred is greater than what the data communication path can carry; which inevitably leads to packet loss and decrease of throughput for the involved packet flows. QoS is about controlling situations when network congestion occurs, to assure that chosen packet flows get better service than other flows.

One way of assuring QoS to flows is to reserve resources in networks. A common way to perform these reservations is with immediate reservations² which can be done using [RSVP]. Although RSVP deals with resource reservations it does not provide a mechanism for determining a path that has adequate resources for the requested QoS (see [QoS-Routing], chapter 3.). Mechanisms that provide QoS work often accordingly with RSVP.

Research that address the issue of QoS routing, resource reservation and the intersection of the two is currently of high importance. The demand is high on new or complementing solutions. In research by Roch Guerin et al in [RSVP/OSPF] and [QoS/OSPF], an incorporation of QoS in [OSPF] was presented. This extension of OSPF did not deal with the topic of advance³ reservations and a merging of this with OSPF was therefore planned. However, an OSPF extension in this manner leads to problems with performance⁴ and traffic overhead.

New research at the Multimedia and Networking Lab, at the University of Pennsylvania, targets the design and implementation of an infrastructure that supports route computations for advance reservations. This involves interacting with current network protocols as well as the design and implementation of path selection algorithms together with necessary data structures. To construct a system that can be used for algorithm testing and enables advance reservations in a network, without obstructing the workings of the running routing protocol, is therefore the main objective for this thesis.

1.2 Goals

Initially the goal of the thesis was to add another dimension to OSPF, a temporal one which allowed advance reservations. To achieve this the topology database at the router needed to be extended for tracking of current as well as future bandwidth availability on network links. A second objective was to modify the current path selection algorithm so that it would support computing paths in the future and allow flexibility for the reservations.

However, as the work progressed and the theory became resolved it became clear that the goal should instead be to move the burden of the advance reservations, path calculations and all the necessary new data structures from the routers to another network entity. The final goal of this work evolved into a design for a server that would be in charge of these reservations while maintaining information exchange with routers.

As this server will be in use for future research at the University of Pennsylvania one important aspect has been to make it extendible for new functionality and flexible to changes. The requirements that constituted this work can be summarized as:

¹Quality of Service is a broad generic term for assurance of quality aspects.

²Immediate reservations are for the current time or the immediate near future.

³Advance reservations deals with reservations in the near or far future.

⁴One example of this is the necessary extensions to the work performed by OSPF routers

- Investigate and design a server based architecture that allows for advance reservations.
- Network information retrieval should be made with only a few or no changes to current routing software.
- Design and implementation with a modular approach. Module communication should be made through clean, generic interfaces.

1.3 Related Work

This thesis describes an architecture where a server is in charge of performing operations to ensure QoS routing with resource reservations. This idea is not new and is in fact discussed in several important papers. Schelén, at Luleå University of Technology, has in [Schelen] approached this area with an agent based infrastructure that provides end-to-end QoS even when dealing with multiple domains. Other important research that deals with this is [Qbone]⁵ and [QboneBB]. An influential contribution from [IETF] has been [QoS-Routing]. A similar approach as Schelén's, that also deals with an architecture for advance reservations is presented by Berson in [Bers].

The research mentioned above deals with important aspects when regarding a server for route computation, and resource reservations, in an advance perspective. This thesis took place to investigate and to start implementing a structure that can be used for research purposes in the area of algorithms for advance route computations. A infrastructure of this kind where the importance is on calculations, research and performance testing of these algorithms is something not thoroughly investigated in other papers. Herein lies the main difference between these publications and this work.

1.4 Scope

This thesis addresses the theory and the design behind a server able to perform advance reservations. The focus of this thesis is in the parts behind the server that concentrates on the information retrieval and the topology picture data structure buildup, made possible from the received data.

A server of this kind and magnitude must have several functionalities, which however are not the focus, and beyond the scope, of this thesis. These parts deals mainly with the usage of the topology data. They include:

- Functional interface to routers, or other network clients, for communication regarding network resource reservations.
- Storage of reservations and saving of reserved paths.
- Executions of reservations.
- Communication with other network entities than OSPF running routers⁶.
- Admission control and path lookups with different algorithms.

1.5 Outline

This Master's Thesis has two sections. The first section gives the background information and a analysis of how to use the available topology data. The second part describes the actual design behind the server, its implemented prototype and also what mechanisms that are in work in the server. The thesis is concluded with conclusions and a work evaluation. The outline of the thesis is as follows:

⁵A QoS provided part of Internet2

⁶Which could mean, in the finished state, communication with other servers.

Part I - Introduction and Theory

- **Chapter 1** gives the thesis introduction.
- **Chapter 2** describes the background theory for the thesis: routing, the routing protocol in use and the routing software that was used. This chapter is to make the reader familiar with the concepts and terms involved in network routing and mechanisms.
- **Chapter 3** explains and analyses the theory behind the information exchange between server and routers; how it can be used and how the future aspect of link available bandwidth can be defined. Even more important is the description of core functionalities of the server that are attended here.

Part II - Design and Implementation

- **Chapters 4 to 6** discuss and explain the actual design, with choices and decisions.
- **Chapter 7** explains the implementation details in the realized server prototype.

Part III - Conclusions and Future Work

- **Chapter 8** mentions the continuation of this work.
- **Chapter 9** gives the conclusion of the thesis.
- **Chapter 10** finishes the thesis with a personal evaluation of this work.

2 Theory

Routers are the junction points inside and between networks. Their mission is to maintain and share network topology information between other routers so that they can forward incoming data packets toward their right destinations. Routing is the term used for the method of distributing and computing network topology information between routers.

The information exchange between routers is used to buildup a topology picture at each router. Information about the status of each router's interfaces is shared and spread throughout the network(s) until the routers can calculate the best, or shortest⁷, way to forward incoming packets. Since this information gives a picture of a network's topology it is of necessity for the server, that is discussed here, to participate in this communication.

These sections gives an introduction to routing, the routing protocol OSPF, [GateD] a routing software package and how it was used in this work.

2.1 Routing

Internet is divided into Autonomous Systems (AS⁸). Hierarchical routing is achieved by splitting the routing into two main groups, IGP and EGP. The first perform intra-domain routing while the latter does inter-domain routing. With the intra area routing two different kinds of protocol are in use: Link-State and Distance Vector. An example of the first kind is the well used OSPF which is further described in section 2.2. RIP and BGP are two common examples of EGPs.

2.1.1 Routing Generics

All routing protocols dictate how and what kind of information to be exchanged between routers. The data within these information packages differ between protocols but the general purpose for them is to commune route changes and to re-confirm router status.

All gained data from the information exchange is stored in a routing table, specific for each routing protocol. An algorithm calculates the best path or next destination to forward incoming packets to, for every destination in the network. The routing table is the result from the computations this algorithm. The table is used for maintaining information of where to send incoming packets⁹

2.2 OSPF

OSPF was developed as a TCP/IP routing protocol to meet the demand in the Internet community for a high functionality Interior Gateway Protocol. OSPF is based on link-state technology and important concepts like authentication of route updates, variable length subnet masks and route summarization. The following chapters describe some of OSPF's important features.

2.2.1 Information Sharing

A link describes an interface on a router and its state describes it and its connectivity with neighbor routers. A link description would normally include its IP-address, the mask, what type of network and routers to that network it is connected to.

⁷The *best* or *shortest* way is calculated by choosing the most cost efficient links. The criterion that decides how much it will cost to traverse a link is commonly made to reflect its bandwidth capacity. Other criterion such as latency may however be reflected in the cost as well.

⁸An example of an Autonomous System can be seen in Figure 2, chapter 2

⁹Although the actual forwarding of packets will happen with the usage of a subset of the routing table, called a forwarding table.

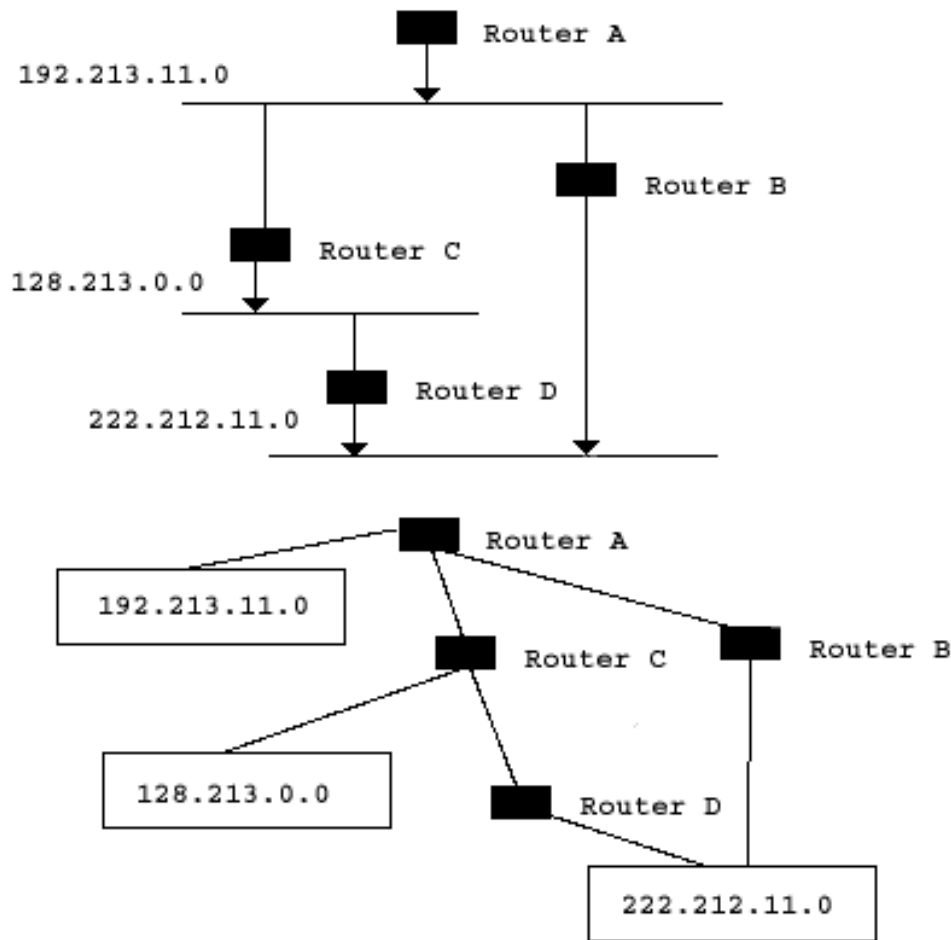


Figure 1: Shortest Path Tree

OSPF produces individual LSAs which describe links. The collected LSAs of all these states, from the routers within a domain, form a link-state database. The LSAs are flooded throughout the domain so that each participating router has an equal copy of the database.

When the building and sharing of the database is complete, each router uses Dijkstra's algorithm and calculates a shortest path tree (see Figure 1) to all destinations. This information makes the router's routing table. When a new router is initialized or if route changes occur, new LSAs are generated and flooded for re-computation of the link-state database.

2.2.2 Areas and routers

Routers within an area are oblivious of the detailed topology information external to that area. Splitting an AS into smaller areas leads thus to reduction of the running cost of the Dijkstra algorithm and also to routing protection.

Four types of routers exist. The internal routers which is only connected within the area, the Area Border Routers(ABR) that connects to multiple areas, the backbone routers which interface to the backbone area and AS Border Router router that communicates with routers in other ASs. Figure 2 shows intra and inter area communications with different kind of routers.

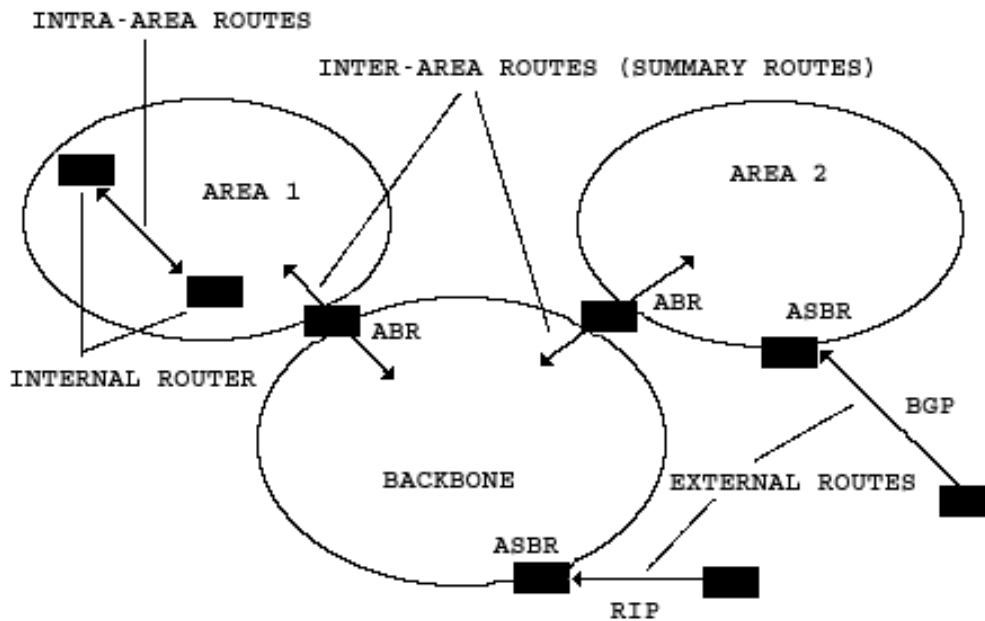


Figure 2: Intra and Inter Area Communication for three areas that together makes an Autonomous System.

2.2.3 Link-State Packets

There are four different kinds of Link-State packets that make up the OSPF database.

- Router and Network LSAs describe how an area's routers and networks are (intra area) connected.
- Summary-LSAs are generated from ABRs and describe a route to a destination outside the area but within the AS. The routes described can be to networks or to ASBRs.
- AS-external-LSAs are flooded throughout the AS. They are generated by ASBRs and each describe a route to a destination in another AS.

2.3 GateD

GateD [GateD] is a routing software package that contains most of the routing protocols that are being used in the contemporary Internet.

In this context the server will be communicating with GateD running routers; where OSPF will be the routing protocol in use. The free version of GateD¹⁰ was used in this project.

As explained in chapter 4 and 5 some changes were made to the GateD package. Those changes involved minor modifications to the actual GateD daemon program and to the software for OSPF. These changes are not supposed to be in the actual finished server but were used in this project since it facilitated the implementation of the server.

¹⁰GateD version 3.6.

3 Analysis

This chapter describes the theory fundamentals behind the construction of the server; what parts that must be addressed and how to do it. It defines what tasks that are necessary to deal with when implementing a functional server and what the server's job is supposed to be. These tasks include: information exchange between routers and the server and topology data picture buildup with an advance perspective.

3.1 A Server with OSPF interaction

The design and construction of a server, that has all the necessary information required for advance reservations, is a complex work that involves interaction between different data structures and network protocols.

The server needs to get information about the route states¹¹ in order to buildup a database that represents properly the topology situation in an OSPF domain¹²

Link changes can occur that affects the topology and the OSPF routes. In order to have accurate, current information it is necessary to participate in the OSPF state sharing. OSPF is a complex routing protocol and even small link changes can cause huge amounts of re-calculations and information flooding. The system needs to deal with this complexity. However, to avoid heavy implementation cost an eavesdropping method was designed.

3.1.1 LSA Eavesdropping

By re-using (eavesdrop on) the LSAs that are sent to a router, the server can glean the same OSPF link-state database as other routers in the area. This requires the server to use this scheme in each area that is to be considered. A good property of this method is that protocol overhead is small and that it is scalable. A requisite is however that at least one special eavesdropping entity is placed per OSPF area and that these eavesdropping boxes need a lot of OSPF functionality in order to operate satisfactory.

In this work ordinary Linux running PCs, with GateD installed, were used to provide the OSPF functionality necessary to get the LSAs to the server.

3.1.2 OSPF complexity

OSPF is a relatively complex router protocol and a requirement for using its LSAs is that a lot of OSPF functionality need to be incorporated into the server.

To reduce the implementation cost for the OSPF specific parts, an approach was taken to avoid redundant actions that already were performed at the OSPF routers. These actions involve some of the OSPF LSAs error checking. The server use these features at the router, instead of implementing them anew. Also, only the packages necessary for the buildup of the network topology are sent to the server; all other packages that involve the maintenance and synchronization of OSPF in the domain are disregarded.

3.2 The Network Picture

Each connected¹³ area gives enough OSPF link-state packages to accurately provide the server with the same network picture as each router in that area. With these LSAs a database can be composed that reflects the topology and connectivity between all connected areas.

¹¹Section 2.2 explains the distribution of these link-states.

¹²In this paper an OSPF domain stands for a network where at least one of the ruling routing protocols is the OSPF protocol.

¹³A connected area refers here to each area that have an "eavesdropping" box which communicates with the server.

3.2.1 Intra-Domain Topology

Network and router LSAs give information about the intra-domain link-state situation. With router LSAs a given routers collected states of its interfaces is presented. These link-states describe the routers connectivity in the area and can be of four different connection types: point-to-point to another router, to a transit or a stub network or to a virtual link. With each link-state additional data is given in form of link metrics, link specific data and ID, which further specifies the link target. Network LSAs describe all routers attached to a network (within the area). Network mask and the router ID of each of the attached routers give information of how to reach these networks.

The intra domain topology picture can be created by making router and network specific objects. Each created object is to be stored in a fast lookup structure and linked to its neighbors. A lookup structure suitable for this, with nice memory properties and management handling is the AVL tree explained in [Wirth].

3.2.2 Inter-Domain Topology

Routing information external to the area is given by summary LSAs and AS-external LSAs¹⁴. These two LSAs are important to define how areas within an AS are connected and the connectivity outside the AS.

With the help of these LSAs, described here and in the previous chapters, a structure can be made that reflects the area internal link-state situation and connectivity outside the area and AS. With this method an AS level database can be constructed which have connections between the ASs. Each AS represented have also area objects which are linked together. This AS and area object linkage require that the database can map IP addresses onto a particular AS¹⁵ and then onto a given area and ultimately onto the specific router. This functionality is important when performing admission control since it is imperative to be able to look up source and destination for a route request.

3.3 Link Bandwidth Over Time

To keep track of link loads over time in a domain, there are a number of issues that needs to be resolved. A database is needed that keeps information about the domain's link-states and capacities¹⁶. A separate data structure is also needed for each link. This second structure will keep track of the link loads and its changes over time.

Link-states can be gleaned from OSPF, as described in section 2.2, where the link cost is specified. This cost does not necessarily have to be a reliable source as to what real capacity the link has. It is in fact up to the one responsible for managing the router to set up the cost value on the interfaces. SNMP is a protocol used to download information or to trap changes of an network entity¹⁷. This protocol could be incorporated and used in the server to get the important information about a links bandwidth capacity.

Information about the link loads will come from the incoming (resource) reservation requests. These requests specify the time period for the data flow and the needed link capacity. How to address receiving, storing and granting of these request together with the usage of SNMP are¹⁸ outside the scope of this thesis and not further explained here.

3.3.1 A structure for link resources over time

For the purpose of keeping track of link bandwidth load over time a structure is intended that maintains link load over a decided time spectrum. Admission control for reservation requests may require the path algo-

¹⁴Explained in Section 2.2.3.

¹⁵In a future server version multiple ASs may be represented.

¹⁶Link bandwidth capacity.

¹⁷Which could be an interface to a router.

¹⁸See section 1.5.

rithm to make bandwidth checks for a given time period for each link that makes the path. These operations can be costly in terms of memory consumption and computational time. To guarantee an efficient way of performing admission control, a structure for fast lookup and with good memory properties is needed.

Different approaches in this area are suggested in research by Schelén in [Schelen] and by Turner in [Turn1] and [Turn2]. Even though the implementation and testing of a data structure of this kind is beyond the scope of this thesis it was imperative to explore them in order to have a good approach when designing the server. A related approach needs to be incorporated in future versions of the server.

Part II

Design and Implementation

4 Design

This chapter defines the design behind the server and its implemented prototype. All features here were not used in the implementation of the prototype but a subset of them ensured enough reliability to test the server's functionality.

An important requirement for the design was that it would have a modular approach. The server would function through modules. Each module was constructed to interact with other modules through a clean, generic interface. This modular propinquity helps to abstract the collaboration and facilitate future changes and updates. The generic interfaces help in abstracting and hiding implementation details which assists in the implementation process. The abstraction of the different parts of the server is made by six main modules which all have fundamentally different purposes.

Apart from these modules an additional extension to the GateD routing software package was designed and implemented as well. This was a step away from the requirement that small or no changes would be made to current routing software. However by using these changes the burden of implementing OSPF specifics was reduced significantly. The extended routing software is not to be used in all the network routers but only in the eavesdropping boxes.

4.1 Overview

A server that allows for advance reservations must be capable of building a database with topological information and link loads over time and also with the functionality to get this data from networks. To be fully operational it also need to handle processing, granting and enforcement of reservation requests. In Figure 3, the full system of such a server is described. Some parts were not implemented during this project but are still described since they explain the general workings of this system and for the server to be fully functional it is clear that they need to exist.

4.1.1 Module Interaction

How the modules communicate through the generic interfaces, and what the interface functions are, is described in [KHed3], a previously written design document. The general architecture and the actions of a fully operational server is pictured in Figure 3 and described as follows:

1. **Network - Server input Module (NSM)** communicates with its networks clients¹⁹ to get data input about network topology. The communication between clients and server is driven by changes in the network, periodic transmissions or server requests.
2. **Task Scheduling Module (TSM)** executes a processing job regarding incoming network data. The data is forwarded to the **Advance Management Module (AMM)**.
 - 3a. New or updated network information is used at AMM to re-construct or change the concerned parts of the topology database²⁰
 - 3b. Data is mapped from OSPF LSAs onto the **Advance Data Structure Module (ADM)** to be part of the topology database.
 - 4a. The **Admission Request Module (ARM)** receives a request from a client. It processes the request and determines after an admission control if the request can be accepted. Communication between client and server secures the decision.
 - 4b. The reservation's impact on link loads is inferred onto ADM. The reservation is stored.
5. The **Request Execution Module (REM)** is notified in due time of the reservation and steps are taken to honor the promised service.

¹⁹Here *clients* and *eavesdropping boxes* refer to the modified GateD running computers that supply the server with topology data.

²⁰Here database and data structure is used interchangeably.

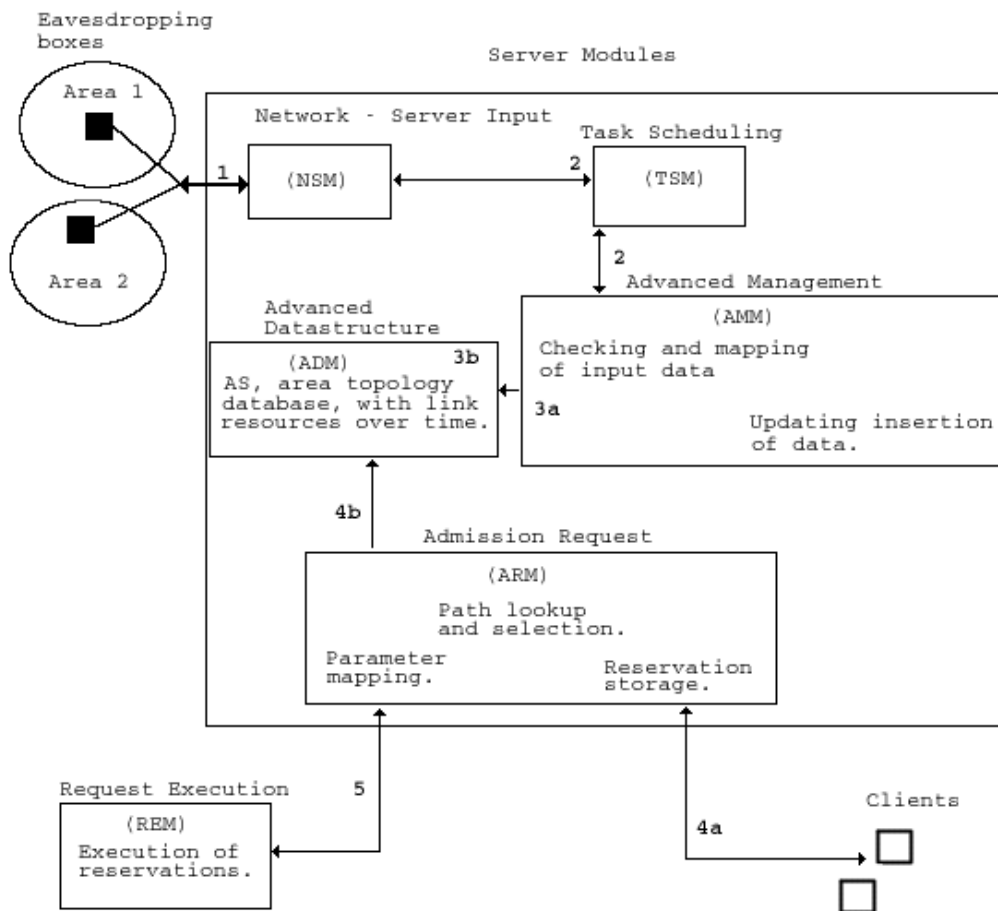


Figure 3: System Overview

Although communication between the modules happen through interfaces, not all function calls are made in this manner. If the processing to be done is substantial, the task can be inserted into a task queue. The execution of these tasks will be directed by the daemon's scheduler (TSM) to ensure fairness.

Some of the described actions here are, as mentioned earlier, beyond the reach of this thesis. The modules for admission control and reservation enforcement (ARM and REM) are glanced at in [KHed2] and [KHed3] and will not be further discussed here. Future work on the thesis will however to a high degree focus on those entities.

4.2 The Server Daemon

The actual server program is a daemon program. A daemon program is commonly a background process, that performs tasks automatically with little or no interaction from the responsible user. The daemon program gleans topology information from its clients, builds up a structure that mirrors that information and does all the necessary chores to maintain this structure.

4.2.1 Program Details

To manage the data structure buildup and management, several features are needed. Some of these features that affect the whole modular system is described below.

- Timers creation and management -continuous data aging and the future aspect of link loads demands that time is kept track of. Timers are used to ensure timeliness and aging of data. They are stored as objects in a time (priority) ordered queue and inspected as their time stamp expires.
- A job scheduling mechanism -where incoming data from network clients and important timers, that have expired, are taken care of first hand. Secondly, a job dispatcher executes jobs that are stored in a FIFO queue. These jobs are less prone to be of acute character and they involve longer operations such as path re-calculation and database maintenance operations.
- Parser functionality for reading a configuration script at startup or at re-initialization. With this method input parameters is read, regarding network clients and the daemon's running mode.

The initialization of other modules and the communication with network clients commence after the configuration reading and the environment setup. By allowing to read the configuration script during run time (at re-initialization) new input parameters can dictate the behavior of the server without the need to change and re-compile code.

- A shutdown and data recovery procedure ensures that stored data is handled properly at shutdown and re-initializations. At startups and initializations this procedure checks if previous shutdown was made in an orderly fashion. In cases of improper shutdown detection, data recovery is attempted from a previously stored backup.
- Of less important nature, for the functionality of the server, is a log function that records important events into a file. This log function has several priority modes and the user can set the mode at initialization to decide how much information it should track. This log function is important when measuring server efficiency and can also be used as a debug tool during implementation.

5 Server Modules

5.1 Overview

How the modules interact and their functionality is explained to some extent in the previous chapters. Some additional details about the Network - Server input Module, GateD extensions and the Task Scheduling Module will be given here. However, the main focus of this part is to explain the design, buildup and management of the topology data structure²¹.

5.2 Task Scheduling

The Task Scheduling Module is the main module of the server. It is responsible for the initialization and driving of the other modules. At startup the configuration script gives details about running modes and network client data. After environment set-up it forks off and continues as a background process i.e. as a daemon program.

The server daemon works within a single process. The approach not to fork off new processes, when dealing with maintenance tasks or incoming new connections, is because of time savings. A source of inspiration for this method is GateD which functions with a similar propinquity. Within the main loop of the program expiring timers and jobs are executed. As first priority scheduling comes incoming data from network clients. A non-blocking socket is used to communicate with network clients. NSM²² is signaled as writing to the socket is detected or reading of socket is done.

The jobs that are executed by the scheduler consists of job objects that keep data and a specific callback function that triggers an action within the same or another module. Examples of these jobs are expiring timers connected with some link-state, and incoming LSAs that are to be processed by the management module.

5.3 GateD extensions

The boxes²³ that provide the server with OSPF link-states can be ordinary PCs, with GateD/OSPF running and participating passively, i.e. not contributing to the OSPF protocol.

The extensions and changes that were necessary to do in GateD/OSPF deal mainly with the sending and receiving of link-state packages. By taking advantage of the error checking functionalities in GateD/OSPF the boxes contribute to the consistency checking of network data, something that is now somewhat relieved from the server. Incoming link-state packages, that may be of use to the server and that passed the GateD/OSPF error and consistency check, are forwarded with an UDP connection to the server. The server acknowledges the transmission, otherwise the package is transmitted again.

The simplicity of this scheme is by choice. These changes were made to feed the server prototype important network data without implementing major features in the server or in the boxes. In future versions a more secure and reliable two way communication is needed between boxes and the server.

5.4 Topology Data

ADM is the model responsible for topology data storage. This is done by the usage of five different kinds of structures or objects²⁴. It separates the incoming OSPF link-states by arranging them into AS, area and LS objects. Each AS object is representing the actual AS of the LSA. With the same methodology is each

²¹Advance Management Module (AMM) and the Advance Data structure Module (ADM).

²²Network - Server input Module

²³Mentioned in section 3.1.1

²⁴These objects represents AS and area networks but also network, router and external (although not yet implemented) link-states.

AS structured into areas and then further into LS objects. The latter can be of external, network or router type.

5.4.1 Link-State Objects

Each LS object is stored in an AVL tree specific for its kind. An area object contains thus trees for network, router and external LS objects. The decision of choosing AVL trees for this purpose comes from its well proven properties in regard to structure maintenance, look up speed and memory usage.

An overview of the area structure can be seen in Figure 4²⁵. It shows an area database with route tree, AVL trees for LS storage, LS objects and linkage between different structures and objects.

5.4.2 LS Object Specifics

A simplified view of a LS object is also shown in Figure 4. It has several common fields with the corresponding LSA as they represent the same information. This can also be seen if comparing the LS object with the LSA in appendix A.

Within each LS object a list with link information is kept. Each list item defines a link to another, neighboring object. A pointer to that object is stored within the same item together with its link specific data. This linkage is important when making admission control such as the running of the SPF algorithm when performing a path look up. An example of a similar approach is the link-state linkage in the OSPF database which normally²⁶ works in the same way.

5.5 Route Tree

An OSPF router keeps a database with itself as a center point when doing routing calculations, i.e. the SPF algorithm, and construction of the routing table. The server on the other hand must be able to make computations with an arbitrary starting point (LS object) as basis. This implies the necessity of a lookup function for each route and LS object. This look up functionality is for example used when identifying the source and destination in a path computation.

The solution is a route tree of AVL type, where each link in the area is represented. External link-state information can however contribute with links that points to different routers. This is possible because of the manner in which ABRs and ASBRs broadcast their routes and how they are incorporated in OSPF. The result is that two ABRs or ASBRs can both send link-state information regarding the same route, but with the possibility to go there through different routers. To avoid multiple nodes in the route tree when dealing with this particular case, each node of this type keep a list with the target objects' data. As with router and network route nodes each node here is identified by IP address and mask²⁷.

Since the route tree is used to identify the source or destination of a route, each route item has also a pointer to the corresponding LS object.

5.5.1 Path Calculation

To further describe the functionality of this module, the following scenario can be considered. A request was made to the server. It specifies the starting point and time, duration and any additional criterion that are important (bandwidth requirements etc). To calculate the path, both starting and end point of the transmission must be determined. A search in the route tree gives the matches with corresponding

²⁵Figure 4 shows how the data objects are related to each other. The figure gives a graphical explanation of the area object structure. The symbol for AVL tree is just a symbol and does not explain any functionality behind this kind of B-tree.

²⁶[OSPF] defines only the workings of the protocol, not the way it is implemented. The actual implementation is up to the manufacturer of the routing software, as long as it follows the standard format.

²⁷One definition of a network route can be given by IP address and a mask.

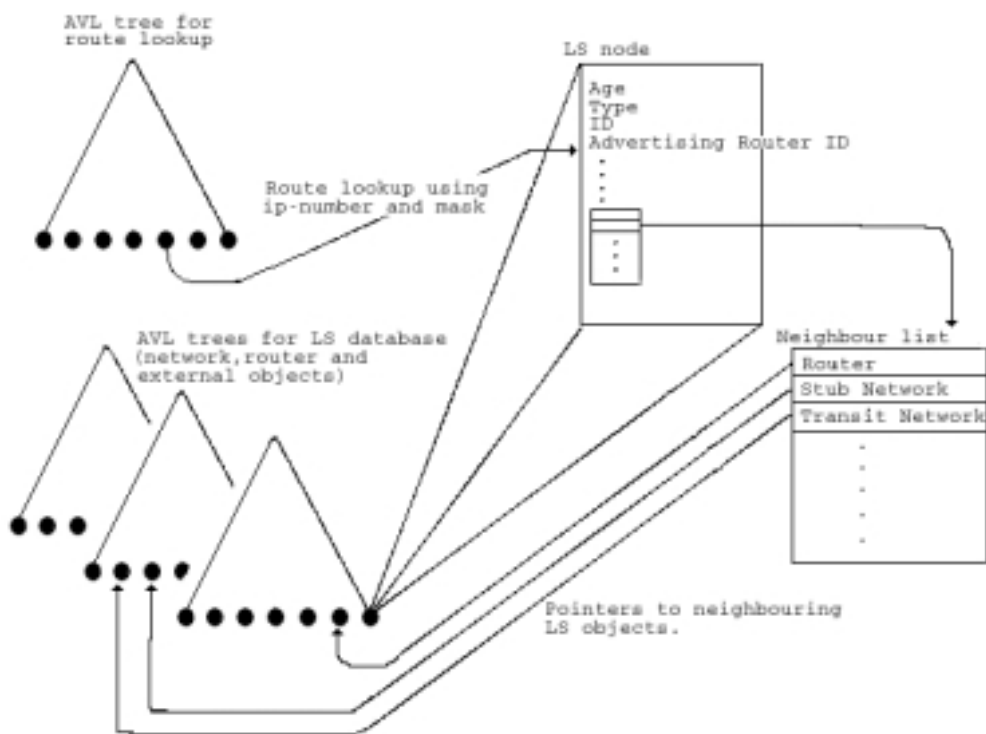


Figure 4: Area Data Structure

LS objects. These objects can now be used in the Dijkstra algorithm, together with additional admission control²⁸ algorithms, to see if a matching path that fulfills the requested criteria can be found. Since the LS objects have already been linked with their neighbors, this algorithm will not have to make additional look ups to find them.

²⁸ Admission control is about investigating if available resources can be used for a request with given demands.

6 Topology Management

The purpose of the module for database maintenance is to use incoming network information, as of yet only LSAs, to construct and maintain the topology database as an accurate model of the involved networks. The difference between AMM and ADM is that the former works upon the latter's data structures. ADM on its part have structures for storage of network data that are also used for admission control purposes.

As only LSAs are currently used in the implemented prototype, no focus has yet been made on the protocols and data necessary to build the structure²⁹ for tracking of future link loads.

6.1 LSA Insertion

Incoming LSAs are identified by type, area, IP number and network mask. After identification a LSA specific lookup is made to check the database for the need for insertion or updating of current data. This lookup process must first go through the right AS, then find the matching area and finally get to the corresponding LS object.

Several area objects are present within each AS object. The LSA area number makes it possible to make the match. A lookup to find the corresponding LS object, of router, network or external type, is made by the advertising router's ID and type of LSA. When a match is found the incoming data is checked towards the object to see if insertion or updating is necessary. If no match is made then a new object must be created for the incoming data.

6.1.1 Data Mapping

At insertion, the OSPF specific link-state data is mapped to formats more suitable for the server. This involves a mapping from network to system specific format and onto the particular LS object. Here the data is further used to construct linkage between other, neighboring, objects.

6.2 Maintenance

Maintenance of the LS objects' databases and the overall structure is another important task to ensure data aging, updates, deletion, insertion and database consistency checking.

The aging of LS data is done by creating a timer object that uses a callback function when its time expires. This usually triggers lookup and checking of the particular LS object. Aging of data can lead to a deletion process, which can be a relatively time consuming operation since many different entities (neighboring objects, external linkage, route lookup tree etc) may be affected. Timer objects are handled by the scheduler.

²⁹Section 3.3.

7 Implementation

This part explains more about the actual prototype implementation.

7.1 GateD Extensions

The approach to re-use the OSPF error checking functionality and to get the same input LSAs as the router was in fact a trivial solution.

By using an UDP, unreliable, connection between the network client and the server, no re-packaging of the LSAs is needed and the LSAs can be sent as they are received. The actual sending of the LSAs happen after the OSPF error checking³⁰. This ensures that the packets have been authenticated, checked for duplicity, data corruption etc. Since the server uses this security approach it is only assured that the packages that are received were approved when they were received at the router and not that they have not been corrupted during the second transfer. The important aspect is however that the server can use the error checking power of the OSPF protocol without actually implementing it.

This scheme is deemed enough for the prototype implementation since a complete error checking would aggravate the implementation burden. Future versions should however have a way to ensure the reliability of the LSA transmission between routers and server.

7.1.1 Reason for Server - Router Communication Scheme

The actual commence of the communication is a probing³¹ from the server to the routers. By doing it this way and not the other way around (as suggested in the [KHed1] proposal) we only need to make changes to one entity if need be (i.e. the server's configuration script). Another benefit from this is that if there is a need for a backup server to go on-line (at server failure), it is easier for the backup server to re-initiate contact and communication with the routers than the other way around.

The future goal here is to make the server the driving force in the communication process, by giving it the power of requesting new data as need for that arises.

7.2 Prototype Implementation

Testing of the server design was made with a server prototype. This prototype included a subset of the designed functionalities.

Some features of the server were only implemented to a limited extent as the primary concern was on the data structure and not on functionalities that will be of more importance in some future version. These, not fully implemented features, include: data rescue, timer and configuration script usage.

The construction of structures within the ADM was done with the limit of only one active area. This lead to an abstraction from the usage of multiple ASs and areas. External link-state data was hence of no use in the prototype and the prototype design considered only network and router LSAs to contribute to the topology picture.

7.2.1 Proof of Operation

The prototype consists of several separate modules. One important phase in the implementation was module testing. It was possible to facilitate the testing of the full system by first testing each module separately before connecting it with the other modules.

³⁰See section 3.1.2.

³¹The act of checking connectivity to a network entity. This is made with a non-blocking TCP/IP call.

7.2.2 Network Interface

The parts³² that are responsible for interpreting OSPF topology data and communicate with network entities have to operate without interruptions and delays. It is vital that these parts are fast and efficient since arriving data can come in short, intense, periods.

Two computers were set up with the extended GateD software in a preliminary test bed. Incoming dummy data was sent continuously, or in accordance with a made pattern³³. Confidence in that arriving data was unchanged from its original was gained by storing³⁴ and later comparing the two.

OSPF packets that were used in testing consisted of the Hello packet and router LSAs. The latter was only used in module testing and not for the full system. These router LSAs were also not sent from the routers in the test bed but were constructed artificially.

7.2.3 Topology Information and Database Buildup

Topology Information and database was constructed and module tested by using artificially constructed LSAs that were fed to the topology modules³⁵ and thus inserted into the database. Data consistency was checked by comparing the decided input with the modules' output.

One goal for testing was to use a modified version of the Dijkstra algorithm to find the path between two given routers. This scenario is explained in section 5.5.1. The construction and testing of the prototype stopped with AMM and ADM. The final goal of using the Dijkstra algorithm to further test these modules was however not reached and will instead be executed in this work's continuation.

³²NSM, ADM and to some extent TSM.

³³Interruptions were possible by simply turning off and on the extension that was responsible for forwarding LSAs and Hello packets.

³⁴This was made with the log function (see section 4.2.1)

³⁵I.e. ADM and AMM.

Part III

Conclusions and Future Work

8 Future Work

Since the implemented prototype only inherited a subset of all the functionalities that should belong to the server a natural continuation of this work is to integrate them all in the server.

The usage of timers, configuration script and the communication with eavesdropping boxes are all features that need to be further improved, especially as the responsibility of the prototype will grow from compatible with one area to multiple areas and even ASs.

Usage of the topology database, the future time aspect, admission control algorithms and also to receive and execute reservation requests are all important when it comes to implement a fully functional server. These aspects are the primary targets for future improvements.

9 Conclusions

A thorough investigation of how to build a system for advance reservation was made. The result lead to a design and an implementation regarding usage of OSPF link-states to buildup of a topology picture. The design and implementation was made with the purpose of being modular and easy to extend with new functionality.

The prototype was only implemented with a subset of the designed features. One goal that was not completely accomplished was the topology database construction.

The implemented subset of designed features was enough to show that the server will be able to buildup a topology picture that reflects the situation in an area. A full testing of the data structures, with path lookup algorithms, was not performed. Testing of the individual modules as well as the prototype as a whole shows however that the implemented prototype can be an important starting ground for the building of a complete server and architecture for advance reservations.

10 Work Evaluation

This work has undergone a few modifications during its progress. At first it was supposed to be a project for two persons, although it had to be changed later as the project had a manpower fallout. Furthermore, as I got more knowledge about this area and discussed the work with the supervisor at the University of Pennsylvania it became clear that the focus should change from making OSPF changes to designing and constructing a server.

The main goal, to investigate and design a server for advance reservations, was achieved. The prototype that was implemented was however not equipped with all the designed features. The database, responsible for keeping topology information, was implemented with limited functionality and was not fully tested. The time consuming design and information search phases, took in retrospective, too long time.

The secondary goals³⁶ were also achieved. The design was made with a modular approach, with clean interfaces between the modules and interaction with OSPF was also achieved although with some minor changes to the GateD software³⁷.

As this work is supposed to perpetuate, and is in fact only a starting ground for future research about advance reservations and algorithm analysis, an effort was made to facilitate the work's continuation by thoroughly documenting it.

This chapter describes first the original plan for the work, why and how it changed. Then follows a description of what steps that were executed to realize the thesis.

10.1 Performed Tasks

About every second week a progress report was sent to the thesis examiner in Sweden who gave feedback and views about the work. Meetings with a supervising professor at the University of Pennsylvania were held at a periodical basis or as need arose for them. As an outcome of these meetings and reports, three design documents were written that described the focus of the work and what was currently being implemented.

10.1.1 Installation

As the work progressed beyond literary study and design decisions a workbench was set up consisting of two PCs with Linux and Windows NT as running OS with GateD as routing software and OSPF as the active protocol.

10.1.2 Literature Study

An extensive study of literature was the starting ground for later design decisions and implementation. RFCs related to OSPF in particular and advance reservations in general, admission control algorithms and different B-trees were all investigated to grasp this complex area.

A valid conclusion about this in retrospective is that although necessary this step took up too much valuable time that could have been used for designing and implementing the server.

10.1.3 Software and Code Studies

The routing software that was installed, explored and later used was [GateD]. A study of the code behind the GateD program was made to get a deeper knowledge about the handling of routing protocols and to fully understand how to affect the original software as little as possible.

³⁶The secondary goals was to use a modular design and to interact, while not disturbing, with current routing protocols (see section 1.2.

³⁷These changes can be avoided with a more thorough server implementation.

Two versions of OSPF routing protocols were also examined; an enhanced one which was used in previous QoS³⁸ research and another one which was the standard OSPF³⁹ used in the free version of GateD. The enhanced version of OSPF contributed to deeper knowledge of how to use the routing data but was later discarded as the thesis proposal was modified.

10.1.4 Design and Thesis Documentation

Design proposals and documents were made on a continuous basis as the work progressed. It was an important source for avoiding misunderstandings and mistakes. The actual thesis writing started at a late stage during the project even though it benefited greatly from the three previously written design documents.

10.1.5 Server Implementation

The server prototype started as a C implementation but was quickly changed into the object oriented C++ language as the ease for making a modular structure benefited from this. The changes in GateD/OSPF were all made in C.

³⁸In research described in [QoS/OSPF].

³⁹OSPF version 2.

Glossary

- **ABR** Area Border Router.
- **ADM** Advance Data structure Module.
- **AMM** Advance Management Module.
- **AS** Autonomous System. A group of routers that are all under one independent administrative domain.
- **ASBR** AS Boundary routers.
- **Backbone** is area 0 in a domain of multiple areas. It is the center of communication between the areas.
- **BGP** Border Gateway Protocol, an Exterior Gateway Protocol.
- **client** refers to the modified GateD running computers that supply the server with topology data.
- **Database** a systematically arranged collection of computer data, structured so that it can be automatically retrieved or manipulated.
- **Dijkstra Algorithm** is a graph algorithm that solves the problem of finding the shortest path between a given source and all other destinations in a graph.
- **eavesdropping boxes** refers to the modified GateD running computers that supply the server with topology data.
- **FIFO** First In First Out is a principle behind the workings of a common list or queue.
- **GateD** The Gate Daemon is a routing software package that includes common routing software.
- **IP** Internet Protocol. The network layer for the TCP/IP protocol suite.
- **IGP** Interior Gateway Protocol. An Internet protocol which distributes routing information to the routers within an autonomous system.
- **EGP** Exterior Gateway Protocol. A protocol which distributes routing information to the routers which connect autonomous systems.
- **Linux** A UNIX-like but free operative system.
- **link-state database** a database that mirrors the relation between networks and routers in an OSPF domain.
- **LSA** Link State Advertisement.
- **Network Mask** Also called *netmask*, a 32-bit bit mask which shows how an Internet address is to be divided into network, subnet and host parts.
- **NSM** Network - Server input Module
- **OSPF** Open Shortest Path First. A routing protocol that allows routers to exchange information regarding costs and availability for reaching other networks and routers. It is a link state network routing protocol that is common in routers of today.
- **QoS** Quality of Service. The performance properties of a network service, possibly including throughput, transit delay, priority.
- **RFC** Request For Comments

- **RIP** Routing Information Protocol, an Exterior Gateway Protocol.
- **server** refers here to a program that collects, store and work with topology data.
- **SNMP** Simple Network Management Protocol.) The Internet standard protocol developed to manage nodes on an IP network. SNMP is not limited to TCP/IP. It can be used to manage and monitor all sorts of equipment including computers, routers, wiring hubs, toasters and jukeboxes.
- **SPF** Shortest Path First.
- **TCP** Transmission Control Protocol. The most common transport layer protocol used on Ethernet and the Internet. TCP is built on top of Internet Protocol (IP) and is nearly always seen in the combination TCP/IP (TCP over IP).

References

- [GateD] Merit GateD Consortium.
URL:<http://www.gated.org>
- [QoS Page] *QoS Extensions to OSPF and Related Works*
URL: http://www.seas.upenn.edu/~guer/qos_ospf.html
- [IETF] IETF, The Internet Engineering Task Force
URL:<http://www.ietf.org/>
- [Roch1] Apostolopoulos G, Guerin R and Kamat S. *Design and Implementation of QoS Routing Extensions to GateD OSPF with Interface to RSVP*.
- [Roch2] Guerin R and Orda A. *Networks With Advance Reservations: The Routing Perspective*. Proceedings of INFOCOM'2000, March 2000.
- [Roch3] Apostolopoulos G, Guerin R, Kamat S and Tripathi S. *Quality of Service Based Routing: A Performance Perspective*. Proceedings of SIGCOMM'98, Vancouver, British Columbia, September 1998. Computer Communication Review, Volume 28, Number 4, October 1998.
- [Roch4] Apostolopoulos G, Guerin R and Kamat S. *Implementation and Performance Measurements of QoS Routing Extensions to OSPF*. Proceedings of INFOCOM'99, New York, NY, March 1999.
- [Roch5] Guerin R, Kamat S and S. Herzog. *QoS Path Management with RSVP*. Proceedings of 2nd Global Internet Mini conference, Phoenix, AZ, November 1997.
- [Qbone] QBone
URL:<http://qbone.internet2.edu>
- [QboneBB] Teitelbaum B and Chimiento P. *Qbone Bandwidth Broker Architecture*. Work in progress.
URL: <http://qbone.internet2.edu/bb/bboutline2.html>
- [Wisch] Wischik D and Greenberg A. *Admission control for booking ahead shared resources*. IEEE Infocom. 1998.
- [QoS-Routing] Crawley E, Nair R, Rajagopalan B and Sandick H. *A Framework for QoS-based Routing in the Internet*. Request for Comments RFC 2386, Internet Engineering Task Force, August 1998.
- [ADiff] Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W. *An Architecture for Differentiated Services*. Request for Comments RFC 2475, Internet Engineering Task Force, December 1998
- [FAdm] Yavatkar R, Pendarakis D and Guerin R. *A Framework for Policy-based Admission Control*. Request for Comments RFC 2753, Internet Engineering Task Force, January 2000.
- [OSPF] Moy J. *OSPF Version 2*. Request for Comments RFC 2328, Internet Engineering Task Force, April 1988.
- [CISCO] *OSPF Design Guide*. URL:<http://www.cisco.com/warp/public/104/1.html>
- [EGP] Rosen EC, Baranek B. *EXTERIOR GATEWAY PROTOCOL (EGP)*. Request for Comments RFC 827, Internet Engineering Task Force, October 1982.
- [RSVP] Herzog S. *Resource ReSerVation Protocol (RSVP)*. Request for Comments RFC 2750, Internet Engineering Task Force, January 2000.
- [RSVP/OSPF] Apostolopoulos G, Kamat S and Guerin R. *Design and Implementations of QoS Routing Extensions and RSVP Interface*, Design Document at URL: http://www.seas.upenn.edu/~guer/publications/qos_ospf_design_doc.ps.gz

- [QoS/OSPF] Apostolopoulos G, Williams D, Kamat S, Guerin R, Orda A, Przygienda T. *QoS Routing Mechanisms and OSPF Extensions*. Request for Comments RFC 2676, Internet Engineering Task Force, August 1999.
- [KHed3] Hedström K. *Design Document Over a Server Based Architecture for Advance Reservations in a Link-State Domain*. Design document (3. ed). November 2000. URL: <http://www.seas.upenn.edu/~kjell/files/design3.ps.Z>
- [KHed2] Hedström K. *Design Document Over a Server Based Architecture for Advance Reservations in a Link-State Domain*. Design document (2. ed). October 2000. URL: <http://www.seas.upenn.edu/~kjell/files/design2.ps.Z>
- [KHed1] Hedström K. *Design Document Over a Server Based Architecture for Advance Reservations in a Link-State Domain*. Design document (1. ed). October 2000. URL: <http://www.seas.upenn.edu/~kjell/files/design1.ps.Z>
- [Turn1] Turner J S. *Terabit Burst Switching*. Journal of High Speed Networks, 1999.
- [Turn2] Turner J S. *WDM Burst Switching*. Proceedings of INET 99, 1999.
- [Schelen] Schelén O. *QoS Agents in the Internet*, Ph.D thesis, Luleå University of Technology, Sweden, 1998.
- [Schel1] Schelén O, Nilsson A, Norrgård J and Pink S. *Performance of QoS Agents for Provisioning Network Resources*. In Proceedings of IFIP Seventh International Workshop on Quality of Service (IWQoS'99), London, UK, June 1999.
- [Schel2] Schelén O and Pink S. *Resource Sharing in Advance Reservation Agents*. In Journal of High Speed Networks, Special issue on Multimedia Networking, Vol 7, No 3-4, 1998.
- [Schel3] Schelén O and Pink S. *An Agent-based Architecture for Advance Reservations*. Proceedings of IEEE 22nd Annual Conference on Computer Networks, Minneapolis, Nov 1997.
- [Schel4] Schelén O and Pink S. *Sharing Resources through Advance Reservation Agents*. Proceedings of IFIP Fifth International Workshop on Quality of Service, New York, May 1997.
- [Schel5] Schelén O and Pink S. *Aggregating Resource Reservations over Multiple Routing Domains*. Proceedings of IFIP Sixth International Workshop on Quality of Service, Napa Valley, California, May 1998.
- [Wolf] Wolf L, Delgrossi L, Steinmetz R, Schaller S and Wittig H. *Issues of Reserving Resources in Advance*. Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video, New Hampshire, April 1995. URL:<http://hulk.bu.edu/nosdav95/papers/wolf.ps>
- [Bers] Berson S, Lindell R and Braden R (1998) *An Architecture for advance reservations in the Internet*. Technical report. URL:<http://www.isi.edu/~berson/advance.ps>
- [Fost] Foster I, Kesselman C, Lee C, Lindell R, Nahrstedt K, Roy A (1999) *A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation*. Intl Workshop on Quality of Service. URL:<ftp://ftp.globus.org/pub/globus/papers/gass.ps.gz>
- [Moy] Moy J T (2000). *OSPF Complete Implementation*. Addison-Wesley. ISBN:0-201-30966-1.
- [Weiss] Weiss MA (1999) *Data structures & algorithm analysis in C++*. (2. ed) Addison Wesley. ISBN: 0201361221.
- [Wirth] Wirth N (1976). *Algorithms + Data Structures = Programs*. Englewood Cliffs, New Jersey: Prentice-Hall. ISBN: 0-13-022418-9.

- [CHor] Horstmann C.S (1991). *MASTERING C++*. *An Introduction to C++ and Object-oriented Programming for C and Pascal Programmers*. Prentice-Hall. ISBN: 0-471-52257-0.
- [Kern] Kernighan BW, Ritchie DM (1988). *The C Programming Language*. (2. ed) Englewood Cliffs, New Jersey. Prentice-Hall. ISBN: 0-13-110370-9.
- [Stev] Stevens RW (1988). *UNIX NETWORK PROGRAMMING*. *Networking APIs: Sockets and XTI*. (2. ed, 1. Vol) Upper Saddle River. Prentice-Hall. ISBN: 0-13-490012-X.
- [BStro] Stroustrup B, Ellis MA (1991). *The Annotated C++ Reference Manual*. Murray Hill, New Jersey. Addison-Wesley. ISBN: 0-201-51459-1.
- [Pohl] Pohl I, Kelley A (1996). *C by Dissection*. (3. ed) Addison-Wesley. ISBN: 0-8053-3149-2.
- [Lamp] Lempert L (1994). *L^AT_EX USER'S GUIDE AND REFERENCE MANUAL*. (2. ed) Addison Wesley. ISBN:0201529831.

Appendix

A: Router-LSAs

A Router-LSAs

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
LS age										Options										1																			
Link State ID																																							
Advertising Router																																							
LS															sequence number																								
LS checksum															length																								
0	V E B									0										# links																			
Link ID																																							
Link Data																																							
Type										# TOS										metric																			
...																																							
TOS										0										TOS metric																			
Link ID																																							
Link Data																																							
...																																							