NOTE TO USERS

This reproduction is the best copy available.

UMI[®]

Scalable Network Control

By

Mohamed EL-Darieby

A thesis submitted to the Faculty of Graduate Studies and Research in
partial fulfillment of the requirements of the degree of Doctor of Philosophy

Ottawa Carleton Institute for Electrical and Computer Engineering

Faculty of Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada. K1S 5B6

May 7th, 2004

© Copyright 2004, Mohamed EL-Darieby

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-94194-9

Our file Notre référence

ISBN: 0-612-94194-9

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

ABSTRACT

The Internet is evolving into a multi-service infrastructure for the telecommunications and computer industries. This infrastructure is required to provide *network services* that better support Quality of Service guarantees regarding performance and reliability. The goal of this thesis is to develop protocols, referred to as *h*-protocols, to better support the provisioning of such services through the efficient computation of end-to-end service routes and efficient fault notification. Standard hierarchical network frameworks were not designed to achieve this. The effectiveness of the *h*-protocols is studied and compared to existing techniques by analysis and simulation.

The *h*-protocols avoid highly utilized network domains by load-balancing data traffic workloads among network resources. The protocols provide routing scalability as network size grows. For a system under study, simulation experiments have shown that only a small increase in service routing computational and communication overheads are required to create same-length paths as the network size grows from 256 to 4096 nodes. The depth of network hierarchy affects protocol performance. A deeper hierarchy ensures smaller service creation time but increases the communication overhead. Hierarchical aggregation of fault notification signals results in a decrease in the number of fault notification messages, i.e., the size of the signaling storm. With this approach, the size of the signaling storm is a function of the number of levels in the hierarchy. With other fault notification protocols the size of the signaling storm is a function of network size. In general, the advantages of the *h*-protocols come at the cost of more overhead when compared to standard hierarchical network frameworks. This suggests that the *h*-protocols are most appropriate for provisioning higher bandwidth services.

DEDICATION

To my Family

ACKNOWLEDGMENTS

Million thanks to

Dorina C. Petriu, Jerome A. Rolia,

Diwakar Krishnamurthy,

and my family

Table of Contents

CHAPTER 1: INTRODUCTION.....	1
1.1. PROTOCOLS FOR NETWORK CONTROL	1
1.2. PROBLEM DEFINITION.....	3
<i>1.2.1. Routing in Large Networks</i>	3
<i>1.2.2. Fault Notification Scalability</i>	4
1.3. THESIS SCOPE AND METHODOLOGY	5
<i>1.3.1. Thesis Scope</i>	5
<i>1.3.2. Methodology</i>	6
1.4. THESIS CONTRIBUTIONS AND PUBLICATIONS.....	7
<i>1.4.1. Dissertation Overview</i>	11
CHAPTER 2: RELATED WORK.....	12
2.1. INTRODUCTION.....	12
2.2. CONNECTION-ORIENTED TRAFFIC ENGINEERING.....	13
<i>2.2.1. Routing</i>	13
<i>2.2.2. Examples of Signaling Protocols</i>	17
<i>2.2.3. Bandwidth Brokers</i>	20
2.3. CONNECTIONLESS TRAFFIC ENGINEERING	22
<i>2.3.1 A Summary on Traffic Engineering Models</i>	23
2.4. SERVICE RECOVERY	24
<i>2.4.1. Fault Notification</i>	26
2.5. CONCLUDING REMARKS.....	27
CHAPTER 3: A MODEL FOR MPLS SERVICE CREATION.....	28
3.1. INTRODUCTION.....	28
3.2. MODELING MPLS SERVICE CREATION.....	28
<i>3.2.1. Layered Queuing Modeling</i>	28
<i>3.2.2. Modeling Service Creation Operations</i>	30
<i>3.2.3. Hierarchical Tunnels</i>	34
3.3. EXPERIMENT DESIGN	36
<i>3.2.3. Performance results</i>	38
3.4. CONCLUDING REMARKS.....	41
CHAPTER 4: SCALABLE SERVICE ROUTING.....	44

4.1. INTRODUCTION.....	44
4.2. THE <i>H</i> -PROTOCOLS.....	45
4.2.1. <i>The H-DRA Protocol</i>	47
4.2.2. <i>The HDR Protocol</i>	53
4.2.3. <i>The HDR-P Protocol</i>	56
4.3. EVALUATION OF THE PERFORMANCE OF THE <i>H</i> -PROTOCOLS.....	58
4.4. EVALUATING THE PERFORMANCE AND SCALABILITY OF THE <i>H</i> -PROTOCOLS.....	62
4.4.1. <i>H-DRA Performance</i>	65
4.4.2. <i>PNNI Path Setup Time</i>	70
4.4.3. <i>HDR Performance</i>	72
4.4.4. <i>HDR-P Performance</i>	76
4.4.5. <i>A Comparison of the Performance of the H-protocols and PNNI</i>	79
4.4.6. <i>Path Computation Load</i>	82
4.4.7. <i>Traffic Engineering and Workload Distribution</i>	84
4.4.8. <i>Summary of Results</i>	89
4.5. CONCLUDING REMARKS.....	91
CHAPTER 5: SCALABLE FAULT NOTIFICATION.....	93
5.1. INTRODUCTION.....	93
5.2. HIERARCHICAL FAULT NOTIFICATION.....	94
5.2.1. <i>Unicast-SFN</i>	94
5.2.2. <i>Enhanced Unicast-SFN</i>	97
5.2.3. <i>Multicast-SFN</i>	99
5.3. EVALUATION OF THE PERFORMANCE OF SFN.....	103
5.3.1. <i>Experimental Design</i>	107
5.3.2. <i>SFN Performance</i>	108
5.4. CONCLUDING REMARKS.....	113
CHAPTER 6: CONCLUSIONS AND FUTURE WORK.....	115
6.1. CONCLUSIONS.....	115
6.2. DISCUSSION OF THE OVERHEAD OF <i>H</i> -PROTOCOLS.....	118
6.3. FUTURE WORK.....	119
APPENDIX A: LIST OF SYMBOLS	121
APPENDIX B: LIST OF ACRONYMS	122
REFERENCES	124

List of Figures

Figure 1.1: GMPLS as a unified control plane	3
Figure 1.2: Thesis scope	5
Figure 1.3: Thesis methodology and outline	7
Figure 2.1: Outline of the "Related Work" chapter	12
Figure 2.2: A hierarchical network.....	16
Figure 3.1: UML scenario diagram for MPLS service creation along with service times.....	32
Figure 3.2: Software architecture of MPLS-based service creation	33
Figure 3.3: LQM representation of hierarchical paths	36
Figure 3.4: Estimated response times for the no aggregation area case.....	40
Figure 3.5: Estimated response times for the 4 aggregation areas case.....	40
Figure 3.6: Estimated response times for the one aggregation area case.....	40
Figure 3.7: CPU utilization of edge router (one CPU per node).....	41
Figure 4.1 (a): Finding " <i>RootBB(src, dst)</i> "	49
Figure 4.1 (b): H-DRA signaling at level 3.....	50
Figure 4.1 (c): H-DRA signaling at level 2.....	50
Figure 4.2: The H-DRA protocol.....	52
Figure 4.3: HDR signaling	54
Figure 4.4: The HDR protocol.....	55
Figure 4.5: HDR-P signaling	56
Figure 4.6: The HDR-P protocol.....	57
Figure 4.7: Average number of H-DRA messages for 3-level network hierarchies.....	67
Figure 4.8: Average number of H-DRA messages for 5-level network hierarchies.....	67
Figure 4.9: Average H-DRA path setup time for a 3-level network hierarchies.....	69
Figure 4.10: Average H-DRA path setup time for 5-level network hierarchies	69
Figure 4.11: Average PNNI path setup time for 3-level network hierarchies	71
Figure 4.12: Average PNNI path setup time for 5-level network hierarchies	71
Figure 4.13: Average number of HDR messages for 3-level network hierarchies.....	73
Figure 4.14: Average number of HDR messages for 5-level network hierarchies.....	73
Figure 4.15: Average HDR path setup time for 3-level network hierarchies.....	75
Figure 4.16: Average HDR path setup time for 5-level network hierarchies.....	76
Figure 4.17: Average HDR-P path setup time for 3-level network hierarchies.....	78
Figure 4.18: Average HDR-P path setup time for 5-level network hierarchies.....	78
Figure 4.19: Message complexity of the h-protocols and PNNI to create a 20-node path	80
Figure 4.20: Message complexity of the h-protocols and PNNI to create a 20-node path	80

Figure 4.21: Path setup times of the h-protocols and PNNI to create a 20-node path.....	81
Figure 4.22: Path setup times of the h-protocols and PNNI to create a 60-node path.....	82
Figure 4.23: Average number of route computations/BB/100 request for 3-level hierarchies	83
Figure 4.24: Average number of route computations/BB/100 request for 5-level hierarchies	83
Figure 4.25: Average cumulative distribution of the inter-domain links for H_{3-16}	85
Figure 4.26: Average cumulative distribution of the inter-domain links for H_{5-4}	85
Figure 4.27: Average cumulative distribution of the inter-domain links for H_{3-36}	86
Figure 4.28: Average cumulative distribution of the inter-domain links for H_{5-6}	86
Figure 4.29: Average cumulative distribution of the inter-domain links for H_{3-64}	87
Figure 4.30: Average cumulative distribution of the inter-domain links for H_{5-8}	87
Figure 4.31: Average admission percentages for all hierarchies for H-DRA, HDR and PNNI.....	89
Figure 5.1: The unicast-SFN algorithm.....	95
Figure 5.2: Unicast-SFN signaling	96
Figure 5.3: The enhanced SFN algorithm.....	97
Figure 5.4: Enhanced SFN signaling.....	98
Figure 5.5: The multicast-SFN algorithm	101
Figure 5.6: Multicast-SFN signaling example	102
Figure 5.7: Average size of <i>standard unicast</i> signaling storm.....	110
Figure 5.8: Average size of unicast-SFN signaling storm for 3-level network hierarchies	110
Figure 5.9: Average size of unicast-SFN signaling storm for 5-level network hierarchies	110
Figure 5.10: Average size of multicast-SFN signaling storm for 3-level network hierarchies.....	111
Figure 5.11: Average size of multicast-SFN signaling storm for 5-level network hierarchies.....	111
Figure 5.12: Average number of BB managing a path for 3-level network hierarchies.....	113
Figure 5.13 Average number of BB managing a path for 5-level network hierarchies.....	113

List of Tables

Table 4.1: Performance of the h -protocols and PNNI.....	61
Table 4.2: Dimensions for network hierarchies studied	64
Table 4.3: H-DRA communication complexity for paths of 20 and 60 nodes	68
Table 4.4: H-DRA setup times for paths of 20 and 60 nodes (in ms).....	70
Table 4.5: PNNI setup times for paths of 20 and 60 nodes (in ms).....	72
Table 4.6: HDR/ HDR-P communication complexity.....	73
Table 4.7: HDR setup times for paths of 20 and 60 nodes (in ms).....	76
Table 4.8: HDR-P setup times for paths of 20 and 60 nodes (in ms).....	78
Table 5.1: Fault notification protocols studied in this chapter.....	103
Table 5.2: Approximate sizes of signaling storm for the protocols studies	106
Table 5.3: Average sizes of signaling storm a 256-node network for the protocols studied.....	111
Table 5.4: Average sizes of signaling storm a 4096-node network for the protocols studied	111

Chapter 1: Introduction

1.1. Protocols for Network Control

The Internet is evolving into a multi-service infrastructure for the telecommunications and computer industries. This infrastructure is required to provide sophisticated *network services* that support Quality of Service (QoS) guarantees regarding performance and reliability. Such network services support new network applications such as IP telephony, media-on-demand, conferencing [1], virtual private networks [60] and Grid and Utility Computing [97]. The goal of this thesis is to develop protocols to better support the provisioning of such network services.

Network service provisioning can be decomposed into three main tasks, namely: network control, service management and service execution. Network control allocates network resources to network services [37]. It deploys Internet Traffic Engineering (TE) [15] techniques in order to guarantee QoS and optimize network cost and performance. Network control also deploys service recovery mechanisms to recover from network failures. Network resources of particular interest include link bandwidth, buffer space and computational resources. Designing a scalable control plane has been identified as major thrust for re-engineering the existing Internet beyond the simple "best-effort" model [88]. This thesis describes a number of *network control* protocols for provisioning network services.

Service management maintains QoS levels as required by network applications. This involves Fault, Configuration, Accounting, Performance and Security (FCAPS) management.

Service execution transports data across the network infrastructure. This involves classifying, marking and forwarding of data packets.

A network service, described hereinafter as *service*, is a network connection that supports QoS between two network end points. A service may traverse a number of network domains because the global Internet is decomposed into network domains for reasons of scale and security. A network domain is a set of interconnected network nodes. Each network domain is controlled by a service provider, which may control, in turn, more than one domain.

Service providers require network domains to be autonomous. Network domain autonomy hides the internal information of a network domain. Domain autonomy enables the deployment of different network control technologies such as Asynchronous Transfer Mode (ATM) [3], Frame Relay (FR) [3] and Wavelength Division Multiplexing (WDM) [24]. The internetworking of these control technologies enables data transfer between network domains.

The Generalized Multi-Protocol Label Switching (GMPLS) framework [73] offers a unified control plane for various network domain control technologies. Figure 1.1 illustrates GMPLS as a unified control plane for FR, ATM and WDM. Within a network domain, GMPLS uses TE techniques to provide QoS guarantees [15], and to enhance service reliability [9].

TE enables the provisioning of services with QoS guarantees and recovery for services in the presence of network failures [16]. In general, TE follows two models, namely the connection-oriented model and the connectionless model. The connection-oriented model of TE [15] [16] uses routing algorithms to avoid network congestions and service recovery techniques to restore services in order to improve service availability. The connectionless TE model [8] optimizes weights of network links in order to balance load over these links and avoid congestion. Current TE mechanisms lack scalable routing and fault notification protocols for large networks with multiple administrative domains. These problems are described in detail in Section 1.2.

Figure 1.1: GMPLS as a unified control plane

This thesis proposes network protocols for efficient computation of end-to-end service routes and for efficient fault notification in such networks. These protocols are described in the context of GMPLS, because GMPLS was chosen by IETF common control and measurement plane Working Group as a standard protocol for network control.

This chapter is organized as follows. Section 1.2 motivates this work and defines the problems that are addressed. Section 1.3 lists the contributions of this thesis. Section 1.4 defines the scope and the methodology followed and provides a roadmap for the remaining chapters.

1.2. Problem Definition

The following sections describe problems of routing and fault notification in large networks.

1.2.1. Routing in Large Networks

Routing mechanisms currently used in the Internet are not suitable for provisioning services in large networks with multiple administrative domains [16] [86]. With the connection-oriented model of TE, centralized or distributed routing algorithms cannot be used to calculate end-to-end routes in large networks. Central route computation requires a computing node to have complete routing information about the network. For this to happen, each network node must advertise its state frequently to computing nodes-- which causes large computational and communication overhead for large networks. This also sacrifices network domain autonomy. Distributed routing

techniques rely on message flooding to calculate a route. They have much larger communication overheads [83] and perform poorly with respect to growth of network size. These conventional routing algorithms are not suitable for provisioning network services.

On the other hand, the Border Gateway Protocol (BGP) [41] is a connectionless inter-domain routing mechanism that is used in the Internet today. It has the following problems [16] [98].

- BGP has scalability problems. As the number of inter-domain routes increases, BGP consumption of a router's computing resources, e.g., CPU and memory, increases [98].
- BGP doesn't offer QoS guarantees and may cause network congestion [16] [98].
- The majority of BGP-routed paths are not optimal end-to-end routes. Most BGP border routers default to "hot potato" routing trying to forward packets as quickly as possible to another domain. This may result in routes that are optimized locally, within a number of network domains, but that are globally sub-optimal.

This thesis proposes a number of inter-domain routing protocols that calculate inter-domain end-to-end routes. These protocols may be used to admit service requests and to avoid congested inter-domain links.

1.2.2. Fault Notification Scalability

Service recovery maintains services in the presence of network faults. Fault notification constitutes a major part of recovery time. Experiments have shown that recovery times for moderate size networks are bounded at a small time, 4 ms, above fault notification time [42].

Scalability is a major concern for fault notification because the process results in flooding the network with recovery signals referred to as a "signaling storm" [12] [40]. The magnitude of the signaling storm grows dramatically with network size because: 1) services are likely to span

larger numbers of network nodes as network size grows; and 2) the number of services carried over an individual link grows with link capacity. This calls for scalable fault notification protocols and mechanisms to avoid signaling storms.

1.3. Thesis Scope and Methodology

The following sections describe the scope, methodology, and contributions of this thesis.

1.3.1. Thesis Scope

The focus of this thesis is on end-to-end routing and fault notification across multiple administrative domains. This section describes assumptions that bound the scope of the work. Figure 1.2 shows different timescales, models, network scopes and functions for TE. The shaded boxes are the focus of this thesis.

The thesis considers TE functions for medium temporal scales ranging from minutes to days. These functions dynamically manage network resources through *routing control* to meet resource optimization objectives. Network design and planning aspects of TE which have much longer timescales of months and possibly years, as well as nodal traffic management functions that operate at packet-level are beyond the scope of the thesis.

Figure 1.2: Thesis scope

The focus is creating end-to-end inter-domain service paths that meet QoS requirements. For this reason, the *connection-oriented* model of TE [15], where network resources are explicitly associated with services, is considered. The connectionless model of TE [8], where there is no explicit association between the services and resources, is not considered. The service routing protocols are deployed at inter-domain level. They rely upon well-studied intra-domain TE methods [84].

Finally, this thesis focuses on the dynamic end-to-end *path restoration* model of survivability as opposed to the *per-link* and *per-segment* protection. In particular, the focus is on fault notification aspects of the service recovery process. Issues such as: working and recovery path setup, fault detection, switchover from the working path to the recovery path and fault repair [12] are beyond the scope of this thesis.

1.3.2. Methodology

This section describes the methodology followed in carrying out this research. A review of TE literature helped guide this work. Network control was chosen as the general theme for the research. Next an analytical performance model for a GMPLS-based service creation architecture was developed. The model gave insights into the architectural features that affect the performance of service provisioning systems. These insights led to the choice of research topics and scope, namely, inter-domain service routing and fault notification. The *h*-protocols were then proposed. Their effectiveness has been analyzed and compared to results from the literature. Simulation experiments are used to validate and further study the performance and scalability of the *h*-protocols. Figure 1.3 illustrates this approach.

Figure 3.1: Thesis methodology and outline

1.4. Thesis Contributions and Publications

This thesis proposes protocols, referred to as *h*-protocols, for scalable service routing and fault notification across administrative domains. The service routing protocols aim to provide QoS guarantees and to enhance the overall utilization of network resources. The protocols select potential network resources to be traversed by services. The selection is controlled in order to load-balance traffic workloads among network domains and resources. The scalable fault notification protocol reduces the size of fault notification signaling storms. The *h*-protocols exploit the structure of hierarchical networks [6] [32] where the network hierarchy spans multiple administrative domains. Nodes at higher levels in the network hierarchy make routing decisions, maintain service information and participate in fault notification. In this thesis, a protocol is considered scalable with respect to a performance aspect if the respective aspect changes sub-linearly with the problem size (i.e., network size).

In general, the *h*-protocols can be used for creating and recovering critical services that contribute most to data traffic. Analysis of real life inter-domain data traffic for different service providers indicates that about 10% of services were responsible for about 90% of the bytes transmitted [87]. However, the advantages of the *h*-protocols come at the cost of larger service setup times and more communication and storage overhead when compared to existing protocols, namely the Private Network-to-Network Interface (PNNI) [3] protocol.

PNNI is a standard hierarchical network framework that is used in today's Internet. However, PNNI was not designed to support either traffic engineering or service recovery. In PNNI, hierarchical nodes do not contribute to routing decisions nor do they participate in service recovery. They only abstract and distribute link state information to their peer nodes.

This thesis focuses on inter-domain issues and exploits existing intra-domain TE techniques. The thesis contributions are as follows:

1. **New inter-domain routing protocols that achieve inter-domain TE objectives.** The protocols scale to large networks and assume a hierarchical network structure. To achieve TE, services are routed away from highly utilized inter-domain links. Route calculations at a higher level in the hierarchy guide route calculation at the next lower level in order to avoid highly utilized network domains as well. This requires the maintenance of state information about individual network resources and network domains. The protocols rely on intra-domain QoS routing algorithms [37] [83] to calculate intra-domain routes.

The protocols are: a) the Hierarchical Distributed Routing (HDR) protocol where routing computations at a hierarchy level are completed before routing computations start at the next lower level; and b) the in-Parallel Hierarchical Distributed Routing (HDR-P) protocol where route computation at a lower level in the hierarchy can start before a complete route at a

higher level in the hierarchy is calculated. Intra-domain routes are calculated in parallel within many network domains. HDR and HDR-P can be deployed based on the requirements of service users and providers. HDR-P has smaller service setup times when compared with the HDR. However, HDR has a smaller communication overhead and is easier to deploy.

2. **A novel protocol for end-to-end service routing that requires no global state.** H-DRA routes services with no central calculation of intra-domain routes. It enables the deployment of flooding-based routing algorithms, e.g., DRA [84], algorithms in large networks. These algorithms calculate intra-domain routes with no requirements for global maintenance of network state. H-DRA selects potential physical domains to be traversed by a service route. The selection of domains depends on resource availability of the links interconnecting them. DRA is employed only in these selected domains.
3. **A new Scalable Fault Notification (SFN)** protocol that reduces the size of fault notification signaling storms. In SFN, failure notification signaling messages travel vertically up and down the network hierarchy instead of horizontally along the paths. This provides signaling scalability for longer routes. SFN further reduces the number of fault notification signals by hierarchically aggregating/ deaggregating signaling messages based on the information maintained in the hierarchy. In case of a failure, nodes in the hierarchy send a single notification message per a path aggregate; instead of a number of them for each individual path. The performance of SFN is a function of the number of levels in the hierarchy levels while the performance of other standard and published fault notification protocols depends directly on network size. These advantages come at the cost of increased control overhead required in the network hierarchy and the non-trivial cost of implementing multicast.

4. **A queueing-based analytical performance model for MPLS-based service creation** is developed to study how GMPLS-based service creation is affected by network configuration. Network configuration is characterized by the number of network domains, the number of network nodes per domain and the rate of requests for services. The service creation system is modeled as layers of software processes that consume network, processor and disk resources in response to service creation requests. The model considers the following questions: a) how will the increase in demand for the services affect the performance of the service creation? b) what is the impact of increasing the number of network domains and network elements on the performance of service creation?

Parts of this thesis appear in the following publications:

1. M. El-Darieby, D. C. Petriu and J. Rolia, "HDRA: A Hierarchical Distributed Routing Algorithm," in Proceedings of the IEEE 2003 Global Communications Conference (Globecom'03), USA, December, 2003.
2. M. El-Darieby, D. C. Petriu and J. Rolia, "Hierarchical End-to-End Service Recovery," in Proceedings of the 8th IFIP/IEEE Symposium on Integrated Network Management (IM'03), USA, March, 2003.
3. M. El-Darieby, D. C. Petriu and J. Rolia, "A Hierarchical Distributed Protocol for MPLS Path Creation," in Proceedings of the 7th IEEE Symposium on Computers and Communications, (ISCC'02), Italy, July, 2002.
4. M. El-Darieby, D. C. Petriu and J. Rolia, "Scalability Analysis of Virtual Network-Based Service Provisioning," in Proceedings of the 7th IFIP/IEEE Symposium on Integrated Network Management (IM'01), USA, May, 2001.

5. M. El-Dariby and J. Rolia, "Performance Modeling of a Service Provisioning Design," in Lecture Notes in Computer Science, Vol. 1890, Editor: Linnhoff C, Springer Verlag, 2000.

1.4.1. Dissertation Overview

This thesis is organized as follows. Chapter 2 places this work in context of research work in the literature. Chapter 3 describes analytical performance models for GMPLS-based service creation developed to study parameters that affect scalability. Chapter 4 describes the HDR, HDR-P and H-DRA protocols. Chapter 5 describes the scalable fault notification protocol. Complexity analysis and simulation results that assess the performance of these protocols are presented in Chapters 4 and 5. The results are presented and compared to standard and published work. Chapter 6 summarizes the results of the thesis, offers conclusions and lists topics for future research.

Chapter 2: Related Work

2.1. Introduction

This chapter reviews work related to Internet Traffic Engineering (TE). TE controls Internet traffic in order to distribute it among network resources, avoid network congestions, and recover services in presence of network failures [16]. TE differs from traditional Internet mechanisms, such as shortest path routing [8], that do not consider these more advanced objectives.

This chapter is organized as follows. Section 2.2 describes work related to the connection-oriented model of TE and its major components; namely routing, signaling and bandwidth brokers (BB). Section 2.3 describes the connectionless model of TE. Section 2.4 describes service recovery. Section 2.5 gives a summary of this related work. Figure 2.1 illustrates the contents of this chapter.

Figure 2.1: Outline of the "Related Work" chapter

2.2. Connection-oriented Traffic Engineering

This section describes GMPLS research efforts related to connection-oriented TE. It focuses on the following techniques of TE: 1) *Routing* that selects network resources to participate in path provisioning; and 2) *Signaling* which allocates, re-allocates and releases resources. The section also describes the role of bandwidth brokers in deploying these techniques. GMPLS is typically deployed in conjunction with a constraint-based routing algorithm [37] and a signaling protocol, e.g., RSVP-TE [18]. The term MPLS is used to refer to the commonalities of GMPLS and MPLS in the remainder of this thesis.

2.2.1. Routing

Routing is a key component of connection-oriented TE [17]. This section describes different types of routing algorithms used to compute feasible paths from a source to a destination node subject to multiple constraints [37]. The constraints may reflect the availability of network resources as well as QoS requirements. The resulting routes are not necessarily the shortest, in terms of number of hops [14] [83].

Routing algorithms are classified into source routing, distributed routing and hierarchical routing. This classification is based on where routing computations are performed. The reader is referred to [83] for a detailed discussion of routing algorithms.

In *source routing*, a source node performs centralized route computations. Ideally, a source node has accurate knowledge of traffic demands and network-wide state information. However, this assumption is rarely satisfied. Generally, nodes exchange link state information through flooding of state advertisement messages. State information is rarely up to date. As a source node receives a routing request, it uses its current, but possibly out of date, information to calculate the route. Source routing is attractive in terms of controllability of routes and TE objectives. However, it

requires high processing power from networking nodes and is usually associated with a large state advertisement overhead that causes scalability problems. Chapter 3 considers a service creation model that is based on source routing.

In *distributed routing* algorithms, route computation is distributed among a number of network nodes. Although, this seems to overcome the overheads of source routing, it is not the case with all distributed routing algorithms. Some algorithms require each node to maintain network state information [36]. These algorithms share the problems of source routing [36]. However, flooding-based distributed routing algorithms do not have such state management overheads. They neither require the central computation of a route nor the maintenance of global state. In flooding-based distributed routing, each router participates in route selection based on its views of network topology. An example of these algorithms is DRA [23] [84].

For each route calculation, DRA selectively floods probing messages from a source towards a destination node. Each message records the route it traverses to reach the destination. Each node forwards probing messages on its outgoing links, except for the link the message arrived on. Message forwarding is controlled based on the availability of resources.

The advantages of DRA come at the expense of increased message overhead. In general, the message complexity of DRA for a single route computation is of $O(E)$ where E is the number of links in the network. E depends on the number of network nodes and their degree of interconnectivity [63]. Considering the increase in number of network nodes and the expectation of higher rates for route requests, the message overhead of DRA becomes prohibitive for large networks. Moreover, such signaling across domains sacrifices the autonomy of network domains. That is, one cannot accept the deployment of DRA in large multi-domain networks. Chapter 4 describes a mechanism that enables the deployment of DRA-like algorithms in large networks.

Hierarchical routing is traditionally used to deal with network scaling problems [6] [11]. Nodes are organized into different interconnected sub-networks called domains [6]. A domain consists of a number of interconnected nodes. A centralized resource manager [79] maintains topological and state information about the nodes and links of a domain. This allows network domain autonomy. In the Internet literature, network domains are referred to as Autonomous Systems (AS) [5] and a domain resource manager is referred to as a Bandwidth Broker (BB) [5]. Section 2.2.3 describes research efforts related to the design and operation of BB.

To build a hierarchical network, nodes at each level of the hierarchy are grouped into domains. Each domain is managed by a BB at the next level of the hierarchy. These BB constitute the nodes at that next higher level. For example, the physical nodes of a network belong to level-1 of the hierarchy. They are grouped into level-1 domains that are managed by next level BB. These BB are level-2 nodes. At level-2, the BB are interconnected in a manner that corresponds to that of level-1 domain connectivity. Level-2 BB are grouped into virtual level-2 domains. Each virtual level-2 domain is abstracted, represented and managed in turn by a single level-3 BB. Level-3 BB are interconnected in a manner that corresponds to the interconnectivity of level-2 virtual domains. The process of grouping BB into logical domains and abstracting such domains into BB continues until a hierarchy of BB is constructed. Figure 2.2 shows an example of a hierarchical network. The network hierarchy is assumed to be static. Fault-tolerant signaling channels between different same-level and child-parent BB are also assumed. This can be realized by dedicating nodal and link resources to these channels. BB* is a convention from the literature [32] that is used to denote the topmost bandwidth broker in the hierarchy.

ATM PNNI (Private Network-to-Network Interface) [3] is an example of a hierarchical routing protocol that scales to very large networks. PNNI supports a hierarchy of up to 105 levels [32].

In the ATM PNNI literature, network domains are referred to as Peer Groups and a BB is referred to as a Peer Group Leader (PGL) [3].

PNNI routing involves two components: a) a topology information distribution protocol and b) a path selection procedure. The topology information distribution protocol allows nodes in a domain to know about the intra-domain connectivity of nodes within their domain as well as about the inter-domain connectivity of network domains. Topology information is flooded to all nodes within a domain at each level of the hierarchy. A PGL abstracts the topology information within its corresponding domain and floods this information to PGL of its peers. As a PGL receives topology information, it floods it to the nodes of its domain. This is repeated for all domains at all levels of the network hierarchy. This information exchange enables source nodes to have abstract information to calculate a route to any destination node.

Figure 2.2: A hierarchical network

In PNNI, an edge node performs routing computations. An edge node uses the topological and state information it maintains to calculate a route within its domain and a domain-level route. A domain-level route lists only the domains to be traversed by the route to a destination node. Edge nodes of the selected domains calculate detailed routes within their respective domains. This distributes the end-to-end route computation among these edge nodes.

In PNNI, route computation workload is unevenly distributed among network nodes. Edge nodes of network domains are overloaded with routing computations while other domain nodes do not take part in routing computations [32]. The average routing computation load imposed on an edge node grows exponentially with the highest level this node appears at in the hierarchy [32].

PNNI is not designed to avoid inter-domain congestions. Edge nodes maintain state information about intra-domain connectivity but no state information about inter-domain connectivity. This may result in routing services over highly utilized inter-domain links [39].

Other hierarchical frameworks include the *viewserver* framework [35]. This framework provides a source node with all information required to calculate end-to-end routes. State information about links and domains are accumulated and forwarded to the source node. The source node, now having appropriate information, centrally calculates a route to the destination node. This violates service providers' requirements of hiding the internal details of network domains.

2.2.2. Examples of Signaling Protocols

In this section, the role of signaling in achieving TE objectives is highlighted. This section describes example signaling protocols used to allocate network resources to services. Signaling may also be used to recover services from failed resources as described in Section 2.4. The NSIS framework [50] is a set of recommendations on implementing signaling protocols.

The most well-known resource allocation signaling protocol is the Resource reSerVation Protocol (RSVP) [75]. RSVP is proposed within the context of Integrated Services Architecture (IntServ) [94]. It allocates network resources according to an application's QoS request. First, an end-to-end route is calculated and then signaling is used to allocate network resources along the route. RSVP maintains "soft state" information about paths. Soft states require periodic refresh messages to update path state along a path. In the absence of refresh messages, the state automatically times out and is deleted. Time-outs allow fault scenarios to be cleaned up.

RSVP is not considered scalable because it requires the maintenance of state information on a per-path basis and because of the large number of refresh messages associated with each path. QoS state aggregation is proposed to address this problem [34]. Aggregating the state of a number of individual RSVP paths into a single path increases the granularity of control state. This reduces the number of refresh messages core nodes need to process and the amount of information maintained at these nodes. IETF has developed Traffic Engineering extensions to RSVP (RSVP-TE) [18]. Chapter 3 describes detailed scenarios of how to create a service using these extensions.

YESSIR [78] is an example of a lighter weight signaling protocol that addresses several issues associated with RSVP. YESSIR simplifies the establishment of reserved paths. It piggybacks its messages to existing traffic in order to reduce its processing overhead. In other words, it uses data channels for signaling. YESSIR supports partial reservations where the traffic of a path may be transported on a best-effort basis on some links of the network. YESSIR is designed for intra-domain operations. Both RSVP and YESSIR are signalling protocols for the connection-oriented model of TE because they enable resource reservations on a per-path basis.

The Border Gateway Reservation Protocol (BGRP) [74] performs resource reservations for aggregate reservations, not on a per-path basis. This provides connection-oriented TE at the aggregate level, not at the per-path level. However, BGRP has excellent scalability features in terms of state storage and signaling that it is used to govern inter-domain resource reservations. BGRP builds a sink-tree that aggregates bandwidth reservations destined to the same destination from all sources in the network. A BGRP tree is rooted at the egress (sink) node of end-to-end paths. The leaves of the tree are the ingress routers of paths. BGRP routers do not maintain state information about individual reservations. BGRP routers maintain only tree information. The sink-tree reduces the number of reservations, saves on memory requirements and reduces the processing overhead of refresh messages. BGRP requires fewer refresh messages than RSVP because BGRP relies on reliable transport protocols to send its refresh messages. However, BGRP signaling messages must travel along the path to the destination domain to ensure resource availability. This increases the number of signaling messages involved in each reservation and causes scalability problems [90].

The BGRP+ architecture [90] is an inter-domain resource reservation approach that tackles scalability problems of BGRP [92]. In BGRP+, not every signaling message has to travel edge-to-edge. Moreover, extra resources may be reserved in advance at a BGRP agent, so that new service requests can be grafted. BGRP+ provides the complete specification of this mechanism, called “quiet grafting.” Together with the “sink tree”-based aggregations, “grafting” provides a scalable solution for inter-domain resource reservations. In addition, this enables BGRP+ to provide statistical quality assurances for aggregate reservations as described in [99].

2.2.3. Bandwidth Brokers

This section describes the role of BB in deploying TE techniques [5]. It also reviews some research efforts to enhance the functionality and performance of BB.

A BB acts as a resource manager responsible for admission control and resource provisioning within the domain it is managing. Inter-domain signaling between adjacent BB is used to dynamically negotiate Service Level Agreements (SLA) and resource allocations between domains. That is, a BB participates in two-level resource management in order to provision end-to-end QoS. Hence, BB are recommended *‘in any network where a neighboring network is accessed and some bilateral agreement is negotiated between the networks’* [5]. For example, BB are used to implement the surveillance and control of (G)MPLS-based networks [37].

BB are software processes that can run on network nodes or on separate server nodes. In the first case, a “leader election” [3] [32] protocol determines which node takes on the role of a BB. When a BB fails, a pre-determined backup node becomes the BB. In the latter case, server nodes are hosted within cluster-based server farms that can grow in capacity based on the intensity of route calculation requests. Server nodes are implemented in branch offices or Internet Data Centers. As a result, storage and communications overhead are of less concern. The first approach is assumed in this thesis.

A BB has the following two control components [53]: 1) an intra-domain control component that performs TE mechanisms and implements QoS routing protocols; and 2) an inter-domain control component to exchange inter-domain routing information. In [79], BB are used to perform intra-domain QoS routing control. The BB builds an overlay network of a set of paths between all node pairs. For every incoming routing request, the BB chooses a path satisfying QoS requirements of the path. Then, it updates link resources usage and returns the explicit route to

the source router. It must be noted that centralized servers have scalability problems, can be performance bottlenecks or single points of failure. The major scalability concerns of BB are memory requirements and the communication overhead between a BB and its routers.

At the inter-domain level, the Simple Inter-BB Signaling (SIBBS) protocol [70] defines how peer BB negotiate inter-domain bandwidth reservation. A BB generates a "resource allocation request" message and sends it to a BB in a peer domain. As a response, it receives a "resource allocation answers" message which acknowledges or rejects the request. However, having large numbers of BB as peers, which is the case in a 2 level network, is not scalable because the overhead of this negotiation process grows dramatically as the number of BB grows. For example, there were about 6500 domains in use in 01/2000 with 42% growth/year [21]. Each domain is managed by a BB. Since, SIBBS assumes a two-level network architecture, this would result in a 6500 BB, in the worst case, negotiating resource allocation. This thesis evaluates the potential of routing protocols that exploit more than two hierarchical levels to help overcome these problems.

A study of Inter-BB signaling is offered in [69]. The objective is to find optimal signaling granularity while taking into account the trade-off between QoS provisioning and scalability. The authors considered the "limited notification" approach where a BB notifies and reserves resources based upon significant changes in resource status, e.g., utilization. This reduces signaling and was found to offer reasonable QoS guarantees. This is similar to the "grafting" approach described earlier in Section 2.2.2.

To enhance BB scalability, a hierarchical architecture of BB is proposed in [82]. This architecture spans a network domain. It decomposes a network domain into a number of sub-domains, a subset of routers. The architecture has a number of edge BB and a centralized BB.

Each edge BB manages a network sub-domain. The centralized BB manages resource allocation among edge BB. In response to a routing request, an edge router forwards the request to the corresponding edge BB. This edge BB makes admission control decisions on behalf of the routers of its sub-domain based on the state information it maintains. If sufficient bandwidth is not available, the edge BB requests a new quota for the path from the central BB. If the request is granted, the edge BB admits the path request and updates path QoS state. This thesis describes a hierarchical architecture for BB to manage multi-domain networks.

2.3. Connectionless Traffic Engineering

This section describes adaptive routing and multi-path routing algorithms. The connectionless TE model uses these algorithms to avoid congestions.

Adaptive routing aims to overcome the limitations of shortest path algorithms. Routers dynamically set and advertise link weights to reflect local traffic conditions on each link [8]. This requires an extension to IGP or BGP. In [10] the authors propose an adaptive TE framework that operates on a network level. It is based on a link cost function being the sum of the measured average link delay. The algorithm adapts to traffic arrival (rates and patterns) and collectively yields an approximation of the global optimum solution. This allows network operators to change configurations of specific network links so as to reduce congestion. However, frequent adaptation to local traffic conditions may cause instability for the network as a whole [22].

In multi-path routing, several different paths are created between each source-destination pair of network nodes. This involves the computation of multiple paths and splitting traffic among these multiple paths. In [81], a centralized multi-path routing algorithm is proposed. The algorithm yields shortest multi-paths between pairs of nodes. This approach, then, determines optimal shares of traffic for the outgoing links of each node. In this way, this routing technique support

connectionless TE. Knowing traffic demands in advance results in better utilization of network resources. In this thesis, the utilization of a resource is defined as the ratio of its used capacity to its total capacity.

Inter-domain routing protocols (e.g., BGP) do not generally provide TE or QoS guarantees. However, methods have been proposed to extend BGP to provide some level inter-domain TE. The idea is to cause specific inter-domain traffic to prefer or avoid certain outbound egress points. BGP provides simple capabilities for this via the control certain attributes. For example, to choose an outbound direction for a domain for a specific destination, egress routers of a domain set the *Local_Pref* [41] attribute to a specific next hop. This directs traffic outgoing from the routers to this hop. In general, these mechanisms are effective, but they are usually applied in a “trial-and-error” fashion. A systematic approach for inter-domain TE is yet to be devised.

The connectionless TE model is easy to deploy. It is also scalable because there is no state maintenance in network nodes. However, with the connectionless approach, TE is achieved indirectly as a result of setting link weights. This leads to the absence of explicit operator control over TE decisions which limits the range of TE capabilities achievable [22] and prohibits long-term evolution of TE capabilities.

2.3.1 A Summary on Traffic Engineering Models

To summarize, connection-oriented TE requires network nodes maintain state information about paths they support. This helps to provide per-service QoS guarantees. However, maintaining the state on per-path basis has scalability problems. The connection-oriented TE model provides recovery mechanisms required to maintain service paths in the presence of network faults. Service recovery is discussed in detail in the following section.

With connectionless TE, there is no explicit service path setup. TE is achieved indirectly as a result of setting link weights and assigning priorities to different classes of services [89]. Hence, this model offers less strict QoS guarantees when compared with the connection-oriented model. This model provides limited recovery mechanisms. However, connectionless TE is easier to deploy, more scalable in that it does not maintain per-path state, and it does not suffer from the processing overhead associated with individual path setup.

2.4. Service Recovery

Service recovery refers to the capability of a network to maintain services in the presence of network faults. Service providers have experienced serious network failures that caused lengthy network outages. The impact of a service outage depends upon its effective duration. In case of a network failure, intra-domain and inter-domain routing protocols (e.g., OSPF, BGP) are notoriously poor at recovering from network failures [39]. Recovery times of between 15 minutes and 5.5 hours have been reported [39]. This is not acceptable for mission-critical applications that rely on such networks. This section gives an overview of service recovery processes. This section considers only data plane failures; for control-plane failures, the reader is referred to [24].

Service recovery involves the following processes [91]: failure detection and fault notification, recovery path setup, switchover from the working path to the recovery path, failure repair and a switchback from the recovery path to the working path. These processes contribute to recovery time-- the most relevant performance measure from a user's point of view.

Failure detection can be done by hardware to detect low layer impairments such as loss of signal. Routers may also use their control plane functionality to detect network faults through the regular exchange of routing protocol control messages. Examples of these messages include

Internet Control Message Protocol or RSVP HELLO messages. Then, a node that detects a failure starts fault notification. Failure notification techniques are described in Section 2.4.2

Service recovery has two basic models for the setup of recovery paths: 1) *protection* where a dedicated recovery path is established for a working path prior to the occurrence of network failures; and 2) *restoration* where a recovery path is established after the network nodes are notified of a failure.

Protection proactively allocates and reserves back-up resources. This provides faster recovery on pre-planned paths at the expense of an inefficient use of resources. Protection is often used with technologies such as wavelength division multiplexing. *Restoration* makes use of available resources based on the network state after the failure. Path resources are allocated reactively. This comes at the cost of slower recovery. Restoration is typically used with IP. It is noted that the terms protection and restoration are often used interchangeably. Estimates for service recovery times of different technologies at the IP, ATM and WDM layers are given in [96].

Recovery can be done on a per-link, on a per-segment of links or on a per-path basis. Link recovery typically restores a link between two network nodes. Per-path and per-segment recovery protect against all possible link failures along a path or a segment of a path. With per-path protection, the routes of the backup and working paths must be completely disjoint except for the end points.

The following recovery modes may be used with end-to-end per-path protection:

- $M:N$ where M recovery paths used to protect N working paths. Examples of this option include $1:1$ where one recovery path protects a working path, $1:N$ where one recovery path

is used to protect N working paths and $M:1$ where one working path is protected by M recovery paths.

- $1+1$ where data traffic is sent on working and recovery paths at the same time. The egress node selects one of the two data streams.

Mode $1:N$ is known as shared recovery mode where recovery paths are pre-calculated before a network failure occurs. Resource allocation is done after the failure occurs. Allocated resources may be shared among a number of recovery paths. This allows efficient use of spare capacity. In this thesis, only the end-to-end restoration of failed paths is considered.

2.4.1. Fault Notification

Fault notification constitutes a major part of recovery time. In this section, MPLS mechanisms for fault notification, namely flooding and signalling, are discussed.

MPLS may use flooding to advertise a failure to other nodes in the network. It can use OSPF link state advertisement to flood these messages. This typically takes tens of seconds [39] and creates a large number of messages. The total number of messages sent using flooding is simply equal to the number of links in the network. This can cause large processing overhead on network nodes.

MPLS may also implement signaling mechanisms for fault notification. This is known hereinafter as standard unicast failure notification. Signaling messages are sent to the ingress and egress of a path along the same route as the path. This is referred to as end-to-end path recovery. The number of signaling messages sent depends on path length. Significant processing time may be incurred at each node before forwarding the packet to the next hop. Moreover, a failed link/node may support a large number of paths, hundreds or thousands, causing many messages to be sent. The resulting signals form a *failure notification signaling storm*. In other words, the

magnitude of the storm grows with the number of services carried over each link. The number of services grows as links support greater bandwidths.

In general, MPLS failure notification is not scalable. That is why it has not been widely adopted by the industry. In [12] the authors propose a scalable path recovery mechanism for MPLS networks. The mechanism builds a point-to-multipoint Reverse Notification Trees (RNT). Each tree is rooted at the node that detects a failure. The leaves of the tree are the ingress routers of affected MPLS paths. Only one signaling message is sent along the shared segments of the RNT. This enables a reduction in the signaling overhead.

2.5. Concluding Remarks

Inter-domain TE aims to avoid highly utilized inter-domain links. Although inter-domain routing is a mandatory component of an end-to-end service provisioning architecture, the problems of inter-domain routing have not been addressed much in the literature [16]. Most research efforts focus on achieving TE within a network domain using the connection-oriented model of TE [86]. However, centralized and distributed computations of service routes that have been used for intra-domain TE do not scale for large multi-domain networks. Connectionless TE methods do not provide per-service QoS guarantees and typically cause congestions at inter-domain links.

Another objective for inter-domain TE is to achieving scalable fault notification and to avoid signaling storms. MPLS end-to-end failure notification has not been widely adopted by the industry because it is not scalable. MPLS use flooding or broadcasting to advertise a network failure to other nodes in the network. The number of signaling messages sent depends on path, network size, and number of paths carried over a failed link.

Chapter 3: A Model for MPLS Service Creation

3.1. Introduction

This chapter provides insights regarding the scalability of MPLS-based service creation. It describes a layered queueing model for creating MPLS paths that span a number of network domains. The model is designed to answer the following questions: a) how do increases in size of physical networks affect MPLS service creation performance under different loads? b) how sensitive is that performance to workload intensity characterized by the rate of requests for service creation? These parameters are believed to affect the scalability of MPLS service creation the most. The model also considers the effect of multiplexing a number of paths onto a single path as described in [34]. The model is based on the Traffic Engineering extensions to RSVP (RSVP-TE) [18] and is published in [58] [60].

This chapter is organized as follows: Section 2 describes the details of the model. The experimental design and performance results are described in Sections 3. Section 4 gives a summary of this chapter.

3.2. Modeling MPLS Service Creation

This section briefly introduces the Layered Queuing Modeling (LQM) [13] [49] approach for modeling the performance of computer systems. The section then describes an LQM model for modeling the performance of MPLS-based service creation.

3.2.1. Layered Queuing Modeling

There are two widely applied approaches to predict a system's behavior: a) use performance measurements from a controlled testbed; or b) use predictive models. Measurements-based

studies typically consume more time and are more expensive and they require a substantial prototype. Estimates from performance models can often reveal performance blunders [48] and provide feedback for making system design decisions even before a prototype exist. The Layered Queuing Modeling (LQM) [13] follows the latter approach to provide performance estimates.

A Layered Queuing Model describes request scenarios submitted to a system and the resources used by each request. Requests pass through layers of software processes, consuming network, processor and disk resources. An LQM is characterized by: 1) mix and rate of request scenarios by end users; 2) resource demands of requests (CPU, disk, network) as they pass through processes; 3) nature of interactions between processes, for example synchronous or asynchronous; 4) number of instances of each process and their maximum permitted level of concurrency; and 5) allocation of software processes to server nodes and resulting loads on the networks connecting them. LQM tools typically have a system description language that defines server nodes and their CPU and I/O rates, the underlying network infrastructure, software processes, their objects, methods and concurrency characteristics, sequences for the flow of control of scenarios and workload intensity and mixes.

The MoL tool [48] [49] can be used to build LQM models. The tool provides fast analytic performance estimates including: estimates for the utilization, mean response time and average queue length of software processes and for the mean response time of requests. These predictions include contention for both device and logical software process resources. The tool implements the algorithm of the Method of Layers [48] [49].

The MoL tool supports the design of full-factorial experiments through the definition of factors and levels for different parameters in a model. The MoL tool has two distinctive characteristics. First, it is *representationally* efficient. It permits the organization of system components

(hardware and software) into hierarchical packages. A package is a collection of system components. These packages can be replicated. As the workload for a system increases, the number of replicates of a server package can increase as well. The visits to the server are divided equally across the replicates. This reflects the behavior of a load balancing strategy in the model. The tool is also *computationally* efficient. Increasing the number of replicates does not increase the size of the model to be solved.

The solution of LQM is beyond the scope of this thesis. In general the layered model is decomposed into a series of queuing network models. Servers represent hardware resources and software processes. The models are solved iteratively using the Mean Value Analysis technique to provide performance measures for the system as a whole [48]. The results presented rely mainly on the LQM tool's ability to sum up resource demands for scenarios and to compute utilizations in the presence of package replication. Response time measures for scenarios are largely affected by a computation for the expected maximum of the response time of parallel activities. The tool relies on standard methods for computing this value. Furthermore, discrete event simulation can also be used to predict the behavior of the system under study. The focus of the thesis is on the challenge of describing an appropriate and useful model rather than on the underlying performance evaluation mechanisms.

3.2.2. Modeling Service Creation Operations

In this section, a scenario for creating MPLS-based services that span many administrative domains is presented. Typical service creation environments consist of: 1) a service management system that respects a service provider's policies, maintains service levels and bills users; and 2) a network system that establishes network paths. The network system consists of many domains. Each domain includes a number of network nodes (e.g., routers). A high-level scenario for

creating inter-domain MPLS-based services within such environments is presented below. This scenario is used to develop an LQM model for creating such services. The scenario is a best-case scenario in that it assumes no faults. Figure 3.1 illustrates this scenario along with estimates for service times for various processes expressed in micro-seconds. The steps of the scenario are:

- 1) a user issues a request for a service;
- 2) the *service management* process is invoked in order to check and respect service provider policies and then invokes a *network management* process to allocate network resources;
- 3) the *network management* process discovers available resources in different domains that may participate in the service. It consults their BB regarding resource availability;
- 4) the *network management* process centrally computes a domain-level route using BB data;
- 5) BB are notified and network resources are allocated within the corresponding domains;
- 6) the *network management* and the *service management* processes are notified when resource allocation is completed. The service is admitted and data packets are allowed to flow.

To allocate intra-domain network resources, ingress routers of network domains calculate an Explicit Route (ER) [33]. An ER is a best candidate route that meets QoS constraints. Signaling messages are exchanged between routers on the ER in order to allocate network resources for a MPLS service [93]. The scenario for intra-domain resource allocation is given below. The scenario uses the RSVP-TE as a signaling protocol. This scenario is abstracted in Step 5 of the high-level inter-domain scenario given above. The intra-domain scenario is as follows:

- a) within each domain, an ingress router calculates an ER and sends PATH messages to Label Switched Routers (LSR) along the ER.;

- b) each involved LSR checks if it has enough resources for the service and if so, as assumed, it forwards another PATH message, *downstream* along the ER, to the next LSR;
- c) Step *b* is repeated until the message reaches the egress router of this domain;
- d) at the egress router, the PATH message is processed, resource availability is checked and the PATH message is forwarded to the following domain;
- e) Steps *a* to *d* are repeated until the destination of the MPLS service is reached;
- f) the destination of the MPLS service sends back RESV messages, *upstream* along ER, to its preceding LSR;
- g) each LSR processes RESV message and allocates a new label for that MPLS path. The LSR confirms resource reservation and forwards a RESV message, *upstream*, along ER; and
- h) receiving the RESV message at the source of the MPLS service indicates that creation of the MPLS service and users can start sending data.

Figure 3.1: UML scenario diagram for MPLS service creation along with service times

These scenarios are used to build an LQM model for a MPLS path creation system using the MoL tool. The LQM model consists of a parent *system* package that groups all other packages. It contains a *user* node for modeling the behavior of users and the rate services are requested, a *SP* package for modeling the service provider environment and the *network* package for modeling the physical network. The *SP* package contains a *service_management* node that respects SP policies and manages services. The interaction between the *SP* and *network* packages is carried via the *WAN1* network component. Figure 3.2 illustrates the components of the model and their interactions.

The *network* package consists of a *network_management* node and many *domain* packages communicating with each other via the *WAN2* network component. The *domain* package contains *Edge_Router*, *LSR* and *BB* packages. An *Edge_router* node provides its functionalities through the following server processes: 1) *Ingress*, 2) *Egress* and 3) and *RSVP_Agent*. The Replication Factor (RF) represents the number of domains in the network.

Figure 3.2: Software architecture of MPLS-based service creation

To determine domain-level service routes, the *network_management* node interacts with *BB* of potential *domains* to be traversed by service routes. The selection of those domains is based on status information maintained by the *network management* process.

The LQM model captures requests among the different processes in the model, as shown in Figure 3.2. The *user* node visits the *service management* process once per each MPLS service creation request. The *network management* process decides which *domains* are used for the MPLS service. For each domain traversed by a service, the *network_management* process visits a *BB* once. A *BB* visits a number of *network nodes*. The *ingress* process of an edge router in each chosen *domain* is invoked to calculate an ER. The *ingress* process visits the *RSVP_Agent* at the same node once. The *RSVP_Agent* process of each LSR along ER is visited once to check resource availability. This is repeated until the egress router of the MPLS service is invoked. Then, the *RSVP_Agent* process of each LSR along the ER is visited once again to confirm resource allocation at each LSR. This is repeated until the *RSVP_Agent* of the ingress router is reached. The latter process visits the *network_management* and the *service management* processes to notify them of the creation of the MPLS service.

Visits between processes are synchronous, except for calls from the *network_management* node to different *BB*. These are represented in the model using asynchronous interactions followed by a join operation. No critical resources (such as operating system threads) are assumed held by a process while waiting for a reply for a forwarded message.

3.2.3. Hierarchical Tunnels

This section shows how to extend the LQM model to consider the impact of the "bundling" of MPLS paths into Aggregation Areas (AA) [52] on service creation time. An AA aggregates individual reservation sessions and is defined as "*a contiguous set of RSVP routers that are*

capable of performing RSVP aggregation along any possible route through this contiguous set."

An AA has two LSR routers known as aggregator and deaggregator that operate as its ingress and egress routers, respectively. The "aggregating" router is the first router that processes the PATH message as it enters the aggregation region. The "deaggregating" router is the last router in aggregation region to process the PATH message. Interior routers perform neither aggregation nor deaggregation.

An aggregate reservation is established between the aggregator and the deaggregator. An aggregate reservation carries the traffic of a large number of smaller paths. Only the aggregator and the deaggregator maintain state about individual reservations. In [52], the authors described a mechanism to identify the aggregator and the deaggregator within a domain. They also described a signaling scenario to establish an aggregate reservation between them.

To augment the LQM model to account for path aggregation, the *aggregation_area* package was added as a child package of the *domain* package. The *aggregator*, *deaggregator* and *forwarding engine (FE)* processes are parts of the *aggregation_area* package. The time taken by an *FE* to forward a packet is assumed to be negligible with respect to message processing. Other assumptions are as follows: the *aggregator* and *deaggregator* processes take the same time as an *RSVP-agent* in the above scenario; an aggregation area cannot span more than one domain. Two configurations of AA are considered and analyzed in the LQM model: 1) each AA cover one quarter of a domain and 2) each AA covers the domain completely. These are considered in addition to the case of having no aggregation areas as discussed above. Figure 3.3 illustrates these extensions to the *domain* package.

Bundling reduces the time required to establish new MPLS services because RSVP mechanisms within a domain are only exploited to create fewer but larger paths. Smaller embedded paths can

be created as parts of an AA without requiring the extensive message exchange of RSVP. This has potential for better scalability.

However, aggregating several end-to-end reservations also presents disadvantages. Only the aggregator and deaggregator maintain state for the different paths. This increases the probability of denying routing requests at the aggregator routers and of inefficient bandwidth assignment. It also increases the load on these routers because they have to process more state refresh messages. Another disadvantage is that a centralized aggregator/ deaggregator for an entire aggregation region may constitute a single point of failure.

Figure 3.3: LQM representation of hierarchical paths

3.3. Experiment Design

In this section, the scalability and performance of an MPLS service creation design is evaluated using the LQM model described above. The main performance metric considered is the time taken to create a MPLS path. This metric is defined as the time taken to establish a MPLS path—which is the summation of the time taken to find the shortest route between a source and destination nodes plus the time taken to allocate and reserve resources of different nodes along the route. The time to find a shortest path by the *Ingress* process is chosen as $O(N \log N)$ [83] where N is the number of network nodes per domain. A multiplier of 1 is used to inflate this

expression into a service time for the routing process [65] [80] [100]. The complexity of calculating a route may have even worse complexities [65]. This can be the case when considering different criteria for choosing a path, for example a path that balances the traffic loads on network links. Signaling time depends on the number of LSR (hops) in the ER.

Hence, service creation time is affected by: 1) *network size*: represented by the number of network domains and the number of nodes in each domain; 2) *workload intensity* imposed on the system; and 3) *different configurations* for aggregation of MPLS paths. The following parameters were considered as experimental factors:

1. # *domains*: that represents the number of domains traversed by a path, taking the levels of 2, 4 and 6. This value represents a path that spans multiple domains; and,
2. # *LSR*: that represents the average number of routers traversed by a path within a domain, with levels of 6, 10 and 20. This grows with the size of the network.
3. # *user nodes*: that represents workload intensity. The number of user nodes takes the levels 2, 25 and 50 nodes with 60 users per node. Users alternate between requesting a new MPLS service and thinking for 60 seconds. This determines the rate of MPLS service creation requests at ingress routers.

Three different configurations of aggregation areas are considered: 1) the service must choose between all LSR (no aggregation area) within a domain; 2) the service exploits paths of 4 aggregation areas; and 3) the service exploits one path to span the entire domain.

The total number of *LSR* in a domain is set to be 40 routers. This is based on the recommendations of the OSPF literature [65]. Also a physical network of a total of 10 domains is

considered. The corresponding service times are shown in Figure 3.1. These times are calculated for a path that traverses 6 domains and 20 routers within each domain.

The model assumes that all MPLS services are statistically identical in terms of the number of physical routers they exploit. Also, it assumes that each path within a domain traverses the same number of routers and that each router is visited, on average, by an equal number of paths. The visits among replicas of software processes are divided across the domains and their routers. Therefore, each process will receive, on average, an equal number of messages. This is a best-case scenario for the network in that load is evenly divided across routers. Also it assumes that resources are always available to satisfy the service creation requests. This is also optimistic.

One of the most difficult tasks when building predictive performance models is to estimate the resource demands for each process per scenario. The estimates used in this model are based on measurement results reported by other researchers for the OSPF protocol [65]. The goal with parameter estimates has been to capture the right order of magnitude. Actual parameter values will depend on product configurations. Even the ratios of CPU demand to network latency change as technology evolves. Regardless, the models can be used to study the sensitivity of system behavior to the CPU service demand and network latency estimates.

3.2.3. Performance results

The model is solved repeatedly to consider the full-factor experiment. The results are presented in Figures 3.4, 3.5 and 3.6. Values for the three experimental factors are represented along the x-axis of each figure. The y-axis shows the performance metric, i.e., mean service creation response time, as perceived by a user process.

The response time depends a great deal on the number of network nodes traversed by a MPLS path. It increases with the number of LSR and the number of domains. The *#LSR* experimental

factor has a big impact on the performance. This is because of the complexity of the shortest path algorithm. From Figure 3.4, for the case of 4 domains and 25 user nodes, response time grows from 194, to 264, to 627 ms as the number of LSR increases from 6, to 10 and to 20 for the no aggregation area configuration.

The number of domains traversed by a path also affects the response time. In Figure 3.4, for the case of 6 LSR and 25 user nodes, the response time grows from 139, to 194, to 237 ms as the number of domains grows from 2 to 4 to 6 for the no aggregation area configuration. This latency is governed by the expected maximum of LSP creation times across the chosen domain. Similar conclusions can be drawn from Figures 3.5 and 3.6. As global networking connectivity for the Internet grows, one can only expect SP to require support from multiple domain causing greater inter-domain latencies.

Aggregating MPLS paths also impacts service creation performance and appears to be useful for larger networks under high workload. This is expected due to the reduction in the number of messages exchanged. For example, for the case of 20 LSR, 6 domains and 50 user nodes the response time is reduced from 2483 ms with no aggregation involved to 1054 ms if the LSP traverses four aggregation areas per domain. It is further reduced to 595 ms when each domain is an aggregation area.

Figure 3.4: Estimated response times for the no aggregation area case

Figure 3.5: Estimated response times for the 4 aggregation areas case

Figure 3.6: Estimated response times for the one aggregation area case

The effect of increasing the arrival intensity of requests service creation requests is shown in Figure 3.7. The edge router CPU is the system bottleneck (it has the highest utilization). Software process threading levels of 10 were adequate to avoid software bottlenecks for the experimental factors considered. The utilization of the edge router CPU is dependent only on the number of user nodes, and almost independent of other factors. It increases from 5%, to 40%, to 80% as the number of user nodes increases from 2, to 25, to 50 nodes. Introducing two CPU in each edge router reduces the CPU utilization by approximately 50% of the values shown in Figure 3.7. This shows how important processor queuing can be in such systems. As resource demands grow, increasing the arrival intensity of requests causes increases in queuing for physical and logical resources leading to larger response times.

Figure 3.7: CPU utilization of edge router (one CPU per node)

3.4. Concluding Remarks

This chapter presented a performance model for MPLS service creation. MPLS service creation typically relies on intra-domain source routing [15] [93]. The mean response time of the service

creation process as a whole is dominated by the response time of the routing process because of the time complexity needed to find a route.

The response time of source routing depends a great deal on the number of network elements in a network domain. This is not a big surprise since a routing process that corresponds to Dijkstra's shortest path algorithm was assumed. Algorithms that take into account traffic engineering objectives would have higher complexities. Because of the time complexity for finding a route, source routing is not considered to scale as network size grows.

The other processes of the service creation process do not contribute much to response time of service creation. The response times of these processes are almost constant with respect to the levels of factor considered.

The number of domains in the network also affects service creation response time but in a lesser way than the number of nodes. As global networking connectivity for the Internet grows, one can only expect service providers to require support from more domains causing longer service creation times.

The performance of MPLS service creation also depends on the arrival rate of requests for services. As resource demands grow, increasing the arrival intensity of requests causes increases in queuing for physical and logical resources leading to larger response times. In a case study, the CPU of the edge router was found to be in a bottleneck. Introducing two CPU in each edge router reduced the CPU utilization by approximately 50% of the values with only one CPU in each edge router.

Aggregating MPLS paths appears to be useful for larger networks under high workload. It reduces the time required to establish new MPLS services because it reduces the number of

messages exchanged. This has potential for better scalability. However, aggregating several end-to-end reservations at the aggregator and deaggregator routers increases the load on these routers because they have to process more state refresh messages. This increases the probability of denying routing requests at aggregator routers and of inefficient bandwidth assignment. In addition, a centralized aggregator/ deaggregator for an entire aggregation region may constitute a single point of failure.

Another alternative to source routing is flooding-based distributed routing algorithms. Such algorithms require no central computation of a route or global state maintained at network nodes. Signaling messages are forwarded upon their arrival to a network node. However, one cannot accept the deployment of DRA in large multi-domain networks due to their large communications overhead. In such networks, as the demand for service creation intensifies, these algorithms would simply cause routers and network links to be flooded with signalling messages. Clearly, this is not scaleable. Several new protocols are proposed in the next chapter that overcome these problems

To summarize, designing a scalable network control plane for service creation calls for new routing protocols that do not suffer from the disadvantages of source and distributed routing algorithms. On one hand, source routing algorithms cause network edge routers to become performance bottlenecks because they bear the load for constraint-based routing computations. Increasing levels of service demands cause longer service setup times due to queuing at these routers. Other network nodes participate only in signaling and resource allocation. On the other hand, flooding-based distributed routing algorithms have a communication overhead that depends on the network size. This high communication overhead hinders the deployment of these algorithms in large multi-domain networks.

Chapter 4: Scalable Service Routing

4.1. Introduction

As demonstrated in the previous chapter, routing is the major performance bottleneck in the service creation process. This chapter describes three new scalable service routing protocols that better support increases in network size and user demands for services. The protocols load-balance traffic workloads among network domains and resources, calculate routes that cross administrative boundaries of different network domains and respect network domain autonomy. In this thesis, a scalable protocol in a performance aspect requires a sub-linear growth in this performance aspect as the size of the problem grows, i.e., as network size increases.

These protocols are the Hierarchical-Distributed Routing Algorithm (H-DRA), the Hierarchical Distributed Routing (HDR) and the in-parallel Hierarchical Distributed Routing (*HDR-P*). H-DRA uses flooding-based routing algorithms, for example DRA, to perform intra-domain routing. HDR performs intra-domain routing computations at a centralized server. *HDR-P* performs intra-domain routing calculations *in parallel* in order to reduce service routing time. In this thesis, the term “*h*-protocol” is used to refer to commonalities of these different protocols.

This chapter is organized as follows: Section 4.2 describes the *h*-protocols. Section 4.3 presents an analysis of the performance of the *h*-protocols and compares this to standard hierarchical networking protocols. Section 4.4 describes simulation experiments for evaluating the performance and scalability of the *h*-protocols. Conclusions are offered in Section 4.5.

4.2. The *H*-Protocols

In this section, a general introduction to the *h*-protocols is given. The operations of the three protocols are described in detail in Sections 4.2.1, 4.2.2 and 4.2.3. The following notation, summarized in Appendix A, is used in the description:

- src, dst : the source and destination nodes of a requested route;
- pid : an identification number that uniquely identifies a path;
- v_k : a level k node/ BB, physical nodes have $k=1$ and BB have $k > 1$;
- $parent_BB_k(v)$: a level k BB that governs node v ;
- P_k : a level k path that traverses a number of level k BB/ nodes;
- $P_k.ingress$: the ingress node for P_k ;
- $P_k.prev$ and $P_k.next$: the nodes preceding and following the current node along P_k , respectively; and,
- $RootBB(src, dst)$: the lowest-level BB that governs both src and dst . It is assumed to be at level $R+1$ (i.e., it is $parent_BB_{R+1}(src)$ and $parent_BB_{R+1}(dst)$).

The *h*-protocols assume a multi-level hierarchical architecture of the Internet. Hierarchical networks aid service routing. In general, the segmentation of the network into smaller domains reduces the complexity of intra-domain routing. It also reduces the number of state-update messages which in turn reduces the storage requirements at network edge nodes. However, network segmentation complicates inter-domain routing in terms of scalability, provisioning with QoS guarantees and maintaining the autonomy of service providers, as described in Section 1.2.1 and in [6] [37].

In the h -protocols, inter-domain routing favors less-utilized domains and inter-domain links to highly utilized ones provided they meet the QoS requirements. This results in avoiding network congestions and in distributing traffic workloads among network resources. However, a created path may not be the shortest, in terms of number of hops.

In order to calculate inter-domain routes, the h -protocols select network domains to be traversed by a route. This involves calculating routes at different levels of the hierarchy. First, *RootBB* is found at level $R+1$, then route calculation is started from level R and going down to level 1. A level k route traverses a number of level k BB. BB at a hierarchy level, say k , calculate or guide the calculation of a level $k-1$ route. Only the level $k-1$ domains that correspond to these BB are considered when calculating a level $k-1$ route. The selection of domains is dependant on state information BB maintain about domains and inter-domain links. The calculated level $k-1$ route traverses a number of level $k-1$ BB that correspond to level $k-2$ domains. The level $k-1$ BB calculate or guide the calculation of a level $k-2$ route. This continues down the hierarchy until potential level 1 domains are selected. Note that the connectivity among level k BB reflects the connectivity among level $k-1$ domains. At each level, the h -protocols use well known intra-domain route calculation algorithms such as Dijkstra's shortest path algorithm and DRA.

The three h -protocols follow different approaches to calculate an end-to-end route. In H-DRA, a level k BB controls the flooding of DRA message within its corresponding level $k-1$ domain in order to calculate a route within the domain. In HDR, a level k BB centrally calculates a route segment within its corresponding $k-1$ domain. The concatenation of different route segments results in the calculation of level $k-1$ route. With HDR, intra-domain route calculation at a hierarchy level is to be completed before the calculations at the next lower level are to start. However, to reduce HDR path setup time, *HDR-P* allows the calculation of an intra-domain

route at a lower level to start before the calculation of the higher level route is completed. After a route has been calculated, the *h*-protocols forward signaling messages along a path to confirm resource availability along that path.

With the *h*-protocols, if enough resources are not available, a route computation failure is generated. This requires an update of the information maintained at BB, and then a crankback procedure [3] [6] must be deployed to restart the computation of the route. This latter case is considered as future work.

The *h*-protocols are based on different assumptions than those of standard hierarchical networks. With the *h*-protocols a BB does not advertise information about its managed domain to peer BB. Only the BB managing a domain maintains information about that domain. BB expose only abstract information to their peer BB or to their managing BB at higher levels in the hierarchy. Hence, other nodes in the network may know only abstract information about resource availability in a domain through interactions with the BB. This complies with service providers' requirements for domain autonomy. However, the *h*-protocols reuse many configuration aspects of standard hierarchical network architectures such as the fully connected topology aggregation scheme [6] that is used to summarize domain information.

4.2.1. The H-DRA Protocol

This section describes the H-DRA algorithm. The algorithm can be decomposed into the following two steps: 1) finding *RootBB*; and, 2) deploying DRA at different hierarchy levels to calculate a route at that level. This is repeated until a level 1 route is calculated. H-DRA signaling is shown in Figures 4.1 (a), (b) and (c) while Figure 4.2 shows the pseudo code for the H-DRA algorithm.

The flooding of DRA *Probe* messages is controlled to meet QoS requirements and to avoid highly utilized inter-domain links. *Probe* messages are delayed for a specific period of time at each node. The duration of this delay period depends on the utilization of the link the message is to be forwarded on. That is, the higher the utilization of the link, the longer the delay a *Probe* message experiences. This way, *Probe* messages forwarded on less utilized links will reach the destination first. H-DRA, thus, prefers routes that have less utilized links.

The utilization of a level k link connecting two BB is calculated based on utilization of inter-domain links between the two domains. The utilization of a level k link is the result of dividing the summation of allocated bandwidth on these level $k-1$ links by the summation of maximum bandwidth available on all these inter-domain links.

Step 1 of the H-DRA algorithm is started when *src* receives a routing request and associates a unique *pid* with it. *src* sends a *Find_root* message up the hierarchy to its *parent_BB₂(src)* in search for *RootBB(src, dst)*. In general, a BB that receives a *Find-root* message, namely *parent_BB_k(src)*, checks the nodes it governs against *dst*. If it finds *dst* to be outside of its view, it sends a *Find_root* message to its parent BB, namely *parent_BB_{k+1}(src)*. This is repeated up the hierarchy until *RootBB(src, dst)* is reached. Figure 4.1 (a) illustrates this step.

Step 2 of the algorithm is started as *RootBB(src, dst)* initiates deploying DRA at level R . Figures 4.1 (b) and (c) show how DRA is deployed to execute step 2 of the algorithm. *RootBB(src, dst)* sends a *Notify* message to *parent_BB_R(src)*. The message specifies *parent_BB_R(src)* and *parent_BB_R(dst)* as the ingress and egress of P_R . *parent_BB_R(src)* starts forwarding DRA *Probe* messages towards *parent_BB_R(dst)*. Each *Probe* message records the route it traverses. A node that receives a *Probe* message deploys the DRA algorithm as specified in Section 2.2.1 above. Eventually, a *Probe* message reaches *parent_BB_R(dst)*. This *Probe* message determines the P_R

route. P_R lists a number of R level BB and specifies the level $R-1$ domains to be flooded with DRA messages.

$parent_BB_R(dst)$ responds by sending an *Ack* message to $parent_BB_R(src)$ along P_R . In response to receiving an *Ack* message, each node along P_R chooses inter-domain links to be traversed by level $R-1$ route and appends this to P_R . Choosing inter-domain links specifies the ingress and egress routes of adjacent domains.

In response to receiving an *Ack* message, $parent_BB_R(src)$ sends a *Notify* message to $parent_BB_{R-1}(src)$ to initiate the calculation of a route at level $R-1$. Notification messages bear information regarding path QoS requirements. This completes the calculation of a level k route. To achieve inter-domain TE, the selection of inter-domain links is based on the utilization information maintained about these links.

The same steps are repeated at different levels of the hierarchy. Figure 4.1 (c) shows these steps at the next lower level. The letter j is used in what follows to denote a general hierarchical level.

Figure 4.1 (a): Finding “ $RootBB(src, dst)$ ”

Figure 4.1 (b): H-DRA signaling at level 3

Figure 4.1 (c): H-DRA signaling at level 2

In general, a $parent_BB_j(src)$ that receives a *Notify* message starts the DRA probing phase. The message contains P_j that lists the domains at this level to be flooded with DRA messages and which inter-domain links connect them. $parent_BB_j(src)$ forwards *Probe* messages towards an egress of that same domain as specified in P_j . When a *Probe* message reaches the egress node of that domain, it first checks if it is $parent_BB_j(dst)$. If it is not, it forwards the *Probe* message along *only* the inter-domain link selected in P_j . In other words, *Probe* messages will not be forwarded to all neighboring domains. This constitutes the reduction in signaling overhead that H-DRA achieves.

For example, in Figure 4.1 (a), (b) and (c), node 4.2.3 receives a request to be connected to node 2.2.2. In search for a BB that governs both nodes, *Find_root* messages will be forwarded from 4.2.3 to 4.2 to 4 to BB^* . In this example, BB^* coincides with *RootBB*, but this may be different for other examples. The latter has view of the source and destination nodes. BB^* sends a *Notify* message to 4 with 4 as an ingress node and 2 as an egress node. *BB 4* starts flooding *Probe* messages towards *BB 2* within *domain** to calculate a level 3 path. The successful *probe* message is assumed to traverse *BB 4* and *BB 2*. This is shown in the figure by the path of the *Ack* messages. This implies that only domain 4 and domain 2 will be flooded with DRA messages. *BB 2* selects inter-domain link in response to receiving the *Probe* message. It chooses the link connecting *BB 2.4* and *4.1*. Then, *BB 2* forwards an *Ack* message towards *BB 4*. *BB 4* notifies *BB 4.2* with the following level-2 route: {[4.2,... 4.1], [2.4,... 2.2]}. Applying the same algorithm at level 2, the level-1 route will be: {[4.2.3,... 4.2.1], [4.1.1,... 4.1.3], [2.4.1,... 2.4.2], [2.2.4, ...2.2.2]}. Within each level 1 domain, each ingress node starts applying DRA to calculate an intra-domain route. This results in a level 1 route: 4.2.3, 4.2.1, 4.1.1, 4.1.3, 2.4.1, 2.4.2, 2.2.4, 2.2.3 and 2.2.2.

H-DRA handles cases of a lack of network resources in one of the logical or physical domains. Such errors may occur due to inaccuracy of the state information maintained at a *parent_BB*. A *Fail* message is sent upstream along P_k towards *parent_BB_k(src)*. *parent_BB_k(src)* will notify *parent_BB_{k+1}(src)* of the failure. *parent_BB_{k+1}(src)* initializes the calculation of new P_{k+1} after state information has been updated.

```

Processing at  $v_k$  {
  While(true){
    Wait (msg);
    Switch (msg→type){
      Case Find_root (){
        If ( $v_k$  does not have dst in its view) Send(Find_root, parent_BB( $v_k$ ));
        Else { Calculate  $P_R$  //@RootBB
              Send(Notify,  $P_R$ .ingress,  $P_R$ ); } }// end case
      Case Notify ( $P_{k+1}$ ){
        If (level==1 and  $v_k \neq src$ ) discard message;
        Initialize  $P_k$  to null
        For all outgoing link,  $M_j$ , of  $v_k$ 
          Send (Probe,  $M_j$ ,  $P_k$ ,  $P_{k+1}$ ); }// end case
      Case Probe ( $P_k$ ){
        If( $v_k$  forwarded a probe for the same pid) discard;
        Append  $v_k$  to  $P_k$ 
        If(level==1 and  $v_k == dst$ ) Send(Ack,  $P_k$ .prev,  $P_k$ );
        Else If ( $v_k$  is parent_BB(dst)){ Select inter-domain link //@dest and  $k > 1$ 
          Initialize  $P_{k-1}$  to null
          Append selection to  $P_{k-1}$ 
          Send(Ack,  $P_k$ .prev,  $P_k$ ,  $P_{k-1}$ );}
        Else If ( $v_k$  is an egress of current domain) Send (Probe,  $P_{k+1}$ .next,  $P_k$ ,  $P_{k+1}$ );
        Else For all outgoing links,  $M_j$ , of  $v_k$ 
          Send(Probe,  $M_j$ ,  $P_k$ ,  $P_{k+1}$ ); }//end case
      Case Ack ( $P_k$ ,  $P_{k-1}$ ){
        If ( $v_k == src$ ) Granted & route established;
        If (level == 1) Send(Ack,  $P_k$ .prev,  $P_k$ );
        Else { Select inter-domain link
              Append selection to  $P_{k-1}$ 
              If ( $v_k == parent\_BB_k(src)$ ) Send(Notify,  $P_{k-1}$ .ingress,  $P_{k-1}$ );
              Else Send (Ack,  $P_k$ .prev,  $P_k$ ,  $P_{k-1}$ );
            }/*end else*/
        }/* end case*/
    }/* end switch */
  }/* end while */
}/* end method*/

```

Figure 4.2: The H-DRA protocol

4.2.2. The HDR Protocol

This section describes the HDR algorithm and gives an example of its operations. The HDR is started when *src* receives a service routing request and uniquely identifies it with a *pid*. Then, HDR locates $RootBB(src, dst)$ as explained above in Step 1 of the H-DRA algorithm and in Figure 4.1 (a). $RootBB(src, dst)$ centrally calculates a route P_R within its level R domain. Then, it sends a *Notify* message to $P_R.ingress$ that starts an RSVP-style signaling phase by forwarding *Probe* messages downstream along P_R .

A node that receives a *probe* message checks its resources and if enough resources are available, it forwards the *Probe* message downstream along P_R . The arrival of a *Probe* message at $P_R.egress$ starts the *ack* phase of HDR at this level. $P_R.egress$ sends back an *Ack* message towards $P_R.ingress$ upstream along P_R . Figures 4.3 illustrates HDR signaling.

In response to receiving an *Ack* message, each node along P_R centrally calculates a level $R-1$ path segment within its level $R-1$ domain. This includes selecting inter-domain links connecting this domain to the previous and following domains along P_R . The selection can be based on TE policies such as using the widest inter-domain link. The protocol relies on standard mechanisms for intra-domain QoS routing [84] to calculate intra-domain routes. Then, the node forwards an *Ack* message upstream along P_R . Eventually, an *Ack* message reaches $P_R.ingress$ which sends a *Notify* message to $parent_BB_{R-1}(src)$ to initiate signaling at level $R-1$. The concatenation of level $R-1$ path segments results in the calculation of P_{R-1} . A *Notify* message, that contains P_{R-1} , is sent to $parent_BB_{R-1}(src)$. $parent_BB_{R-1}(src)$ starts signaling at level $R-1$. This is repeated at all levels of the hierarchy until a level 1 route is calculated.

Figure 4.3: HDR signaling

In HDR, the calculation of a level k route is done in series along P_k starting from the $parent_BB_k(src)$ moving towards $parent_BB_k(dst)$. Moreover, the calculation of a level k route must be completed before the calculation of lower level routes starts. Figure 4.4 lists the pseudo code for HDR.

HDR handles cases of a lack of network resources in one of the logical or physical domains. Such errors may occur due to inaccuracy of the state information maintained at the $parent_BB$. A *Fail* message is sent upstream along P_k towards $parent_BB_k(src)$. This node will notify $parent_BB_{k+1}(src)$ of the failure. $parent_BB_{k+1}(src)$ initializes the calculation of new P_{k+1} after state information has been updated.

For example, to connect nodes 4.2.3 and 2.2.2, *Find_root* messages will be forwarded from 4.2.3 to 4.2 to 4 to BB^* . The latter has view of the source and destination nodes. BB^* calculates P_3 route with 4 as an ingress node and 2 as an egress node and notifies BB_4 of that. BB_4 sends

probing messages along P_3 towards $BB\ 2$ within *domain**. $BB\ 2$ centrally calculates a segment of P_2 within its domain and selects the inter-domain link in response to receiving the *probe* message. We assume it chooses the link connecting $BB\ 2.4$ and 4.1 . Then, $BB\ 2$ forwards an *Ack* message towards $BB\ 4$. $BB\ 4$ notifies $BB\ 4.2$ with the following P_2 : $\{[4.2, 4.1], [2.4, 2.2]\}$. $BB\ 4.2$ starts sending probing messages along P_2 . $BB\ 2.2$ calculates a P_1 segment within its domain in response to receiving a probe message. It then sends an *Ack* message upstream along P_2 . Each BB along P_2 calculates a P_1 segment within its domain and forwards the *Ack* message along P_2 . This results in the following P_1 : $\{[4.2.3, 4.2.1], [4.1.1, 4.1.3], [2.4.1, 2.4.2], [2.2.4, 2.2.3, 2.2.2]\}$.

```

Processing at  $v_k$  {
  While(true){
    Wait (msg);
    Switch (msg→type){
      Case Find_root (){
        If ( $v_k$  does not have  $dst$  in its view)
          Send(Find_root, parent_BB( $v_k$ ));
        Else { $P_k$ = calculate route()//@RootBB
          Send(Notify,  $P_k$ .ingress,  $P_k$ ); } } // end case
      Case Notify ( $P_k$ ){
        If (level==1 and  $v_k \neq src$ ) discard message;
        Else Send (Probe,  $P_k$ .ingress,  $P_k$ ); } // end case
      Case Probe ( $P_k$ ){
        If(level==1 and  $v_k == dst$ ) //@ physical dest node
          Send(Ack,  $P_k$ .previous,  $P_k$ );
        Else If ( $v_k$  is parent_BB( $dst$ )){ //@dest and  $k > 1$ 
           $P_{k-1} +=$  calculate route segment()
          Send(Ack,  $P_k$ .previous,  $P_k$ ,  $P_{k-1}$ ); }
        Else Send(Probe,  $M_j$ ,  $P_k$ ,  $P_{k+1}$ ); } //end case
      Case Ack ( $P_k$ ,  $P_{k-1}$ ){
        If ( $v_k == src$ ) Request granted & Service created;
        If (level == 1) Send(Ack,  $P_k$ .previous,  $P_k$ );
        Else {  $P_{k-1} +=$  calculate route segment()
          If ( $v_k == BB(src)$ ) Send(Notify,  $P_{k-1}$ .ingress,  $P_{k-1}$ );
          Else Send (Ack,  $P_k$ .previous,  $P_k$ ,  $P_{k-1}$ );
        }
      }
    } /* end switch */
  } /* end while */
} /* end method*/

```

Figure 4.4: The HDR protocol

4.2.3. The HDR-P Protocol

To decrease the service routing time of HDR, HDR-P performs intra-domain route calculation *in parallel*. HDR-P does not require the completion of a higher level route calculation before starting the next lower level calculation of routes. In other words, the calculation of a level k route segment may start before the calculation of level $k+1$ route is complete. Figure 4.5 gives an example of HDR-P signaling. Figures 4.6 show the pseudo-code for the HDR-P algorithm.

Route calculation is triggered by the receipt of *Notify* messages. In general, a level j *BB* that receives a *Notify* message checks the resources of its domain and its inter-domain links. If there are enough resources, the *BB* calculates a route segment. This involves selecting an inter-domain link connecting these domains. Then, the *BB* sends a *Notify* message *in parallel* to each level $j-1$ *BB* along the calculated route segment to trigger simultaneous route calculations at these *BB*.

Figure 4.5: HDR-P signaling

The processing carried out by each notified BB is done in parallel. Each BB initiates the calculation of a route within its managed domain. The above procedure is repeated at all levels of the hierarchy until the physical nodes residing at level 1 of the hierarchy are reached. Notification messages are sent by level 2 BB to all physical nodes along calculated routes. The physical nodes try in parallel to allocate their required network resources. If enough resources exist, the nodes admit the request and positively acknowledge its managing BB through update messages.

```

Processing at  $v_k$  {
  While(true){
    Wait (msg);
    Switch (msg→type){
      Case Find_root (){
        If ( $v_k$  does not have  $dst$  in its view) Send(Find_root, parent_BB( $v_k$ ));
        Else Do_route(); //@RootBB}
      Case Notify ( $P_k$ ){
        If (level==1) { Allocate resources;
          If (success) Send (ack, parentBB()); }
        Else Do_route();} // end case
      Case Update (resources_allocated){
        Update local state info for that domain;
        If ( $v_k$  == RootBB(src, dst)) Send(ack, source node) //Request granted;
        If ( $v_k$  has received all updates)
          Send(Update, parent_BB( $v_k$ ), resources_allocated)
        Else wait;
      }
    } /* end switch*/
  } /* end while*/
} /* end method*/

method: DoRoute {
  Calculate an explicit route within that domain connecting ingress and egress;
  If (no route exist) Send (Fail, parent_BB,  $A_{in}$ , code);
  Else {
    Record information about the nodes along the calculated_route
    For (all nodes,  $v_j$  along the calculated_route) //Vertical Signaling
      Send (Notify,  $v_j$ );
  }
}

```

Figure 4.6: The HDR-P protocol

Update messages travel up the hierarchy following the reverse of the route followed by the notification messages. As the root node receives an *Ack* message for that path, it reports the success of path routing to the node that generated the service routing request. This completes the process of creating the path.

For example, in Figure 4.5, *RootBB* calculates a route in domain* that traverses BB 4 and 2. It then notifies the nodes with its decision. In response and at the same time, the two nodes start choosing ingress and egress points at their respective domains. Each BB calculates *in parallel* a route within its domain and notifies the corresponding BB. Assume this route traverses nodes 4.2, 4.1, 2.4 and 2.2.

Each of these BB applies, *in parallel*, the same algorithm. The figure shows the notification messages sent from BB 2.3 to physical nodes 2.3.1 and 2.3.2. If these nodes have enough resources to admit the request, they send *update* messages up the hierarchy. The messages are sent along the route followed by the notification messages but in the opposite direction. As all *update* messages reach BB*, it notifies the source node with the completion of routing.

With the *h*-protocols, in case not enough resources are available, a route computation failure is generated. This requires an update of the information maintained at BB, and then a crankback procedure [3] [6] must be deployed to restart the computation of the route. This latter case is considered as future work.

4.3. Evaluation of the Performance of the *H*-protocols

In this section, the performance and scalability of *h*-protocols are analyzed, simulated and compared to other hierarchical algorithms. The performance of *h*-protocols is characterized in terms of: 1) computational complexity of deploying a protocol; 2) communication overhead that reflects the number of messages exchanged among network nodes when establishing paths; and

3) information storage requirements of the protocols. Table 4.1 summarizes the time, message and information storage complexities of the h -protocols and PNNI. The following notation is used in this performance evaluation:

- m : the average number of nodes or BB in a domain;
- d : average number of nodes or BB in a domain traversed by a route;
- $(R+1)$: the rank of a hierarchical level where $RootBB$ of a route lies; and,
- D : the total number of domains involved in a route computation.

With the $path_length$ value, it is possible to give the compute and message complexity of the h -protocols. For a path of a given length, d is given by:

$$Path_length = d^R. \quad (4.1)$$

The total number of domains involved in calculating this route is given by:

$$D = \sum_{i=1}^{i=R} d^i. \quad (4.2)$$

This is because only one domain is involved at level R , d domains at level $R-1$, d^2 domains at level $R-2$ and d^R domains at level 1.

According to [63], the number of edges in an N -node domain, $E(N)$, can be estimated by:

$$E(N) = \frac{N}{2(t+1)} \left(1 - \frac{1}{N^{t+1}} \right). \quad (4.3)$$

where the value of t depends on the type of network as given in [63]. Based on these equations, the message and path setup complexities of each protocol are described below. Then the information storage requirements of the h -protocols are compared to those of PNNI.

H-DRA message complexity of deploying DRA at R levels would be of $O(D E(m))$ messages. The time DRA takes to calculate a route in a domain is of $O(d)$ time units. The time complexity of H-DRA is of $O(d D)$ time units.

The message complexity of HDR is of $O(d D)$ where a two phase signaling approach is assumed. Each BB is assumed to deploy a TE policy such as widest-first shortest path source routing algorithm that has a complexity of $O(E \log m)$ [36]. Therefore, the routing time complexity of HDR is of $O(D E(m) \log m)$.

In the *HDR-P*, the message complexity is of $O(d D)$. Since intra-domain route calculation is done at each level in parallel. That is the routing time at each level would be of $O(E \log m)$ as well. For R levels of BB, the route calculation time would be of $O(R E(m) \log m)$ units.

For the same network hierarchy, PNNI has a total number of messages on the order of $O(d^R)$. This assumes a two-phase signaling protocol. The total routing time would be number of physical domains traversed by the route, d^{R-1} , times the time complexity of the deployed source routing algorithm. Assuming Dijkstra's algorithm, with a complexity of $O(E(m) \log m)$ as a source routing algorithm, then the routing time complexity of PNNI is of $O(d^{R-1} E(m) \log m)$.

The *h*-protocols have a higher message complexity than PNNI because extra signaling is required at different levels of the hierarchy to enable inter-domain TE. The value for D is larger than d^R as shown in Equation 4.2. H-DRA has a higher message complexity than HDR because it uses flooding-based intra-domain routing. Message forwarding can be done at wire speed in case implemented at hardware levels. However, HDR is expected to have a greater elapsed time because of centralizing intra-domain route computations. However, PNNI does not provide inter-domain TE.

To study the storage complexity of state information, the amounts of information stored at BB and physical nodes are characterized. The h -protocols require BB to maintain information about its domain and its inter-domain links. The amount of information stored at each BB depends mainly on the number of nodes in a domain and the aggregation scheme used. For accurate schemes such as full mesh, a BB has to maintain information about $O(q^2)$ links that constitute the full mesh where q is the average number of edge routers of a domain. Other schemes such as the star network requires each BB to maintain $O(q)$ storage units.

In the h -protocols, physical network nodes maintain only local state. The amount of local state maintained is of $O(p)$ where p represents the number of links attached to each node. In contrast, PNNI uses link state-based protocols to announce routing information. Consequently, a physical edge node is required to maintain global state information about its domain, $O(m)$, and information about the connectivity between domains. Thus, the h -protocols have higher information storage requirement than PNNI.

Table 4.1: Performance of the h -protocols and PNNI

Protocol	Message Complexity	Time Complexity	Information Storage Complexity	
			<i>Within BB†</i>	<i>physical nodes</i>
H-DRA	$O(D E(m))$	$O(d D)$	$O(q^2)$	$O(p)$
HDR	$O(d D)$	$O(D E(m) \log m)$	$O(q^2)$	$O(p)$
HDR-P	$O(d D)$	$O(R E(m) \log m)$	$O(q^2)$	$O(p)$
PNNI	$O(d^R)$	$O(d^{R-1} E(m) \log m)$	<i>N/A</i>	$O(m)$

†: using the fully-meshed aggregation scheme.

To summarize, the h -protocols have higher message and information storage complexities when compared to PNNI. The H-DRA and HDR-P protocols have smaller path setup times than PNNI and HDR, which has the worst path setup time. The increased complexities of the h -protocols are the price for balancing data workloads among network nodes and maintaining domain autonomy.

4.4. Evaluating the Performance and Scalability of the H-protocols

This section evaluates, by simulation, the performance and scalability of the h -protocols. It also characterizes the computational workload imposed on BB and shows how the h -protocols distribute data workloads among inter-domain links. Simulation experiments are used to study and answer the following questions.

- How scalable are the h -protocols in terms of path setup time and communication complexity considering different network sizes?
- What is the effect of the structure of network hierarchy on the performance of the h -protocols?
- What is the average computational workload imposed on a BB for HDR and HDR-P? Note that H-DRA requires BB to only forward *Probe* messages.
- How does the h -protocol distribute data traffic on inter-domain links?
- For each scenario, what is the percentage of service requests that are admitted?

The main metrics of interest for this study are:

1. the path setup time of the h -protocols defined as the time taken to calculate a route;
2. the communication complexity of the h -protocols which is defined as the number of messages exchanged to enable the calculation of the route;
3. the average number of route computation performed by each BB when deploying HDR;
4. the distribution of utilization of inter-domain links. This distribution is based on inter-domain link utilization collected throughout the duration of the simulation experiments; and,
5. the percentage of admitted service routes.

The first two metrics are used to study the performance and scalability of the *h*-protocols as the network size grows and for different structures of the network hierarchy. Simulation results for these two metrics for HDRA, PNNI, HDR, HDR-P are reported in Sections 4.4.1, 4.4.2, 4.4.3 and 4.4.4, respectively. The third metric characterizes the computational overhead the *h*-protocols impose on BB. Simulation results for the metric are reported in Section 4.4.5. The fourth metric is used to report on how well data workloads are balanced across inter-domain links. The fifth metric describes how successful a protocol is in accepting routing requests. These metrics are affected by the size of the network and the structure of the network hierarchy. Results for the fourth and fifth metrics are reported in Section 4.4.6.

The OMNeT++ simulator [71] was used to further evaluate the performance of the protocols by developing simulation models for different network topologies and for the hierarchy structures. OMNeT++ is a generic and flexible object-oriented discrete event simulation environment. It is in widespread use for simulating computer networks and other complex IT systems. OMNeT++ has many concepts and features as those provided by other simulation packages such as OPNET. However, OMNeT++ is an open-source platform that supports code reusability. OMNeT++ comes with a large and reusable class library.

A simulation model in OMNeT++ consists of modules coded in C++. The modules are assembled together into larger modules using a high-level language, referred to as NED. NED is a topology description language used to define networks of modules. Modules communicate with message passing. Messages can contain arbitrarily complex data structures.

The pseudo-code presented in Figures 4.2, 4.4 and 4.6 is used to implement BB modules in OMNeT++. The NED language is used to group nodes and BB into domains. The language is

also used to build the hierarchy of BB. The simulation models used utilities provided by OMNeT++ such as random number seed generation tool and graphical output vectors.

Networks of 256, 1296 and 4096 nodes and hierarchies of 3 and 5 levels were considered. These result in the hierarchical structures shown in Table 4.2. The notation H_{l-m} is used to denote a hierarchy of l levels and an average domain size of m .

Table 4.2: Dimensions for network hierarchies studied

Hierarchy	H_{3-16}	H_{5-4}	H_{3-36}	H_{5-6}	H_{3-64}	H_{5-8}
Network size	256	256	1296	1296	4096	4096
Number of levels	3	5	3	5	3	5
Number of nodes per domain	16	4	36	6	64	8
Number of physical domains	16	64	36	216	64	512

Full duplex 155 Mbps links were assumed. Background traffic load for each link was chosen randomly from a uniform distribution with loads between 30 and 50 Mbps. Path routing requests arrive at each physical node of the hierarchy following a Poisson distribution with average inter-arrival times of 50 ms. Source and destination nodes of each path are randomly chosen. 2000 path requests are generated during each simulation run. No crankback is assumed when resource requirements can not be satisfied. For each scenario, eight simulation runs are carried out. The results reported are the average of individual replications of experiments. To avoid the transient response [101] at the start of the simulation, data collection was started after 1000 ms of the start of each experiment. Simulated path requests arrive until the simulation time is over. A confidence level of 90% is used to calculate confidence intervals for that average. The fully connected topology aggregation scheme [6] is used to build the network hierarchies.

Sections 4.4.1, 4.4.2, 4.4.3 and 4.4.4 present the communication complexity and path setup time of H-DRA, PNNI, HDR and HDR-P for the three networks considered, respectively. They also

show the effect of different hierarchy structures on the performance of the protocols. The results are consistent with the analytic expressions presented in Table 4.1.

In order to characterize the route computational workload imposed on BB, the number of route computations performed at each level of a network hierarchy for every 100 route requests is measured and divided by the number of BB at this level. This is done for HDR and HDR-P, and not for H-DRA, because they involve centralized intra-domain route computation. This number is averaged and reported with a 90% confidence level along the simulation experiments. Section 4.4.5 reports the average number of route computations performed by a BB for HDR and HDR-P protocols.

To study how the h -protocols distribute traffic workloads among inter-domain links, inter-domain link utilizations were collected every 100 ms, which is 2 times the average arrival rate for requests, throughout the duration of the simulation experiments. The utilization for each link is recorded for each simulation run. This procedure was followed for the h -protocols and PNNI.

Section 4.4.5 uses the distribution of data workloads to show how H-DRA and HDR distribute traffic workload among inter-domain links. An imbalance in link utilization, i.e., a large number of links have high utilization while others have low utilization, may reduce the percentage of requests that are admitted. The admission percentage is calculated by dividing the number of routes created successfully by the total number of routes requested. Admission percentage is averaged across the 8 simulation runs and reported with 90% confidence level. Section 4.4.6 presents average admission percentages of H-DRA, HDR and PNNI protocols.

4.4.1. H-DRA Performance

This section reports on the communication complexity and path setup time of H-DRA. Figures 4.7 and 4.8 show the communication complexity of H-DRA for the three network sizes

considered. Figure 4.7 shows the communication complexity for the 3-level network hierarchies, while Figure 4.8 considers the 5-level network hierarchies. The y-axis represents the number of signaling messages required to create each path with 90% confidence intervals given for each data point. The x-axis of the figures shows the length of created paths.

For the same network size, the communication complexity of H-DRA is affected by the depth of network hierarchy. From the figures and Table 4.3, a deeper hierarchy results in reducing the communication overhead required to provisioning a path for the same physical network. For the same network size, a deeper hierarchy results in smaller average domain size, which results in a smaller number of H-DRA probing message. This complies with Equation 4.3 in Section 4.3.

The scalability of H-DRA communication complexity can be studied as the network size increases from 256 to 1296 and to 4096 nodes in Figures 4.7 and 4.8, respectively. H-DRA requires a small increase in communication overhead to create a path of a specific length in spite of the increase in network size. From Table 4.3, the 3- and 5- level network hierarchies require a relatively small sub-linear increase in communication overhead compared to the growth in network size. This small increase is attributed to the scalability of H-DRA. However, for the 5-level network hierarchies, the increase is relatively smaller than for the 3-level hierarchies. This is due to the nearly equal network domain sizes of 4, 6 and 8 nodes with the 5-level hierarchies. For the 3-level hierarchies, domain sizes become significantly different, i.e., 16, 36 and 64 nodes. This increases the communication overhead within each network domain, and consequently the communication overhead of each network hierarchy is increased.

The length of the setup path affects H-DRA communication complexity. The longer the path the higher the communication overhead required. Typically, a longer path implies a larger number of domains involved in creating the path.

H-DRA has much higher communication overhead when compared to PNNI because it deploys DRA to calculate intra-domain routes. PNNI uses RSVP-like signaling to reserve resources along a calculated path. Thus, PNNI message complexity is equal to two times the path length. However, using DRA for intra-domain routing enables selecting the less utilized routes and consequently balancing data workloads among inter and intra- domain links. Section 4.4.5 provides a compares the performance of H-DRA to the other *h*-protocols and PNNI.

Figure 4.7: Average number of H-DRA messages for 3-level network hierarchies

Figure 4.8: Average number of H-DRA messages for 5-level network hierarchies

Table 4.3: H-DRA communication complexity for paths of 20 and 60 nodes

Path length	H ₃₋₁₆	H ₅₋₄	H ₃₋₃₆	H ₅₋₆	H ₃₋₆₄	H ₅₋₈
Network size	256	256	1296	1296	4096	4096
20	208	111	451	160	1000	298
60	423	266	953	376	2280	680

Figures 4.9 and 4.10 show the path setup times for H-DRA for the networks considered for the 3-level and the 5-level network hierarchies respectively. The y-axis of each figure represents the average time H-DRA takes to establish a path while the x-axis shows the length of the path. A summary of H-DRA setup times for paths of 20- and 60- nodes is given in Table 4.4. These results will be compared to setup time results of other protocols later.

H-DRA path setup time is primarily affected by the depth of the network hierarchy and path length, for the same network size. These effects are shown in the figures and in Table 4.4. The path setup time is significantly increased as the depth of the hierarchy is increased. This conclusion applies for all the 256-, 1296- and the 4096- node networks. This is due to the signaling done at higher levels of the network hierarchy. With H₅₋₄, two-phase signaling is performed in-series at levels 4, 3 before it is performed at level 2 of the hierarchy. With H₃₋₁₆, the signaling at level 2 of the hierarchy is started once *RootBB* has been identified. However, deeper H-DRA hierarchies require sub-linear increases in communication overhead to create a path of the same length.

The scalability of H-DRA path setup time can be studied as the network size increases from 256 to 1296 and to 4096 nodes in Figures 4.9 and 4.10, respectively. H-DRA requires a small increase in path setup time to create a path of a specific length in spite of the increase in network size. This is realized from Table 4.4 comparing the paths setup time of different hierarchies of

the same depth. In the table, there is a relatively small increase in path setup time compared to the growth in network size. This is attributed to the scalability of H-DRA.

H-DRA takes longer times to establish longer paths. This is an expected result because longer paths typically span a larger number of domains. H-DRA path setup times are compared to other *h*-protocols and PNNI in Section 4.4.5.

Figure 4.9: Average H-DRA path setup time for a 3-level network hierarchies

Figure 4.10: Average H-DRA path setup time for 5-level network hierarchies

Table 4.4: H-DRA setup times for paths of 20 and 60 nodes (in ms)

Path length	H ₃₋₁₆	H ₅₋₄	H ₃₋₃₆	H ₅₋₆	H ₃₋₆₄	H ₅₋₈
Network Size	256	256	1296	1296	4096	4096
20	120	304	199	433	343	578
60	375	817	702	1213	1338	1466

4.4.2. PNNI Path Setup Time

This section compares path setup time for H-DRA and PNNI. Figures 4.11 and 4.12 show the results for the 3-level and 5-level hierarchical structures. Average PNNI path setup times are shown, on the y-axis. Path lengths are shown along the x-axis. Table 4.5 shows a summary of setup times for paths of 20- and 60- nodes.

PNNI uses RSVP-like signaling to reserve resources along a calculated path. Thus, the number of signaling message PNNI requires to create a path is two times the path length. This complexity is dependent only on path length and is not dependent on other experimental factors. Thus, the message complexity of PNNI is not shown in the figures.

For the same network size, PNNI path setup time is primarily affected by domain size. From Table 4.5, PNNI takes longer times to calculate an intra-domain route within a larger domain. That is why H₃₋₆₄ takes longer times than H₃₋₃₆ which in turn takes longer than H₃₋₁₆. This is shown in Figure 4.12 and Table 4.5. Thus, a deeper hierarchy results in reducing domain size for a specific network and consequently PNNI path setup time is reduced.

Note that, as the domain size grows over 40 or 50 nodes, PNNI path setup time grows in a non-linear manner in terms of path length as shown in Figure 4.11 for H₃₋₆₄. For hierarchies H₃₋₃₆ and H₃₋₁₆ where the domain size is less than 45 nodes, PNNI path setup time grows in a linear manner in terms of path length. This complies with the recommendation of the OSPF literature of requiring a domain size of less than 45 nodes to achieve acceptable OSPF performance [65]. To

remove this non-linearity in performance, a deeper hierarchy, such as H_{5-8} , reduces the domain size to 8 and the non-linearity is removed as shown in Figure 4.12. Thus, deeper hierarchies may be advantageous for larger networks as long as domain size is less than 40 nodes.

PNNI takes longer times to establish longer paths. This is an expected result because setting up longer paths involves calculating a larger number of intra-domain route segments. H-DRA has smaller path setup time when compared to PNNI. This is because of the distributed intra-domain routing approach followed in H-DRA, compared to centralized intra-domain routing of PNNI. Comparing Tables 4.4 and 4.5 shows this. Section 4.4.5 compares PNNI performance to that of the h -protocols based on the information in these tables.

Figure 4.11: Average PNNI path setup time for 3-level network hierarchies

Figure 4.12: Average PNNI path setup time for 5-level network hierarchies

Table 4.5: PNNI setup times for paths of 20 and 60 nodes (in ms)

Path length	H ₃₋₁₆	H ₅₋₄	H ₃₋₃₆	H ₅₋₆	H ₃₋₆₄	H ₅₋₈
Network Size	256	256	1296	1296	4096	4096
20	379	328	856	586	2176	1194
60	732	709	1776	1282	4926	2343

4.4.3. HDR Performance

This section reports HDR communication complexity and path setup time. It also shows the effect of the hierarchy structure on these complexities. Figures 4.13 and 4.14 show the average number of HDR signaling messages required to create a path. The y-axis represents the number of signaling messages HDR requires to establish each path. The x-axis of the figures shows the length of created paths. Table 4.6 summarizes HDR communication complexity to create 20- and 60- node paths.

The depth of the hierarchy and the length of the established path affect the communication overhead of HDR. A deeper hierarchy increases the communication overhead of HDR. The longer the established path the higher the communication overhead required. Typically, a deeper hierarchy or a longer path implies a larger number of domains involved in creating the path. This implies a larger number of signaling messages between the BB managing these domains.

HDR is scalable in terms of communication overhead as the network grows in size. In Figures 4.13 and 4.14, HDR requires a sub-linear increase in communication overhead as the network size grows from 256 to 1296 to 4096 nodes. This is expected because HDR relies on centralized computation of intra-domain routes. That is why HDR requires less communication overhead over H-DRA. Comparing Tables 4.6 and 4.3 shows this. Section 4.4.5 compares HDR communication complexity to that of the other *h*-protocols and PNNI based on the information in these tables.

HDR requires more communication overhead than PNNI because of extra signaling performed at higher levels of the hierarchy to balance traffic loads among inter-domain links. PNNI uses RSVP-like signaling to reserve resources along a calculated path. The reader is referred to Section 4.4.5 for a comparison of the performance of both protocols.

Figure 4.13: Average number of HDR messages for 3-level network hierarchies

Figure 4.14: Average number of HDR messages for 5-level network hierarchies

Table 4.6: HDR/ HDR-P communication complexity

Path length	H ₃₋₁₆	H ₅₋₄	H ₃₋₃₆	H ₅₋₆	H ₃₋₆₄	H ₅₋₈
Network Size	256	256	1296	1296	4096	4096
20	100	141	120	167	147	200
60	307	392	400	476	538	527

The path setup times for HDR are given in Figures 4.15 and 4.16. Average HDR path setup times are shown in milliseconds along the y-axis of the figures while the x-axis represents the length of setup path. Numerical values for HDR setup times to create paths that traverse 20 and 60 nodes are given in Table 4.7.

For the same network size, the path setup time of HDR, like PNNI, is primarily affected by domain size. From the figures and Table 4.7, HDR takes longer times to calculate an intra-domain route within a larger network domain. That is why H_{3-64} takes longer times than H_{3-36} which in turn takes longer than H_{3-16} . A deeper hierarchy results in reducing domain size for a specific network and consequently HDR setup time is reduced.

The scalability of HDR path setup time can be studied as the network size increases from 256 to 1296 and to 4096 nodes in Figures 4.15 and 4.16, respectively. HDR requires a small increase in path setup time to create a path of a specific length in spite of the increase in network size. This is realized from Table 4.7 comparing the paths setup time of different hierarchies of the same depth. In the table, there is a relatively small increase in path setup time compared to the growth in network size. This is attributed to the scalability of HDR.

Note that, as the domain size grows over 40 or 50 nodes, HDR path setup time grows in a non-linear manner in terms of path length, similar to PNNI. This is because both protocols depend on centralized computations of intra-domain routes. This is shown in Figure 4.15 for H_{3-64} . For hierarchies H_{3-36} and H_{3-16} where the domain size is less than 45 nodes, HDR path setup time grows in a linear manner in terms of path length. This complies with the recommendation of the OSPF literature [65] where a domain size of less than 45 nodes is recommended to achieve acceptable OSPF performance [65]. To remove this non-linearity in performance, a deeper hierarchy, such as H_{5-8} , reduces the domain size to 8 and the non-linearity is removed as shown

in Figure 4.16. Thus, deeper hierarchies may be advantageous for larger networks as long as domain size is less than 40 nodes.

From Table 4.7, HDR takes longer times to establish longer paths. This is an expected result because setting up longer paths involves calculating a larger number of intra-domain route segments. However, paths of relatively the same length requires basically the same setup time.

HDR, like PNNI, requires more path setup time than H-DRA. This is due to the dependence of HDR and PNNI on centralized intra-domain route computation algorithms. H-DRA uses DRA to calculate these route segments. This can be shown by comparing Tables 4.7 and 4.4. The reader is referred to Section 4.4.5 for a comparison of the performance of these protocols.

HDR has a slightly larger path setup time than PNNI because of signaling at higher hierarchical levels. Both protocols calculate the same number of intra-domain route segments. However, HDR relies on signaling among hierarchical nodes to initiate this calculation. Comparing Tables 4.5 and 4.7 shows this. Section 4.4.5 compares HDR communication complexity to that of the other *h*-protocols and PNNI based on the information in these tables.

Figure 4.15: Average HDR path setup time for 3-level network hierarchies

Figure 4.16: Average HDR path setup time for 5-level network hierarchies

Table 4.7: HDR setup times for paths of 20 and 60 nodes (in ms)

Path length	H ₃₋₁₆	H ₅₋₄	H ₃₋₃₆	H ₅₋₆	H ₃₋₆₄	H ₅₋₈
Network Size	256	256	1296	1296	4096	4096
20	440	430	936	713	2283	1355
60	919	981	2057	1639	5344	2751

4.4.4. HDR-P Performance

HDR-P has the same communication complexity as HDR. HDR-P involves the same number of BB to calculate a route and requires the same number of signaling messages between these BB. Since cases of route computation failure were not considered, there is no crankback and no wasted messages for HDR-P. The observations about HDR communication complexity, described in Section 4.4.3, are the same for HDR-P communication complexity. Figures 4.15 and 4.16 give the communication complexity of HDR and HDR-P protocols in terms of the length of created path. Table 4.6 gives the number of messages HDR-P requires to create 20- and 60-nodes paths for the different hierarchies studied.

To study the path setup time of HDR-P, Figures 4.17 and 4.18 show this for the 3-level and 5-level network hierarchies, respectively. Average HDR path setup times are shown in ms along the y-axis of the figures while the x-axis represents the length of setup path. Numerical values for HDR setup times to create paths that traverse 20 and 60 nodes are given in Table 4.8.

The effect of hierarchy structure can be seen in the figures. Deeper hierarchies result in smaller HDR-P path setup times but more HDR-P communication overhead. This is shown in Table 4.8 and is due to smaller domain sizes.

HDR-P is scalable in terms of path setup-time as the network grows in size. From the figures and Table 4.8, HDR-P requires a small increase in path setup times to create a path of a specific length in spite of the increase in network size from 256 to 1296 to 4096 nodes.

The HDRP path setup time is marginally affected by the length of the established path. For example, H_{3-36} requires 205, 227 and 244 ms to establish paths of 20, 40 and 60 nodes, respectively. This is a relatively small increase compared to HDR or PNNI requirements.

HDR-P has the smallest path setup time when compared to HDRA, HDR and PNNI. HDR-P reduces the path setup time of HDR through exploiting the autonomy of BB and separation of network domains. Comparing Tables 4.8, 4.7, 4.5 and 4.4 shows this. Section 4.4.5 compares HDR-P performance to that of the other *h*-protocols and PNNI based on the information in these tables. However, more development overhead is required for deploying HDR-P. This is to account for the synchronization of parallel computations performed at different BB.

Figure 4.17: Average HDR-P path setup time for 3-level network hierarchies

Figure 4.18: Average HDR-P path setup time for 5-level network hierarchies

Table 4.8: HDR-P setup times for paths of 20 and 60 nodes (in ms)

Path length	H ₃₋₁₆	H ₅₋₄	H ₃₋₃₆	H ₅₋₆	H ₃₋₆₄	H ₅₋₈
Network Size	256	256	1296	1296	4096	4096
20	205	102	464	132	721	194
60	244	115	504	146	760	207

4.4.5. A Comparison of the Performance of the H-protocols and PNNI

This section compares the performance of the h-protocols and PNNI. Figures 4.19 and 4.20 show the communication complexity and Figures 4.21 and 4.22 report path setup time of the protocols to create paths of 20- and 60- nodes, respectively. The figures also show the effect of the hierarchy structure on the performance. The y-axis represents the number of signaling messages or setup time a protocol requires to establish a path. The x-axis of the figures shows the network hierarchy associated with each performance metric. The figures summarize the information presented in Tables 4.3 to 4.8

To compare the communication complexity of the protocols, HDRA has the highest overhead compared to PNNI, HDR and HDR-P because it deploys DRA to calculate intra-domain routes. HDR and HDR-P have the same communication complexity. PNNI has the smallest communication overhead when compared to the other protocols. PNNI uses RSVP-like signaling to only confirm resource reservations. The h-protocols rely on signaling not only for confirming resource reservation but also to calculate inter-domain routes and load balance traffic workloads among inter-domain links. That is why HDR has a higher communication overhead than PNNI. These results are shown in Figures 4.19 and 4.20.

The structure of network hierarchy affects the communication complexities of the protocols. A deeper hierarchy reduces the communication overhead required to provision a path. This is because the deeper the hierarchy, the smaller the average domain size for the same network size and the faster intra-domain routing calculations. These conclusions are valid for the different networks considered and for different path lengths.

Figure 4.19: Message complexity of the h-protocols and PNNI to create a 20-node path

Figure 4.20: Message complexity of the h-protocols and PNNI to create a 20-node path

To compare the path setup time of the protocols, HDR-P has the smallest path setup time when compared to H-DRA, HDR and PNNI. HDR-P exploits network domain autonomy to calculate intra-domain route segments in parallel. However, a greater effort is needed to implement and deploy HDR-P in networks. This is because it is necessary to synchronize the parallel computations performed at different BB. H-DRA has a smaller path setup time when compared to PNNI and HDR. This is because of the distributed intra-domain routing approach followed in

H-DRA, compared to centralized intra-domain routing of PNNI and HDR. HDR has a slightly larger path setup time than PNNI. This is due to extra signaling performed among BB to guide route calculation that balances loads. These conclusions are shown in Figures 4.19 and 4.20.

The four protocols are scalable protocols in terms of path setup time. Both protocols require a small increase in path setup times to create a path of a specific length in spite of the increase in network size from 256 to 1296 to 4096 nodes.

The structure of network hierarchy affects the path setup of the protocols. A deeper hierarchy increases the path setup times H-DRA requires but reduces the path setup times for HDR, HDR-P and PNNI. This is because the deeper the hierarchy, the smaller the average domain size for the same network size and the faster intra-domain routing calculations.

Figure 4.21: Path setup times of the h-protocols and PNNI to create a 20-node path

Figure 4.22: Path setup times of the h-protocols and PNNI to create a 60-node path

4.4.6. Path Computation Load

This section characterizes the computational workload imposed on BB to perform routing computations. In order to characterize the route computational workload imposed on BB, the number of route computations performed at each level of a network hierarchy for every 100 route requests is divided by the number of BB at this level. This is represented along the y-axis of Figures 4.23 and 4.24. The x-axis of the figures shows the level of the network hierarchy.

BB at higher hierarchy levels are subject to a higher average computation overhead than BB at lower levels. This is because the number of BB at the lower level, say L , would be d times the number of BB at level $L+1$ (where $L > 1$). For example in Figure 4.23 for hierarchy H_{3-36} , BB at levels 2 and 3 perform on average 10.5 and 46.5 route computations, respectively. The same conclusion applies for hierarchies H_{5-6} , and H_{3-64} . Note that not all route computations are associated with BB*. That is, paths may have their topmost managing BB at a lower level than that of BB*. Note also that a uniform set of path requests is assumed in this thesis. This represents a worst case scenario because it does not follow the locality characteristics of typical workloads.

In general, the higher the level of a BB, the higher average computation overhead imposed on the BB. The top most BB of a network hierarchy, i.e., BB*, performs 40 to 55 route computations for each 100 route requests. This may cause BB* to be a performance bottleneck. Increasing the number of CPU's at a BB, as described in Section 3.2.3, reduces the average number of computations performed per CPU at this BB. This principle can be applied at different BB to enhance the performance by balancing the computation workload on BB. Finding the optimum configuration for a network hierarchy based on the availability of resources is considered as future work.

Figure 4.23: Average number of route computations/BB/100 request for 3-level hierarchies

Figure 4.24: Average number of route computations/BB/100 request for 5-level hierarchies

4.4.7. Traffic Engineering and Workload Distribution

This section presents simulation results that characterize how the *h*-protocols distribute data traffic among inter-domain links and compares this to PNNI. Figures 4.25 to 4.30 show the distribution of inter-domain link utilization for the six hierarchies under study. The y-axis of the figures gives the cumulative number of inter-domain links with a utilization that is less than or equal to a specific utilization. The x-axis gives the corresponding link utilizations. Running a Kolmogorov-Smirnov Test [101] confirmed that the results presented in the figures for H-DRA and HDR are significantly different from PNNI results with 90% confidence level.

The figures show that the *h*-protocols better balance data traffic loads across inter-domain links than PNNI. For example, for the 256 node network, Figures 4.25 and 4.26 show that PNNI has a larger number of links with utilization less than or equal to 45% when compared to H-DRA and HDR. The cumulative number of links with utilization of less than or equal to 60% or with utilization less than or equal to 65% for the three protocols is basically the same in the two figures. This implies that both HDR and H-DRA have a greater portion of inter-domain links with moderate utilization between 45% and 65%. For utilizations in the range of 60 to 80%, both HDR and H-DRA have a greater portion of inter-domain links within this range than PNNI.

The same conclusions apply for larger networks as is shown in Figures 4.27 to 4.30. Thus, the *h*-protocols better balance data traffic loads across inter-domain links than PNNI.

In general with the *h*-protocols, the larger the network size, the lower the link utilization range that corresponds to most links. This is because the same data workload was applied in the simulation experiments to all three networks and larger networks have more inter-domain links. For example, in a 256- and 4096- node network, most links have a utilization range of 45 to 65% and 30 to 45%, respectively.

A deeper hierarchy implies a larger number of inter-domain links at the physical level of the hierarchy when compared to a hierarchy of lesser depth. The number of inter-domain links of each hierarchy is shown along the y-axis of Figures 4.25 to 2.30. For example, for the 256 node network, hierarchy H_{3-16} has 32 inter domain physical links while hierarchy H_{5-4} has 160.

Figure 4.25: Average cumulative distribution of the inter-domain links for H_{3-16}

Figure 4.26: Average cumulative distribution of the inter-domain links for H_{5-4}

Figure 4.27: Average cumulative distribution of the inter-domain links for H_{3-36} Figure 4.28: Average cumulative distribution of the inter-domain links for H_{5-6}

Figure 4.29: Average cumulative distribution of the inter-domain links for H₃₋₆₄ Figure 4.30: Average cumulative distribution of the inter-domain links for H₅₋₈

The distribution of data workloads affects the admission percentage of requested routes. Figure 4.31 shows, along the y-axis, the average admission percentage of H-DRA, HDR and PNNI protocols for each network hierarchy, shown along the x-axis. A “t-Test: for Paired Two Sample for Means” [101] shows that the PNNI results are significantly different from those obtained for H-DRA and HDR for all reported results with confidence level of 90%.

H-DRA has the highest admission percentage whereas PNNI has the worst performance in accepting new route requests. This is because H-DRA has the best performance in balancing the loads among inter-domain links. H-DRA selects the least utilized links when calculating a route, as described in Section 4.2.1. In addition, the decision of forwarding an H-DRA probe message is based on the utilization of the link the message is to be forwarded on. Network nodes forward probing message based on the information they maintain locally. This is the most up-to-date state information because it does not involve the time delay associated with flooding link state information across the network.

Using centralized intra-domain route computation lowers the route admission percentage of HDR and PNNI when compared to H-DRA. With HDR, BB at different levels of the hierarchy calculate inter domain routes. The accuracy of information maintained at BB at different levels in the hierarchy affect routing decisions made at lower levels. The higher the level of a BB, the more inaccurate information it may have. Also, the utilization and capacity of inter-domain links at higher levels in the hierarchy is based on aggregating the utilization and capacity of lower levels inter-domain links. This applies for H-DRA as well as for HDR. The accuracy of inter-domain link aggregation affects the accuracy of information maintained at higher levels of the hierarchy. Hence, deeper hierarchies generally have lower admission ratio.

PNNI calculates routes based on state information maintained at source nodes. Such information is least likely to reflect the current state of a link because of the time delay involved in flooding state information across the network to source nodes. As shown in Figure 4.31, PNNI has lower admission percentage when compared to HDR and H-DRA for all hierarchies considered.

For the three networks considered the admission percentage increases as the network size increases for the three protocols. In general, larger networks have fewer congested inter-domain

links and consequently higher admission percentage. In some respect this is an anomaly of the experiment because data traffic is not scaling with network size.

Finally, PNNI uses crankback [3] to enhance its admission ratio. The percentages given in the figure did not account for successful admissions that result from PNNI crankback. The *h*-protocols can be also extended to incorporate crankback mechanisms in order to enhance their admission percentage. With the *h*-protocols, a failure in calculating a route could, for example, cause *Update* messages to travel up the hierarchy to update the state information maintained at BB. Exploration of crankback for the *h*-protocols is considered future work.

Figure 4.31: Average admission percentages for all hierarchies for H-DRA, HDR and PNNI

4.4.8. Summary of Results

The *h*-protocols avoid highly utilized inter-domain links. Simulation results show that the *h*-protocols cause a large number of inter-domain links to have moderate utilization while the number of inter-domain links with very high or with very low utilization is low. This causes the *h*-protocols to have higher admission percentage of requested routes when compared to PNNI. PNNI initially rejects more route requests because it has a large number of inter-domain links

with very high or with very low utilization. It is noted that PNNI crankback mechanism can then be used to find alternative routes.

Other factors that may affect the admission percentage of a hierarchical routing protocol are: the type of intra-domain route computation algorithm and the depth of the hierarchy. Flooding-based routing algorithms typically have a higher admission percentage than centralized algorithms. They use link state information maintained at network routers controlling these links. Centralized routing algorithms typically rely on information disseminated across the network. Information dissemination involves a time delay which may cause the inaccuracy of information.

The deeper the hierarchy, the lower the admission percentage for hierarchical routing protocols. The accuracy of information maintained at BB affect routing decisions made at lower levels. The higher the level of a BB, the more inaccurate information it may have.

The results presented in this chapter also compare the performance of the H-DRA, HDR and the HDR-P protocols. H-DRA has the highest communication overhead compared to PNNI, HDR and HDR-P because it deploys DRA to calculate intra-domain routes. However, H-DRA has a smaller path setup time when compared to PNNI and HDR. This is because of the distributed intra-domain routing approach followed in H-DRA, compared to centralized intra-domain routing of PNNI and HDR. HDR-P has the smallest path setup time when compared to H-DRA, HDR and PNNI. However, a greater effort is needed to implement and deploy HDR-P in networks. This is because it is necessary to synchronize the parallel computations performed at different BB.

H-DRA and HDR-P are scalable protocols in terms of path setup time. Both protocols require a small increase in path setup times to create a path of a specific length in spite of the increase in network size from 256 to 1296 to 4096 nodes.

The HDR and PNNI protocols calculate the same number of intra-domain route segments. However, HDR has a slightly larger path setup time than PNNI. This is due to extra signaling performed among BB to guide route calculation that balances loads.

The structure of network hierarchy affects the communication and path setup complexities of the *h*-protocols. A deeper hierarchy reduces the communication overhead H-DRA requires to provision a path. However, it increases the path setup times H-DRA requires. Deeper hierarchies result in smaller HDR and HDR-P path setup times but more communication overhead. This is because the deeper the hierarchy, the smaller the average domain size for the same network size and the faster intra-domain routing calculations. However, BB at higher hierarchy levels are subject to a higher average computation overhead than BB at lower levels.

4.5. Concluding remarks

This chapter presented novel scalable service routing protocols, referred to as the *h*-protocols. The protocols scale up to large networks by exploiting a hierarchical network architecture. Nodes at higher levels of the network hierarchy select routes based on state information of intra- and inter-domain links. Links are chosen if they have enough resources to meet QoS requirements. Based on the results of the simulation study, the *h*-protocols require a small increase in the computational or communication overhead to create same-length paths as network size increased.

The *h*-protocols address the disadvantages of hierarchical inter-domain routing algorithms such as PNNI. The *h*-protocols better balance data traffic loads across inter-domain links than PNNI. Experimental results have shown that the *h*-protocols have a greater portion of inter-domain links with moderate utilization and fewer links with very high or low utilization than PNNI.

Distribution of data workloads affects the admission percentage of requested routes. The *h*-protocols have higher admission percentage than PNNI. H-DRA has the highest admission percentage among all protocols due to its usage of flooding-based routing algorithms. Using centralized intra-domain route computation lowers the route admission percentage of HDR and PNNI when compared to H-DRA. PNNI may use crankback mechanism [3] to enhance its admission percentage. The *h*-protocols can be extended to incorporate crankback mechanisms in order to further enhance their admission percentage.

Chapter 5: Scalable Fault Notification

5.1. Introduction

Quick recovery from network failures is an important aspect of QoS for network services. End-to-end service recovery generally requires the notification of ingress nodes about network failures, as described in Section 2.4. Such end-to-end fault notification typically results in flooding the network with recovery messages. This is referred to as a signaling storm [40]. A large signaling storm may affect the performance of network nodes. This chapter describes a new Scalable Fault Notification protocol (SFN) that reduces the magnitude of such storms and analyzes its performance. SFN assumes a hierarchical network architecture as described in Section 2.2.1 and works in conjunction with the *h*-protocols.

The goal of SFN is to reduce the magnitude of a signaling storm. SFN signaling effectiveness is improved by having failure notifications travel up and down a network hierarchy instead of horizontally along the paths of affected services. SFN further alleviates the size of the signaling storm by using multicast to aggregate signaling messages. The protocol uses path management information maintained in the network hierarchy to aggregate messages. The *h*-protocols create and maintain such information.

To the best of the author's knowledge, this work is the first to propose hierarchical fault notification. SFN is presented in the context of MPLS because the fastest end-to-end path recovery is achievable at the MPLS layer. This is in addition to the other advantages of MPLS described in Section 1.1.

This chapter is organized as follows: in Section 2, SFN is described. Section 3 provides an analysis of SFN performance. Section 4 describes simulation experiments conducted to evaluate the performance and scalability of SNF and compare it with other approaches. Conclusions are offered in the Section 5.

5.2. Hierarchical Fault Notification

In this section, two versions of SFN are described. One version of SFN is based on unicast signaling and is called *unicast-SFN* and the other takes advantage of the savings multicast achieves and is called *multicast-SFN*. The following notation is used in this description. It is also summarized in Appendix A:

- v_f : a level l node that detects the failure;
- P : a path that is affected by the failure;
- $Plist_k$: a list of pid of affected paths that is composed by a level k BB and sent in a *Failure* message to a level $k+1$ BB; and,
- $AggP_k$: a level k aggregate path that may aggregate a number of level $k-1$ aggregate paths, each of which may aggregate in turn a number of level $k-2$ aggregate or a set of non-aggregated paths. In other words, an aggregated path is a rooted hierarchy of aggregate and non-aggregated paths.

5.2.1. Unicast-SFN

This section describes the unicast-SFN algorithm and gives an example of unicast-SFN signaling. The section also describes an enhancement to this protocol to reduce the number of messages involved in the notification process. Figure 5.1 lists the pseudo code for the unicast-SFN protocol and Figure 5.2 shows an example of unicast-SFN signaling.

SFN is executed in response to a failure detected by node v_f . A *Failure* message is sent up the hierarchy to $parent_BB(v_f)$ for each failed path, P . The message contains the pid and ingress router of P . In general, a $parent_BB_k(v_f)$ that receives a *Failure* message determines if it governs $P.ingress$. If it does not, $parent_BB_k(v_f)$ forwards the *Failure* message up the hierarchy to $parent_BB_{k+1}(v_f)$. This is repeated up the hierarchy until $RootBB(P)$ is reached. $RootBB(P)$ sends a *FailureNotify* message down the hierarchy to $parent_BB_L(P.ingress)$.

In general, a $parent_BB_k(P.ingress)$ that receives a *FailureNotify* message forwards it down the hierarchy to notify $parent_BB_{k-1}(P.ingress)$ about the failure in P . This is repeated down the hierarchy until the *FailureNotify* message reaches $P.ingress$. This completes the notification process for this path.

```

Processing at  $v_k$  {
  While (1){
    Wait (msg){
      Switch (msg.type){
        Case Failure (Pid, P.ingress ) {
          If ( $v_k$  is  $parent\_BB(P.ingress)$ )
            Send (FailureNotify,  $parent\_BB(P.ingress)$ , pid) ;
          Else
            Send (Failure,  $parent\_BB(v_k)$ , FailedPlist $_k$ );
          }/*end case*/
        Case FailureNotify (P.ingress, pid ) {
          If ( $v_k$  is a physical node)
            Switchover & stop();
          Else
            Send (FailureNotify,  $parent\_BB(P.ingress)$ , pid);}
          }/* end case*/
        }/*end Switch*/
      }/*end while*/
    }/*end method*/
  
```

Figure 5.1: The unicast-SFN algorithm

Figure 5.2: Unicast-SFN signaling

As an example, Figure 5.2 shows a uni-directional path that is affected by a failure in the link connecting nodes 2.3.2 and 2.1.2. The path follows the thick lines at level 1. The ingress and egress nodes of the path are nodes 4.3.4 and 2.1.4, respectively. The master node for the failed link, node 2.3.2, sends *Failure* message #1 in Figure 5.2, to BB 2.3. Then, BB 2.3 forwards message #2 to BB 2. BB 2 in turn forwards message #3 to BB*. Since BB* has a view of the ingress router of the path, it sends a *FailureNotify* message #4 to BB 4. BB 4 notifies BB 4.3 with message #5 that in turn notifies node 4.3.4 of the failure with message #6.

The number of unicast-SFN notification messages required for each failed path will be $2R$. One extreme case is if the two nodes are in the same network domain then *RootBB* is at level 2 and the number of notification messages is 2. The other extreme is where *RootBB* coincides with the root of the network hierarchy. SFN fault notification messages are assumed to have higher priority over other signaling messages.

5.2.2. Enhanced Unicast-SFN

To enhance the performance of unicast-SFN, the *Failure* message is modified to contain a list, *Plist*, of failed paths on its way up the hierarchy. *Plist* contains the *pid* and ingress routers of paths affected by the failure. This decreases the number of messages a BB receives when compared to sending a *Failure* message for each failed path. However, it requires more processing to compose and decompose the *Plist* for each *FailureNotify* and *Failure* message, respectively. Figure 5.3 lists the pseudo code for the *Enhanced unicast-SFN* protocol while Figure 5.4 gives an example of the signaling of the protocol.

```

Processing at  $v_k$  {
  While (1){
    Wait (msg);
    Switch (msg.type){
      Case Failure (Plist) {
        For (each path  $P_g \in Plist$ ){
          If ( $v_k$  is parent_BB ( $P_g.ingress$ )) {
            Send (FailureNotify,  $P_g.ingress$ ,  $P_g$ );
            Remove  $P_g$  from Plist }
          }
        If (Plist != null)
          Send (Failure, parent_BB ( $v_k$ ), Plist);
        /*end case*/
      Case FailureNotify (FailedPlist $k+1$ ) {
        If ( $v_k$  is a physical node)
          Switchover & stop();
        Else
          Send (FailureNotify, parent_BB( $P.ingress$ ), pid);
        /* end case*/
      }/*end Switch*/
    }/*end while*/
  }/*end method*/

```

Figure 5.3: The enhanced SFN algorithm

Figure 5.4: Enhanced SFN signaling

A $parent_BB_k(v_f)$ that receives a *Failure* message scans *Plist*. It sends a *FailureNotify* message down the hierarchy to $parent_BB_{k-1}(P_g.ingress)$ if it governs $P_g.ingress$ where $P_g \in Plist$. The entry for P_g is removed from *Plist*. Then, $parent_BB_k(v_f)$ forwards a *Failure* message up the hierarchy to $parent_BB_{k+1}(v_f)$. This is repeated up the hierarchy until $RootBB(P)$ is reached. $RootBB(P)$ has a view of *all* ingress routers of the paths that belong to *Plist*. $RootBB(P)$ sends a *FailureNotify* message down the hierarchy to each $parent_BB_L(P_g.ingress)$ where $P_g \in Plist$.

A $parent_BB_k(P_g.ingress)$ that receives a *FailureNotify* message forwards it down the hierarchy to notify $parent_BB_{k-1}(P_g.ingress)$ about the failure in P_g . This is repeated down the hierarchy until the *FailureNotify* message reaches $P_g.ingress$. This completes the notification process for this path.

As an example, Figure 5.4 shows three uni-directional paths that are affected by a failure in the link connecting nodes 2.1.2 and 2.3.2. The paths follow the thick lines at level 1. The ingress nodes of the three paths are as follows: 4.3.4 for path 1, 2.3.1 for path 2 and 2.1.4 for path 3. The master node for the failed link, node 2.3.2, sends *Failure* message, #1 in Figure 5.4, to BB 2.3 that contains the *pid* of the failed paths. BB 2.3 scans the list of ingress nodes of failed paths looking for the nodes that it views. It sends a *FailureNotify* message #2 to node 2.3.1 about the failure in path 2 and removes the *pid* of path 2 from the list. As the list is not empty, BB 2.3 sends a *Failure* message #3 to BB 2. BB 2 finds the ingress node of path 3 to be within its view, so it sends *FailureNotify* message #4 to BB 2.1 and removes path 3 from the list. BB 2.1, in turn, sends a *FailureNotify* message #5 to physical node 2.1.4. Now, BB 2 realizes that some paths (i.e., 1) are out of its view, so it sends a *Failure* message #6 to BB*. Since BB* has a view of node 4.3.4, it sends a *FailureNotify* message #7 to BB 4. BB 4 notifies BB 4.3 that in turn notifies node 4.3.4 of the failure --message #8 and #9. BB* stops forwarding messages.

5.2.3. Multicast-SFN

To further reduce the size of fault notification signaling storm, SFN exploits the proximity of ingress nodes of failed paths. Multicast-SFN sends one notification message to ingress routers of failed paths that lie in the same network domain. Multicast-SFN pre-aggregates such paths into an *aggregate path*. Sending one notification message for an aggregate path instead of a message per failed path constitutes the reduction in signaling storm that multicast-SFN achieves. The pseudo code for multicast-SFN is shown in Figure 5.5 and an example of multicast-SFN signaling is given in Figure 5.6.

An aggregate path resembles an Aggregation Area (AA) [34] described above in Section 3.2.3. However, AA are generally used to reduce the soft state maintained at network nodes about non-

aggregated paths. Only the ingress and egress routers of an AA maintain information about non-aggregated paths. Path aggregation is implemented at all levels of the network hierarchy. Signaling aggregation can be performed based on path information maintained in network hierarchy and created by H-DRA or by HDR. For example, Figure 5.6 shows 4 paths following the thick lines at level 1. The ingress and egress nodes of the three paths are as follows: 4.3.4 and 2.2.1 for path 1, 4.3.2 and 2.2.1 for path 2, 4.2.1 and 2.2.1 for path 3, and 2.1.5 and 1.4.1 for path 4. Paths 1 and 2 are aggregated into $AggP_2$ at level 2 because they have the same ingress router at level 1. $AggP_2$ BB 4.3 has its ingress router at level 2. $AggP_2$ and path 3 have the same ingress router at level 2, so they are aggregated into path $AggP_3$.

The multicast-SFN protocol is started when v_f detects a network failure and sends a *Failure* message to $parent_BB_k(v_f)$. $parent_BB_k(v_f)$ applies the *Enhanced unicast-SFN* algorithm described above with an extra step: $parent_BB_k(v_f)$ composes a new list, $Plist_k$, of failed paths replacing non-aggregated paths with their aggregate path. If a path is not to be aggregated with other paths, it is added to the list as a non-aggregated path. Then, $parent_BB_k(v_f)$ forwards the *Failure* message to $parent_BB_{k+1}(v_f)$. This is repeated up the hierarchy until *RootBB* has been notified of the failure.

RootBB sends *FailureNotify* messages down the hierarchy, as described in Section 5.2.1, to all BB_k that have views of the ingress routers of aggregate and non-aggregated paths in the message. In case of a non-aggregated path, P , BB_k sends only one *FailureNotify* message down the hierarchy to a BB at the next lower level of the hierarchy that has a view of $P.ingress$. In case of an aggregate path, $AggP_k$, BB_k sends a *FailureNotify* message to each of its children BB, or nodes, that has a view of $P_g.ingress$ where $P_g \in AggP_k$. This is repeated down the hierarchy until all the ingress routers have been notified of the failure.


```

Processing at  $v_k$ {
While (1) {
  Wait (msg);
  Switch (msg.type){
    Case Failure ( $P_{list_{k-1}}$ ) {
      Initialize  $P_{list_k} = \text{null}$ ;
      For (each path  $P_g \in P_{list_{k-1}}$ ){
        If ( $v_k$  is parent_BB ( $P_g.ingress$ ))
          Send (FailureNotify,  $P_g.ingress$ ,  $P_g$ );
        Else{
          If ( $P_g \notin AggP_h$ )  $P_{list_k} += \{P_g\}$ 
          Else If ( $P_g \in AggP_h$ ) {
            If ( $AggP_h \notin P_{list_k}$ )  $P_{list_k} += \{AggP_h\}$ ;
             $P_{list_k} \rightarrow AggP_h += P_g$ 
          }
        }
      }
      // end for
      If ( $P_{list} \neq \text{null}$ )
        Send (Failure, parent_BB ( $v_k$ ),  $P_{list_k}$ );
    }/*end case*/
    Case FailureNotify ( $P_g$ ) {
      If ( $v_k$  is a physical node)
        Switchover & stop();
      If ( $P_g$  is an Aggregate path)
        For (each  $P_j \in P_g$ )
          Send (FailureNotify, parent_BB $k-1$  ( $P_j.ingress$ ),  $P_j$ )
        Else // not aggregate path
          Send (FailureNotify, parent_BB $k-1$  ( $P_g.ingress$ ),  $P_g$ );
    }/*end case*/
  }/*end Switch*/
}/*end while*/
}/*end method*/

```

Figure 5.5: The multicast-SFN algorithm

Figure 5.6: Multicast-SFN signaling example

For the example network hierarchy shown in Figure 5.6, as the physical link connecting node 2.1.2 and node 2.3.2 fails, node 2.1.2 sends a *Failure* message #1, in the figure, to BB 2.1. BB 2.1 realizes that the ingress for path 4 is within its view. So, it sends a *FailureNotify* message #2 to the ingress of that path, node 2.1.5, and removes its entry from the list. BB 2.1 also realizes that paths 1 and 2 are aggregated into $AggP_2$, and replaces the entries for paths 1 and 2 with a single entry for $AggP_2$. Then, it sends a *Failure* message #3 to BB 2 with the new list of failed paths. BB 2 finds it has no view of ingress routers of the paths in the list. BB 2 realizes that $AggP_2$ and path 3 are aggregated into $AggP_3$, and replaces their *pid* with the *pid* of $AggP_3$. Then, it sends a *Failure* message #4 to BB*. BB* has a view of the ingress routers of the aggregate path. It sends a *FailureNotify* message #5 to BB 4 that views the ingress nodes of $AggP_3$. BB 4 decomposes the aggregate path into its component paths: $AggP_2$ and path 3. It sends *FailureNotify* messages to BB 4.2 about the failure in path 3 and to BB 4.3 about the failure in $AggP_2$. BB 4.2 sends a *FailureNotify* message to node 4.2.1 about the failure in path #3. BB 4.3

notifies the ingress routers of the different component paths of $AggP_2$. That is, it sends a *FailureNotify* messages to nodes 4.3.4 and 4.3.2. This completes the failure notification process.

5.3. Evaluation of the Performance of SFN

In this section, the performance of the hierarchical fault notification approaches, namely unicast-SFN and multicast-SFN is analyzed and compared to the performance of non-hierarchical approached, namely *standard unicast* and RNT which are described in Section 2.4.2. The performance of unicast-SFN is compared to that of *standard unicast* and the performance of multicast-SFN is compared to that of RNT. The performance of all these protocols is characterized in terms of the approximate size of signaling storms generated in response to a network failure. Table 5.1 illustrates this dichotomy of the protocols.

Table 5.1: Fault notification protocols studied in this chapter

	Unicast-based	Multicast-based
Non-hierarchical	Standard Unicast	RNT
Hierarchical	Unicast SFN	Multicast_SF

The analysis of the four protocols is based on Equations 5.1 and 5.2 below. Equation 5.1 characterizes the storm size for unicast-based protocols, which is given by:

$$S_{unicast} \approx 2 \mu_{unicast} F \quad (5.1)$$

where: $S_{unicast}$ represents the approximate number of messages in the signaling storm for unicast-based protocols;

- $\mu_{unicast}$ is the average path length between any two nodes in the network; and,
- F represents the number of paths affected by a failure.

Equation 5.2 characterizes how multicast-based approaches, namely multicast-SFN and RNT, take advantage of the savings multicast achieves. The equation is a power-law relationship [44] that relates the total number of multicast links with the average length of a path followed by a notification message and with the number of ingress routers. The size of the signaling storm with these approaches is given by:

$$S_{multicast} \approx m^k \mu_{multicast} \quad (5.2)$$

where: $S_{multicast}$ is the number of multicast links in a multicast tree which represents the approximate number of messages in the signaling storm with multicast-based approaches;

- $\mu_{unicast}$ is the average length of a path followed by a notification message;
- m is the number of receivers in the tree which represents the number of ingress routers to be notified; and,
- k was found to have a value of 0.8 for a number of real life and generated topologies [63].

For the non-hierarchical approaches, i.e., *standard unicast* and RNT, μ is estimated as the effective diameter of the network. This can be defined as the average number of hops between any two nodes in the network [44]. The effective diameter of an N -node network is given, with a probability of about 80%, by:

$$\mu_{non-hierarchical} \approx \left(\frac{N^2}{N + 2E(N)} \right)^{1/H} \quad (5.3)$$

Where:

- H is the hop-plot exponent for a graph [44]. For real life network topologies, the average value for H has been shown to be 4.7 [44]; and,

- $E(N)$ is the number of edges in such a graph which can be estimated by [63]:

$$E(N) = \frac{N}{2(t+1)} \left(1 - \frac{1}{N^{t+1}} \right) \quad (5.4)$$

where t is chosen to equal -0.80 which is the value found for many Internet topologies [63].

Substituting from (5.4) into (5.3) with the values given above, μ can be approximated by:

$$\mu_{non-hierarchical} \approx \left(\frac{0.2 N}{1.2 - N^{-0.2}} \right)^{1/4.7} \quad (5.5)$$

Thus, the approximate size of the signaling storm of the *standard unicast* approach is given by Equation 5.6.

$$S_{standard-unicast} \approx 2 F \left(\frac{0.2 N}{1.2 - N^{-0.2}} \right)^{1/4.7} \quad (5.6)$$

The approximate size of RNT storm, with μ is given by Equation 5.5, can be estimated by:

$$S_{RNT} \approx m^{0.8} \left(\frac{0.2 N}{1.2 - N^{-0.2}} \right)^{1/4.7} \quad (5.7)$$

For the hierarchical approaches, i.e., the SFN ones, the approximate size of the signaling storm is a function of the number of levels in the hierarchy. Unicast-SFN signals up and down the hierarchy for each non-aggregated path. Then, the maximum value for μ is $2L$, where L is the number of levels in the hierarchy. Thus, the approximate size of the signaling storm in case of unicast-SFN is given by:

$$S_{unicast-SFN} \approx 2LF \quad (5.8)$$

Multicast-SFN builds a number of multicast trees to perform fault notification. Each multicast tree is rooted at a BB that has a view of the ingress router of paths contained in the list. The maximum value for μ is L , where L is the number of levels in the network hierarchy. Thus, the approximate size of the signaling storm in case of multicast-SFN is given by:

$$S_{unicast-SFN} \approx m^{0.8}L \quad (5.9)$$

The approximate sizes of the signaling storm generated by the four protocols are summarized in Table 5.2. Using hierarchical networks and multicast for performing fault notification signaling is clearly advantageous. The size of the signaling storm with non-hierarchical approaches, *standard unicast* and RNT depends on network size. With hierarchical approaches, the size of the signaling storm is a function of the number of levels in the network hierarchy. With unicast approaches, the size of the signaling storm depends on the number of failed paths, F . With multicast approaches, it depends on the number of ingress nodes to be notified of failure, m . Typically the use of multicast would represent a significant reduction in signaling storm size.

Table 5.2: Approximate sizes of signaling storm for the protocols studies

Fault Notification Protocol	Approximate Size of Signaling Storm
<i>Standard Unicast</i>	$2F \left(\frac{0.2N}{1.2 - N^{-0.2}} \right)^{1/4.7}$
RNT	$m^{0.8} \left(\frac{0.2N}{1.2 - N^{-0.2}} \right)^{1/4.7}$
Unicast_SFN	$2LF$
Multicast_SFN	$m^{0.8}L$

5.3.1. Experimental Design

To further evaluate SFN behavior, this section describes simulation experiments that study:

- the performance and scalability of SFN in reducing signaling storm size when compared to other protocols for different network sizes;
- the effect of the structure of network hierarchy on SFN performance; and,
- the overhead of SFN.

The main performance metrics of interest for our study are:

1. the size of the fault notification signaling storm in response to a network failure; and,
2. the amount of storage for path state information maintained in the network hierarchy.

The size of the fault notification signaling storm is defined as the number of signaling messages generated to notify the ingress routers of the failed paths about the failure. This metric is used to study the performance and scalability of SFN as the network size grows and for different structures of the network hierarchy. This metric is affected by the number of failed paths and the number of hops between the failure and the ingress router of each failed path. The size of signaling storm reported in this section is the average of the signaling storm sizes over the replications of the experiments. Confidence intervals are illustrated for a 90% confidence level.

The amount of storage for path state information maintained in the network hierarchy is used to characterize the overhead of SFN and the h -protocols. The number of BB that maintain information about a path expresses this overhead. The exact amount of information depends on the data structures used to store the information.

The OMNet++ simulator [71], described in Section 4.4, was used to implement simulation models for different network topologies and for different hierarchy structures. Networks of 256-, 1296- and 4096- nodes were considered and hierarchies of 3 and 5 levels. The different structures for the network hierarchies are described in Table 4.2 in Section 4.4.

For these experiments, path routing requests arrive following a *Poisson* distribution with a mean inter-arrival time of 50 ms. The source and destination nodes of each path are chosen randomly. 2000 requests are generated for each simulation run. The *h*-protocols are used to create the requested paths. Information about created paths is maintained at BB.

At the end of each run, 15 network failures were generated and injected into the simulation model to exercise the protocol. The failed nodes are selected randomly. The *standard unicast*, unicast-SFN and multicast-SFN protocols are deployed to notify ingress routers of the failure. RNT was not implemented. The number of SFN messages caused by each failure is counted. Eight simulations runs are carried out and their results are averaged and reported for 90% confidence intervals.

The simulation model was designed to capture only the size of signaling storms. The model does not include features to characterize the computational processing associated with signaling.

5.3.2. SFN Performance

This section presents the results of the simulation experiments. These results are divided into two sets. The first set characterizes the size of signaling storm generated in response to a network failure for different network sizes. The second set of results characterizes the storage overhead associated with SFN and the *h*-protocols.

Figures 5.7– 5.11 show the size of the fault notification signaling storm, along the y-axis, for *standard unicast*, unicast-SFN and multicast-SFN. The storm size is shown in terms of the number of failed paths which is shown along the x-axis of the figures.

To study the performance of the protocols, the sizes of the signaling storm for each protocol are compared. Tables 5.3 and 5.4 summarize the comparison for the 256- and 4096 node networks for 300 and 700 affected paths. From the tables, using hierarchical network and signaling aggregation with multicast techniques reduces the size of the signaling storm significantly. The conclusions of this comparison comply with those of Table 5.2.

To study the scalability of SFN, the increase in the size of the signaling storm is quantified as network size grows. From Tables 5.3 and 5.4, a sub-linear increase in the size of the signaling storm of both unicast-SFN and multicast-SFN is required as the network size grows from 256 to 4096 nodes. The increase in the size of the signaling storm is sub-linear and relatively small when compared to that of *standard unicast*. This conclusion applies for hierarchies of 3-levels and 5 levels and proves SFN to be a scalable protocol for failure notification.

To study the benefit of using multicast for SFN notification, the size of the signaling storm with unicast_SFN is compared to that size when deploying multicast-SFN. From the figures and Tables 5.3 and 5.4, multicast-SFN results in a significantly smaller size for signaling storms.

The effect of the structure of network hierarchy on the performance of the two versions of SFN is shown in the figures. The deeper the hierarchy, the larger the size of the signaling storm for the same network size. This is because a deeper hierarchy results in a smaller domain size and consequently the ingress routers of the failed paths will be more sparingly distributed among network domains.

Figure 5.7: Average size of *standard unicast* signaling storm

Figure 5.8: Average size of unicast-SFN signaling storm for 3-level network hierarchies

Figure 5.9: Average size of unicast-SFN signaling storm for 5-level network hierarchies

Figure 5.10: Average size of multicast-SFN signaling storm for 3-level network hierarchies

Figure 5.11: Average size of multicast-SFN signaling storm for 5-level network hierarchies

Table 5.3: Average sizes of signaling storm a 256-node network for the protocols studied

		H_{3-16}	H_{3-16}	H_{5-4}	H_{5-4}
paths	Standard_unicast	Multicast_SFN	Unicast_SFN	Multicast_SFN	Unicast_SFN
300	4300	790	1900	1310	2300
700	12400	1837	5692	3144	6400

Table 5.4: Average sizes of signaling storm a 4096-node network for the protocols studied

		H_{3-64}	H_{3-64}	H_{5-8}	H_{5-8}
paths	Standard_unicast	Multicast_SFN	Unicast_SFN	Multicast_SFN	Unicast_SFN
300	9800	1100	4300	1860	6000
700	23800	2390	10300	4254	16200

The second set of simulation results characterize the information storage overhead associated with SFN and the h -protocols. The amount of this information depends on the type of data structure used to store the information and the number of BB involved in maintaining information about a path. Since the type of a data structure is an implementation issue, this overhead is expressed in terms of the number of BB managing a path. Figures 5.12 and 5.13 show this, along the y-axis, in terms of path length which shown along the x-axis.

Another reason for choosing the number of BB managing a path to express information storage requirements of the h -protocols is that different protocols may require different information to be maintained at each BB about the paths created. This is again an implementation issue. For example, multicast SFN requires the aggregation of information about aggregate paths while unicast_SFN does not have this requirement. Also, HDR-P requires more information stored at each BB to enable parallelism. This is not required by HDR. Hence, the number of BB managing a path is used to gain insights on how the different experimental factors may affect information storage.

Network size affects the information storage overhead of the h -protocols. The larger the network, the larger the number of domains crossed by a path, and the larger the number of BB managing this path. For example, a path of 70 nodes is managed by an average of 12.8, 18.9 and 28.0 BB with 3-level hierarchies for the 256-, 1296- and 4096-node networks, respectively.

The length of the path affects the number of BB managing that path. The longer the path, the larger the number of BB managing it. For example, to manage a path of 20 nodes, H_{3-36} requires an average of 6.7 BB while the same hierarchy requires 12.1 BB to manage a path of 60 nodes.

The figures also show the effect of different hierarchy structures on the number of BB needed to manage a path. The deeper the hierarchy, the more the storage information overhead needed. For

example, a path of 30 nodes is managed by an average of 8.1 BB in H_{3-16} and of 30.6 BB in H_{5-4} . This is because a deeper hierarchy generally results in a smaller domain size and consequently the ingress routers of the failed paths will be more sparingly distributed among network domains.

Figure 5.12: Average number of BB managing a path for 3-level network hierarchies

Figure 5.13 Average number of BB managing a path for 5-level network hierarchies

5.4. Concluding Remarks

The new Scalable Fault Notification (SFN) protocol results in smaller signaling storms for larger networks. In SFN, signaling messages travel vertically up and down a network hierarchy instead of horizontally along service routes. This reduces the number of signaling messages to notify the ingress router of the path. SFN extends the h -protocols to handle fault notification. Note that Peer Group Leaders, in PNLI, do not currently participate in such fault notification.

The analysis and simulation presented in this chapter shows SFN to be a scalable technique for failure notification. The protocol provides better performance as compared with non-hierarchical approaches. Aggregating signaling messages takes advantage of the savings multicast achieves as compared to the unicast case. However, this comes at the non-trivial cost of implementing multicast.

The hierarchical network structure affects the size of the signaling storm. The deeper the hierarchy, the greater the number of signaling messages for the same network size and for the same service workload. This increases fault notification overhead and latency but offers advantages for inter-domain routing and TE.

The advantages of SFN come at the cost of increased control overhead required in the network hierarchy. The protocol uses path management information maintained in the network hierarchy to aggregate messages so that a single notification message would be raised per path aggregate instead of a number of them for each path.

Chapter 6: Conclusions and Future Work

6.1. Conclusions

This chapter provides a summary of the conclusions for this thesis followed by a description of future work. This thesis proposes several scalable service routing and fault notification protocols, referred to as the *h*-protocols. The protocols exploit scalability features of hierarchical networks. Nodes at higher levels in a network hierarchy participate in making end-to-end routing decisions, maintain information about services created and participate in fault notification to recover failed services.

The protocols are: 1) the Hierarchical-Distributed Routing Algorithm (H-DRA) that uses flooding-based routing algorithms to perform intra-domain routing; 2) the Hierarchical Distributed Routing (HDR) protocol where intra-domain routing computations are carried out in series; 3) the in-parallel Hierarchical Distributed Routing (*HDR-P*) protocol where intra-domain routing calculations are done in parallel in order to reduce service routing time; and 4) the Scalable Fault Notification (SFN) protocol where fault notification signals travel up and down the hierarchy, instead of along service routes. The operations of these protocols and their performance results are presented in Chapters 4 and 5. The conclusions of this thesis are:

1- The proposed service routing protocols avoid network congestions and distributes data workloads among inter domain links. The protocols calculate inter-domain end-to-end routes by selecting potential network domains and links to be traversed by services. The selection is controlled in order to load-balance traffic workloads among network domains and links.

Simulation results show that the *h*-protocols balance data traffic loads across inter-domain links better than PNNI. With the *h*-protocols for the networks considered few inter domain links have either very high, (i.e., >80%), or very low, (i.e., <20%) utilization. PNNI was found to cause network congestions with a large number of inter-domain links with utilizations larger than 80%. This complies with the observations about current routing protocols such as PNNI and OSPF published in [37] [38].

An unbalanced distribution of data workloads (i.e., a large number of links have high utilization while others have low utilization), may reduce the admission percentage of requested routes. The *h*-protocols have a higher admission percentage than PNNI. A detailed discussion of the admission percentage of the *h*-protocols versus PNNI is given in Section 4.4.5. However, the *h*-protocols have a higher message complexity than PNNI. This is due to the extra signaling performed at different hierarchy levels with the *h*-protocols to enable TE. A discussion of the overheads of the *h*-protocols is given in Section 6.2.

2- Flooding-based routing algorithms can be used in large networks to create end-to-end services without maintaining global network state. H-DRA uses flooding-based distributed routing algorithms to select network domains to be traversed by services. Using these algorithms increases the admission percentage of routing protocols. Simulation results conducted for different network sizes described in Section 4.4.5, show that H-DRA has the highest admission percentage compared to protocols such as HDR and PNNI that rely on centralized intra-domain routing.

3- The *h*-protocols provide network control scalability as network size grows. A relatively small increase in computational or communication overhead is required to create the same-length path as the network size increases from 256 to 1296 and to 4096 nodes.

HDRA requires small increases in the number of messages compared to increases in network size. With HDR and HDR-P a small increase in path setup time is incurred to create a path as the network size grows as described above. The *h*-protocols exploit hierarchical network structures to achieve this scalability. Section 4.2 describes how the *h*-protocols achieve this effect and Section 4.4 presents simulation results that illustrate it.

4- The structure of network hierarchy affects the performance of the *h*-protocols. A deeper hierarchy results in reducing HDRA communication overhead required to create a path for the same physical network. A smaller domain size reduces the communication overhead of DRA. However, deeper hierarchies result in larger HDRA setup times. This is because of the time required for BB to exchange signaling messages at higher levels of the hierarchy.

Deeper hierarchies result in smaller HDR/ HDR-P setup times but more HDR/ HDR-P communication overhead. A smaller domain size reduces the route computation time of a centralized routing algorithm. However, this implies that a larger number of domains are involved in creating the path, which results in higher communication overhead.

However, deeper hierarchies generally have a lower route admission percentage. The accuracy of information maintained by BB at different levels in the hierarchy affect routing decisions made at lower levels. The higher the level of a BB, the more inaccurate information it may have. Also, the utilization of inter-domain links at higher levels in the hierarchy is based on aggregating the utilization and capacity of lower levels inter-domain links. The accuracy of inter-domain link aggregation affects the accuracy of information maintained at higher hierarchy levels.

5- SFN reduces the size of the signaling storm when compared with non-hierarchical approaches and limits the growth in the size of the signaling storm as network size increases. SFN also alleviates the number of fault notification signals by aggregating them.

Aggregation is based on service creation information maintained at BB and created by the service routing protocols. Signaling Aggregation takes advantage of the savings multicast achieves. Using multicast is a price to pay for faster fault notification. This is not exclusive to SFN; for instance, RNT [12] described in Section 2.4.2 also depends on multicast.

With SFN, only a relatively a small increase in the size of the signaling storm is incurred compared with the increase with the unicast approach for the same network size and the same number of affected paths.

Complexity analysis and simulation results show that the performance of SFN is a function of the number of levels in the hierarchy while performance of other fault notification protocols depends on network size. SFN has much smaller average distance between two nodes in the multicast tree than RNT.

6.2. Discussion of the Overhead of *H*-protocols

In general, the advantages of the *h*-protocols come at the cost of increased control overhead required in the network hierarchy when compared to PNNI. However, PNNI was not designed to avoid congestions and does not target service survivability. In PNNI, hierarchical nodes do not take routing decisions nor do they participate in service recovery. They only abstract and distribute link state information to their peer nodes.

The *h*-protocols require BB to maintain information about the domain it is managing and inter-domain links. The amount of information stored at each BB depends mainly on the number of nodes in a domain and the aggregation scheme used.

For HDR and HDR-P, BB at higher hierarchy levels may be, depending on the workload, subject to a higher average computation overhead than BB at lower levels. This is because the number of

BB at the lower level, say L , would be larger than the number of BB at level $L+1$. However, higher-level BB can be off-loaded to server clusters.

The h -protocols can be used to create and support the management of high bandwidth services. These services contribute with a large percentage to data traffic. Analysis of real life inter-domain data traffic for different service providers indicates that about 10% of the total services were responsible for about 90% of the bytes transmitted [87].

6.3. Future Work

This thesis discussed technical aspects of hierarchical network control. The h -protocols establish and restore end-to-end point-to-point paths. The protocols can be extended in the future to provision services with point-to-multipoint and multipoint-to-multipoint topologies.

The protocols avoid highly-utilized inter-domain links. New ways to characterize the utilization of a domain as a whole will be the focus of future research. This may help in avoiding domains with high utilizations and can result in evenly distributing data traffic workloads among domains.

To increase route admission percentage for the h -protocols, the protocols must be extended to perform crankback. This should be done with no increase in either the overhead or the complexity of the h -protocols.

The complexity and simulation-based analysis of the protocols was based on a uniform hierarchical structure. It would greatly enhance the conclusions of this thesis to study hierarchies with more realistic structure, for example, for some portion of the Internet. It will also be of importance to study how to integrate the h -protocols with protocols such as BGRP, described in Section 2.2.2, in order to achieve scalability of state maintained in the hierarchy. It would also enhance these conclusions to use real life workloads to study the performance of the h -protocols.

The major focus of this thesis is to study how the *h*-protocols load-balance data workloads among inter-domain links and the performance these protocols. For full deployment of the *h*-protocols in real life networks, administrative issues associated with these protocols must be studied. This includes trust and security management between owners of different network domains. It also includes studying billing, charging and service level agreement management between these providers.

The *h*-protocols can be generalized to provide wide-area resource management for computer systems. The *h*-protocols are concerned only with one resource: network bandwidth. Extending the *h*-protocols to consider other resources such as CPU time, storage memory, storage disks and other computational resources will enable these protocols to be general purpose wide-area resource management protocols. For example, current Grid computing systems rely on a centralized server, a.k.a., meta-scheduler [97], to allocate resources across the Grid. This server can be a performance bottleneck and a single point of failure as the size of the Grid increases. A generalized version of the *h*-protocols can provide scalable Grid resource management.

Appendix A: List of Symbols

- src, dst : the source and destination nodes of a requested route;
- pid : an identification number that uniquely identifies a path;
- v_k : a level k node/ BB, physical nodes have $k=1$ and BB have $k > 1$;
- $parent_BB_k(v)$: a level k BB that governs node v ;
- P_k : a level k path that traverses a number of level k BB/ nodes;
- $P_k.ingress$: the ingress node for P_k ;
- $P_k.prev$ and $P_k.next$: the nodes preceding and following the current node along P_k , respectively;
- $RootBB(src, dst)$: the lowest-level BB that governs both src and dst . It is assumed to be at level $R+1$ (i.e., it is $parent_BB_{R+1}(src)$ and $parent_BB_{R+1}(dst)$).
- v_f : a level 1 node that detects the failure;
- P : a path that is affected by the failure;
- $Plist_k$: a list of pid of affected paths that is composed by a level k BB and sent in a *Failure* message to a level $k+1$ BB; and,
- $AggP_k$: a level k aggregate path that may aggregate a number of level $k-1$ aggregate paths.

Appendix B: List of Acronyms

AA	Aggregation Areas
ATM	Asynchronous Transfer Mode
AS	Autonomous Systems
BB	Bandwidth Broker
BGRP	Border Gateway Reservation Protocol
ER	Explicit Route
FR	Frame Relay
IntServ	Integrated Services Architecture
TE	Internet Traffic Engineering
LSR	Label Switched Routers
LQM	Layered Queuing Model
PGL	Peer Group Leader
PNNI	Private Network-to-Network Interface
QoS	Quality of Service
RSVP	Resource reSerVation Protocol
RNT	Reverse Notification Trees
SFN	Scalable Fault Notification protocol

SLA	Service Level Agreement
SIBBS	Simple Inter-BB Signaling
BGP	The Border Gateway Protocol
GMPLS	The Generalized Multi-Protocol Label Switching
HDR	The Hierarchical Distributed Routing
H-DRA	The Hierarchical-Distributed Routing Algorithm
HDR-P	The in-Parallel Hierarchical Distributed Routing
RSVP-TE	Traffic Engineering extensions to RSVP
WDM	Wavelength Division Multiplexing

References

- [1.] "B-ISDN Service Aspects," ITU-T Recommendation I.211.
- [2.] "Terms and Definitions related to the Quality of Telecommunications Services," ITU-T Recommendation E.800.
- [3.] "Private Network-Network Interface specification," (PNNI 1.0), ATM Forum Specification <af-pnni-0055.000>, March, 1996.
- [4.] A. Lazar, "Programming Telecommunication Networks," IEEE Network Magazine, vol. 11, no. 5, pp: 8-18, September, 1997.
- [5.] A. Terzis, L. Wang, J. Ogawa and L. Zhang, "A Two-Tier Resource Management Model for the Internet," in Proceedings of the Global Internet Conference, December, 1999. <http://irl.cs.ucla.edu/~andreas/>
- [6.] B. Awerbuch, Y. Du and Y. Shavitt, "The Effect of Network Hierarchy Structure on Performance of ATM PNNI Hierarchical Routing," in Proceedings of the International Conference on Computer Communications and Networks, p. 73, USA, October, 1998.
- [7.] B. Awerbuch, Y. Du, B. Khan and Y. Shavitt, "Routing Through Networks with Hierarchical Topology Aggregation," Journal of High Speed Networks, vol. 7, no.1, pp: 57-73, November, 1998.
- [8.] B. Fortz and M. Thorup, "Internet Traffic Engineering By Optimizing OSPF Weights," in Proceedings of IEEE INFOCOM, pp: 519-528, Tel-Aviv, March, 2000.
- [9.] B. Rajagopalan, D. Saha, G. Bernstein and V. Sharma, "Signaling for Fast Restoration in Heterogeneous Optical Mesh Networks," in Proceedings of SPIE Asia Pacific Conference on Optical Communication (APOC), China, November, 2001.
- [10.] A. Elwalid, C. Jin, S. Low and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," in Proceedings of IEEE INFOCOM, USA, April, 2001.

- [11.] C. N. Chuah, L. Subramanian, R. H. Katz and A. D. Joseph, "QoS Provisioning Using A Clearing House Architecture," in Proceedings of the 8th International IEEE Workshop on Quality of Service (IWQoS), pp: 115-124, USA, June, 2000.
- [12.] C. Huang, V. Sharma, K. Owens and S. Makam, "Building Reliable MPLS Networks Using a Path Protection Mechanism," IEEE Communications Magazine, vol. 40, no. 3, pp: 156–162, March, 2002.
- [13.] C. M. Woodside, "Performability Modeling for Multi-Layered Service Systems", in Proceedings of the 3rd Workshop on Performability of Computer and Communications Systems, USA, 1996.
- [14.] L. Cormen, "Introduction to Algorithms," MIT Press, USA, 1999, ISBN 0-2620-3293-7.
- [15.] D. Awduche, "MPLS and Traffic Engineering in IP Networks," IEEE Communications Magazine, vol. 37, no. 12, pp: 42-47, December, 1999.
- [16.] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja and X. Xiao, "Overview and Principles of Internet Traffic Engineering," IETF RFC 3272.
- [17.] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus, "Requirements for Traffic Engineering Over MPLS," IETF RFC 2702.
- [18.] D. Awduche, L. Berger, D. Gan, T. Li and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," IETF RFC 3209.
- [19.] D. Awduche, T. Li and G. Swallow, "IGP Requirements for Traffic Engineering with MPLS," IETF RFC 2702.
- [20.] D. Caswell and S. Ramanathan, "Using Service Models for Management of Internet Services," IEEE Journal on Selected Areas in Communications, vol. 18, no. 5, 2000.
- [21.] G. Huston, "Commentary on Inter-Domain Routing in the Internet," IETF RFC 3221.
- [22.] D. Basak, H. T. Kaur and S. Kalyan, "Traffic Engineering Techniques and Algorithms for the Internet," RPI Tech Report, www.ecse.rpi.edu/Homepages/shivkuma/

- [23.] D. Ghosh, V. Sarangan and R. Acharya, "Quality of Service Routing in IP Networks," IEEE Transactions on Multimedia, vol. 3, no. 2, pp: 200-208, June, 2001.
- [24.] G. Bernstein, B. Rajagopalan and D. Saha "Optical Network Control: Architecture, Protocols and Standards" Addison Wesley, 2003, ISBN: 0201753014.
- [25.] D. Reeves and H. Salama, "A Distributed Algorithm for Delay-Constrained Unicast Routing," IEEE/ ACM Transactions on Networking, vol. 8, no. 2, pp: 239-250, April, 2000.
- [26.] D. Tennenhouse, J. Smith, W. Sincoskie, D Wetherall and G. Minden. "A Survey of Active Network Research," IEEE Communications Magazine, vol. 35, no. 1, pp: 80-86, January, 1997.
- [27.] D. A. Villela, A. T. Campbell and J. Vicente, "Virtuosity: Programmable Resource Management for Spawning Networks", Computer Networks, Special Issue on Active Networks, vol. 36, no. 1, pp: 49-73, June, 2001.
- [28.] A. Feldman, A. Greenberg, C. Lund, N. Reingold and J. Rexford, "NetScope: Traffic Engineering for IP Networks," IEEE Network Magazine, vol. 14, no. 2, pp: 11-19, March, 2000.
- [29.] D. Wetherall, U. Legedza and J. Guttag, "Introducing New Internet Services, Why and How?" IEEE Network Magazine, vol. 12, no. 3, pp: 12-19, June, 1998.
- [30.] E. Basturk, A. Birman, G. Delp, R. Guérin, R. Haas, S. Kamat D. Kandlur, P. Pan, D. Pendarakis, R. Rajan, D. Saha and D. Williams, "Design and Implementation of a QoS Capable Switch-Router," Computer Networks, vol. 31, no. 1-2, pp: 17-30, January, 1999.
- [31.] B. Fortz and M. Thorup, "Optimizing OSPF/ IS-IS Weights in a Changing World," IEEE Journal on Selected Areas in Communications, vol. 20, no.4, pp: 756-767, 2002.
- [32.] E. Felstine and R. Cohen, "On the Distribution of Routing Computation in Hierarchical ATM Networks," IEEE Transactions on Networking, vol. 7, no. 6, pp: 906-916, December, 1999.
- [33.] E. Rosen, A. Viswan and R. Callon, "Multi-protocol Label Switching Architecture," IETF RFC 3031.

- [34.] F. Baker, C. Iturralde, F. Faucheur and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 reservations," IETF RFC 3175.
- [35.] G. Alaettinoglu and A. Shankar, "The Viewserver Hierarchy for Inter-domain Routing Protocols," IEEE Journal on Selected Areas in Communications, vol. 13, no. 8, pp: 1396-1410, 1995.
- [36.] G. Apostolopoulos, R. Guerin, S. Kamat and A. Orda, "Intra-Domain QoS Routing in IP Networks, A Feasibility and Cost/Benefit Analysis," IEEE Network Magazine, vol. 13, no. 5, pp: 42-53, September, 1999.
- [37.] G. Ash, "Traffic Engineering & QoS Methods for IP-, ATM- & TDM- Based Multi-service Networks," IETF Internet Draft, work in progress, <draft-ietf-tewg-qos-routing-04.txt>, April, 2002.
- [38.] G. Ash, B. Jamoussi, Y. Lee and O. Aboul-Magd, "QoS resource management in MPLS-Based networks," IETF Internet Draft, work in progress, <draft-ashqos-routing-00.txt>, March, 1999.
- [39.] G. Ash, G. L. Choudhury, J. Han, D. Sapozhnikova, M. Sherif, M. Noorchashm and A. Maunder, "Proposed Mechanisms for Congestion Control/ Failure Recovery in OSPF & ISIS Networks," IETF Internet Draft, work in progress, June, 2002.
- [40.] X. Yang and B. Ramamurthy, "Inter-Domain Dynamic Routing in Multi-Layer Optical Transport Networks," in Proceedings of IEEE GLOBECOM'03, USA, December, 2003.
- [41.] J. Stewart, "BGP-4 Inter-domain routing in the Internet," ISBN 0-201-37951-1, Addison Wesley, 1999.
- [42.] G. Li, J. Yates, R. Doverspike and D. Wang, "Experiments in Fast Restoration using GMPLS in Optical/ Electronic Mesh Networks," in Proceedings of the Optical Fiber Communications Conference, March, 2001.
- [43.] I. Castineyra, N. Chiappa and M. Steenstrup, "The Nimrod Routing Architecture," IETF RFC 1992.

- [44.] J. Chuang and M. Sirbu "Pricing Multicast Communications: A Cost-Based Approach," in Proceedings of the Internet Society INET'98 Conference, Switzerland, July, 1998.
- [45.] J. Gao, P. Steenkiste, and A. Fisher, "A Programmable Router Architecture Supporting Control Plane Extensibility," IEEE Communications Magazine, vol. 14, no. 2, pp: 152-159, March, 2000.
- [46.] J. Hura and A. Lazar, "A Programmable Transport Architecture with QoS Guarantees," IEEE Communications Magazine, vol. 12, no. 2, pp: 54-62, October, 1998.
- [47.] J. Rexford, J. Hall and K. Shin, "A Router Architecture for Real-Time Communication in Multi-computer Networks," IEEE Transactions on Computers, vol. 47, no. 10, pp: 1088-1101, October 1998.
- [48.] J. Rolia, "Predicting software performance," Ph.D. Dissertation, University of Toronto, Canada, 1992.
- [49.] J. Rolia and K.C. Sevcik, "The Method of Layers," IEEE Transactions on Software Engineering, vol. 21, no. 8, pp: 689-700, August, 1995.
- [50.] I. Freytsis, "Next Steps in Signaling: Framework," IETF Internet Draft: <draft-ietf-nsis-fw-02.txt>, working-in-progress, February, 2003.
- [51.] K. Almeroth, "The Evolution of Multicast, from MBone, Inter-domain Multicast to Internet 2 Deployment," IEEE Network Magazine, vol. 14, no. 1, pp: 10-20, January, 2000.
- [52.] K. Kompella and Y. Rekhter, "LSP hierarchy with MPLS TE," IETF Internet Draft <draft-kompella-lsp-hierarchy-00.txt> working-in-progress, February, 2003.
- [53.] I. Khalil and T. Braun, "Implementation of a Bandwidth Broker for Dynamic End-to-End Resource Reservation in Outsourced Virtual Private Networks," in Proceedings of the 25th IEEE Conference on Local Computer Networks, USA, p. 511, November, 2000.
- [54.] L. Mathy, D. Hutchison and S. Simpson, "Modeling and Improving Flow Establishment in RSVP," in Proceedings of the 6th International workshop on Protocols for High Speed Networks, August, 1999.

- [55.] M. Asawa, "Measuring and Analyzing Service Levels: A Scalable Passive Approach," in Proc. of the 6th IEEE/IFIP International Workshop on Quality of Service (IWQoS), 1998.
- [56.] M. Decina and V. Trecordi, "Convergence of Telecommunications and Computing to Networking Models for Integrated Services and Applications," in Proceedings of the IEEE, vol. 85 no. 12, pp. 1887-1914, December, 1997.
- [57.] M. El-Darieby and A. Bieszczad, "Intelligent Mobile Agent: Towards Network Fault Management Automation," in Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management (IM'99), USA, pp: 611-622, May, 1999.
- [58.] M. El-Darieby and J. Rolia, "Performance Modeling of a Service Provisioning Design," in "Trends towards a Universal Service Market," (Eds. Claudia Linnhoff-Popien), Lecture Notes in Computer Science (LNCS) vol. 1890, pp: 81-92, Springer, 2000.
- [59.] M. El-Darieby, D. Petriu and J. Rolia, "A Hierarchical Distributed Protocol for MPLS Path Creation," in Proceedings of the 7th IEEE Symposium on Computers and Communications (ISCC), Italy, July, 2002
- [60.] M. El-Darieby, D. Petriu and J. Rolia, "Scalability Analysis of Virtual Network-Based Service Provisioning," in Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM'01), USA, May, 2001.
- [61.] M. El-Darieby, D. Petriu and J. Rolia, "Hierarchical End-to-End Service Recovery," in Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management (IM'03), Colorado Springs, USA, pp: 648-661, May, 2003.
- [62.] M. El-Darieby, D. Petriu and J. Rolia, "HDRA: Hierarchical Distributed Routing Algorithm," in Proceedings of IEEE Globecom, USA, December, 2003.
- [63.] M. Faloutsos, P. Faloutsos and C. Faloutsos, "On Power-Law Relationships of the Internet Topology" in ACM SIGCOMM Computer Communication Review, a publication of ACM SIGCOMM, vol. 29, no. 4, pp: 251-262, October, 1999.
- [64.] M. Ramalho, "Intra- and Inter-Domain Multicast Routing Protocols, A Survey and Taxonomy," IEEE Communications Surveys, vol. 3, no. 1, 2000. available at:
<http://www.comsoc.org/livepubs/surveys/public/1q00issue/ramalho.html>

- [65.] M. Ahmed, "Performance-Oriented Analysis of the Open Shortest-Path-First Routing Protocol," M.Eng. Thesis, Carleton University, Canada, September, 1999.
- [66.] M.E. Kounavis, A. Campbell, S. Chou, J. Vicente and H. Zhuang, "The Genesis Kernel: A Programming System for Spawning Network Architectures," IEEE Journal of Selected Areas in Communications, vol. 19 no. 3, pp: 511-526, March, 2001.
- [67.] M.K. Girish, B. Zhou and J. Q. Hu, "Formulation of the Traffic Engineering Problems in MPLS Networks," in Proceedings of the 5th IEEE Symposium on Computer and Communication (ISCC), France, July, 2000.
- [68.] Multiservice Switching Forum Architectural document, at: www.msforum.org
- [69.] M. Gunter, T. Braun, "Evaluation of Bandwidth Broker Signaling," in Proceedings of the 7th International Conference on Network Protocols, Canada, October, 1999.
- [70.] P. Chimento and B. Teitelbaum, "Simple Inter-domain Bandwidth Broker Signaling," in Proceedings of the 1st Joint Internet 2/ DOE QoS Workshop QBone, USA, 2000.
- [71.] OMNET++ simulator, available at: <http://www.hit.bme.hu/phd/vargaa/omnetpp.htm>
- [72.] P. Chandra, A. Fisher, C. Kosak, T. Ng, P. Steenkiste, E. Takahashi and H. Zhang, "Darwin: Customizable Resource Management for Value-Added Network Services," in Proc. of the 6th IEEE International Conference on Network Protocols, USA, October, 1998.
- [73.] A. Banerjee, J. Drake, D. Awduche, L. Berger K. Kompella and Y. Rekhter, "Generalized Multi-protocol Label Switching: Overview of Signaling Enhancements and Recovery Techniques," IEEE Communications Magazine, vol. 39, no. 7, pp: 144-151, July, 2001.
- [74.] P. Pan, E. Hahne, and H. Schulzrinne, "BGRP: A Tree-Based Aggregation Protocol for Inter-Domain Reservations," *Journal Communication and Networks*, vol. 2, no. 2, pp: 157-167, June, 2000.
- [75.] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, "Resource reservation protocol (RSVP) version 1," IETF RFC 2205, September, 1997.
- [76.] R. Cohen, R. Emek and E. Felstine, "Framework for Multicast in Hierarchical Networks," in Proceedings of IEEE INFOCOM, Tel-Aviv, pp: 1673-1682, March, 2000.

- [77.] R. Guerin and V. Peris. "Quality-of-Service in Packet Networks: Basic Mechanisms and Directions" *The International Journal of Computer and Telecommunications Networking*, vol. 31, no. 3, pp: 169-189, February, 1999.
- [78.] P. Pan and H. Schulzrinne, "YESSIR: A Simple Reservation Mechanism for the Internet," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp: 89- 101, April, 1999.
- [79.] R. Rabbat, K. Laberteaux, N. Modi and J. Kenny, "Traffic Engineering Algorithms Using MPLS for Service Differentiation," in *IEEE International Conference on Communications, (ICC), USA, June, 2000*.
- [80.] A. Neogi, T. Chiueh and P. Stirpe, "Performance Analysis of an RSVP-Capable Router," *IEEE Network Magazine*, vol. 13, no. 5, pp: 56-63, September, 1999.
- [81.] S. Vutukury and J.J. Garcia-Luna-Aceves, "A Simple Approximation to Minimum-Delay Routing," in *Proceedings of ACM SIGCOMM '99*, pp: 227—238, September, 1999.
- [82.] Z.-L. Zhang, Z. Duan, and Y. T. Hou, "On Scalable Networking Resource Management Using Bandwidth Brokers," in *Proceedings of the 8th IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), Italy, April, 2002*.
- [83.] S. Chen and K. Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation Networks, Problems and Solutions," *IEEE Network Magazine*, vol. 12, no. 5, pp: 64-79, November, 1998.
- [84.] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in High-Speed Networks Based on Selective Probing," in *Proceedings of 23rd Annual Conference on Local Area Networks (LCN'98)*, pp: 80-89, USA, October, 1998.
- [85.] S. Keshav and R. Sharma, "Issues and Trends in Router Design," *IEEE Communications Magazine*, vol. 36, no. 5, pp: 144- 151, May, 1998.
- [86.] S. Norden and J. Turner, "New Inter-domain QoS Routing Algorithms," *Tech. Report-WUCS-02-03*, Dept. of computer Science, Washington Univ., March, 2002.
- [87.] S. Uhlig and O. Bonaventure, "Implications of Inter-domain Traffic Characteristics on Traffic Engineering," *European Transactions on Telecommunications*, January, 2002.

- [88.] S. Shenker, "Fundamental Design Issues for the Future Internet," IEEE Journal on Selected Areas in Communications, vol. 13 no. 7, 1176-1188, September, 1995.
- [89.] V. P. Kumar, T.V. Lakshman and D. Stiliadis, "Beyond Best Effort, Router Architectures for the Differentiated Services of Tomorrow's Internet," IEEE Communications Magazine, vol. 36, no. 5, May, 1998. <http://www1.bell-labs.com/user/stiliadi/router/router.html>
- [90.] S. Salsano, M. Winter, N. Miettinen, "The BGRP Plus Architecture for Dynamic Inter-Domain IP QoS," International Workshop on Inter-domain Performance and Simulation, Austria, February, 2003.
- [91.] V. Sharma and F. Hellstrand, "A Framework for MPLS-based Recovery," IETF RFC 3469.
- [92.] S. Giordano, M. Potts, M. Smirnov and G. Ventre, "Advanced QoS Provisioning in IP Networks: The European Premium IP Projects," IEEE Communications Magazine, vol. 41, no.1, pp: 30-36, January, 2003.
- [93.] X. Xiao, "Traffic engineering with MPLS in the Internet," IEEE Network Magazine, vol. 36, no.3, March, 2000.
- [94.] X. Xiao, "Internet QoS: A Big Picture," IEEE Network Magazine, vol. 13, no. 2, pp: 8-18, March, 1999.
- [95.] Y. Bernet, "The complementary roles of RSVP and differentiated services in the full service QoS network," IEEE Communications Magazine, vol. 38, no. 2, pp: 154-162, February, 2000.
- [96.] A. Fumagalli and L. Valcarenghi, "IP Restoration vs. WDM Protection: Is There an Optimal Choice?" IEEE Network Magazine, vol. 14, no. 6, pp: 34-41, December, 2000.
- [97.] K. Krauter, R. Buyya and M. Maheswaran, "A Taxonomy and Survey of Grid Resource Management Systems," Software Practice and Experience, vol. 32, no. 2, pp: 135- 164, February, 2002.
- [98.] Wei Li, "Inter-domain Routing: Problems and Solutions," Feb 2003. Experimental Computer Systems lab, Technical Report-128, Department of Computer Science, State University of New York, available at http://www.ecsl.cs.sunysb.edu/tech_reports.html

- [99.] H. Fu and E. Knightly, "Aggregation and Scalable QoS: A Performance Study," in Proceedings of IEEE workshop on QoS (IWQoS) 2001, Germany, June, 2001
- [100.] D. Frigioni, M. Ioffreda, U. Nanni, and G. Pasqualone, "Experimental Analysis of Dynamic Algorithms for the Single Source Shortest Paths Problem," in ACM Journal of Experimental Algorithms, vol. 3, no. 5, Article 5, 1998.
- [101.] R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling," Wiley Inter-science, New York, NY, April, 1991