# Internet Traffic Engineering: QoS Translation and Survivability

DISSERTATION

Presented in Partial Fulfillment of the Requirements for

the Degree Doctor of Philosophy in the

Graduate School of The Ohio State University

By

Mukul Goyal, B.Tech., M.S.

* * * * *

The Ohio State University

2003

Dissertation Committee:

Dr. Ming T. Liu, Adviser

Dr. Wu-chi Feng, Adviser

Dr. Dong Xuan

Approved by

____________________________

Adviser

Computer and Information
Science Department

## ABSTRACT

The problem of traffic engineering has been widely recognized as critical to the development of operational Internet. Traffic engineering involves evaluation and optimization of the performance of operational IP networks. This thesis studies the performance evaluation problem from the perspectives of the users and the providers. Scalability requirements necessitate the use of an analytical framework to translate network level service quality in terms of the performance of user level applications. This thesis presents an important building block for such a framework in the form of an analytical model for TCP throughput prediction using easily available input information. Evaluating the performance of the networks from the providers' perspective naturally involves taking into account the service quality requirements of the users. The ever-increasing dependence on the Internet has made it important that the Internet service can survive the failures in the networks. The cost of survivability and the speed of recovery constitute the two important aspects of the network survivability. This thesis evaluates the cost of network operation for the providers taking into account the survivability requirements. This is followed by an investigation into the role of simple topological modifications in reducing the cost of survivability. The speed of recovery problem is studied from the perspective of different time scales. At the macro time scale, the objective is to enable fast detection and correction of network failures by the network administrators. This thesis compares the two most relevant alternatives for the purpose. At the micro time scale, the objective is fast rerouting of the traffic around the failures by the in-built survivability mechanisms while the failures have not yet been corrected. This thesis investigates the operation of popular survivability mechanisms operating at the IP and optical layers and suggests ways and means of improving their speed of recovery from network failures.

# Internet Traffic Engineering: QoS Translation and Survivability

By

Mukul Goyal, Ph.D.

The Ohio State University, 2003

Dr. Ming T. Liu, Adviser

The problem of traffic engineering has been widely recognized as critical to the development of operational Internet. Traffic engineering involves evaluation and optimization of the performance of operational IP networks. This thesis studies the performance evaluation problem from the perspectives of the users and the providers. Scalability requirements necessitate the use of an analytical framework to translate network level service quality in terms of the performance of user level applications. This thesis presents an important building block for such a framework in the form of an analytical model for TCP throughput prediction using easily available input information. Evaluating the performance of the networks from the providers' perspective naturally involves taking into account the service quality requirements of the users. The ever-increasing dependence on the Internet has made it important that the Internet service can survive the failures in the networks. The cost of survivability and the speed of recovery constitute the two important aspects of the network survivability. This thesis evaluates the cost of network operation for the providers taking into account the survivability requirements. This is followed by an investigation into the role of simple topological modifications in reducing the cost of survivability. The speed of recovery problem is studied from the perspective of different time scales. At the macro time scale, the objective is to enable fast detection and correction of network failures by the network administrators. This thesis compares the two most relevant alternatives for the purpose. At the micro time scale, the objective is fast rerouting of the traffic around the failures by the in-built survivability mechanisms while the failures have not yet been corrected. This thesis investigates the operation of popular survivability mechanisms operating at the IP and optical layers and suggests ways and means of improving their speed of recovery from network failures.

Dedicated to my parents.

# ACKNOWLEDGMENTS

I would like to express my gratitude towards my family - my parents, my wife, my brother, my mentors - Dr. Wu-chi Feng, Dr. Ming T. Liu, Dr. K.K. Ramakrishnan, Dr. Jennifer Yates, Dr. Raju Rajan, Dr. Roch Guerin, Dr. Guangzhi Li, Dr. Raj Jain and all my friends and colleagues who with their acts of kindness made it possible for me to achieve whatever little I have achieved.

# VITA

|  |  |
|---|---|
| 1995 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | B.Tech. Electronics and Communication Engineering, Kurukshetra University, India |
| 1999 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | M.S. Computer and Information Science, The Ohio State University, Columbus, Ohio |

# PUBLICATIONS

## Research Publications

M. Goyal, R. Guerin, R. Rajan "Predicting TCP Throughput From Non-invasive Network Sampling". Proc. IEEE INFOCOM, June 2002.

M. Goyal, G. Li, J. Yates "Shared Mesh Restoration: A Simulation Study". Proc. Optical Fiber Communications Conference, March 2002.

A. Shaikh, M. Goyal, A. Greenberg, R. Rajan, K.K. Ramakrishnan "An OSPF Topology Server: Design and Evaluation". *Journal of Selected Areas in Communications: Special Issue on Recent Advances On Fundamentals of Network Management*, 20(4):746–755, May 2002.

M. Goyal, A. Durresi, P. Misra, C. Liu, R. Jain "Effect of Number of Drop Precedences in Assured Forwarding". Proc. IEEE GLOBECOM, pages 188–193, December 1999.

S. Kota, M. Goyal, R. Goyal, R. Jain ”Broadband Satellite Network: TCP/IP Performance Analysis". Proc. IFIP TC6 WG6.2 Fifth International Conference on Broadband Communications (BC'99), November 10-12, 1999, Hong Kong, IFIP Conference Proceedings 159, published as Danny H. K. Tsang, Paul J. Khn (Eds.), ”Broadband Communications: Convergence of Network Technologies," Kluwer 2000, ISBN 0-7923-8677-9, pages 273–282.

S. Kota, M. Goyal, R. Goyal, R. Jain ”Multimedia Satellite Networks and TCP/IP Traffic Transport”. Proc. Internet, Multimedia Systems and Applications (IMSA’99), pages 436–443, October 18-21, 1999, Nassau, The Bahamas.

S. Kota, M. Goyal, R. Goyal, R. Jain ”Multimedia Satellite Networks and TCP/IP Traffic Transport”. *International Journal of Computers and Applications*, 23(2):115–128, 2001.

A. Durresi, S. Kota, M. Goyal, R. Jain, V. Bharani ”Achieving QoS for TCP Traffic in Satellite Networks with Differentiated Services”. *Journal of Space Communications*, 17(1-3):125–136, 2001.

A. Durresi, M. Sridharan, C. Liu, M. Goyal, R. Jain ”Multilevel Explicit Congestion Notification”. Proc. 10th IEEE International Conference on Computer Communications and Networks (ICCCN2001), pages 483–488, October 14-19, Phoenix, Arizona, 2001.

R. Goyal, R. Jain, M. Goyal, S. Fahmy, B. Vandalore, S. Kota, N. Butts, K. Bhasin, T. VonDeak ”Buffer Management and Rate Guarantees for TCP/IP over Satellite-ATM Networks”. *International Journal of Satellite Communications*, 19(1):111–129, 2001.

S. Kota, A. Durresi, M. Goyal, R. Jain, V. Bharani ”A Simulation Study of QoS for TCP over LEO Satellite Networks with Differentiated Services”. Proc. Opnetwork 2000, August 28-31, 2000, Washington DC.

B. Vandalore, S. Fahmy, R. Jain, R. Goyal, M. Goyal ”General Weighted Fairness and its Support in Explicit Rate Switch Algorithms”. *Computer Communications*, 23(2):149–161, January 2000.

B. Vandalore, S. Fahmy, R. Jain, R. Goyal, M. Goyal ”Overload Based Explicit Rate Switch Schemes with MCR Guarantees”. Proc. ICCCN ’99, October 11-13, 1999, Boston.

R. Goyal, R. Jain, S. Kota, M. Goyal, S. Fahmy, B. Vandalore ”Traffic Management for TCP/IP over Satellite-ATM Networks”. *IEEE Communications Magazine*, 37(3):56–61, March 1999.

B. Vandalore, S. Fahmy, R. Jain, R. Goyal,M. Goyal ”QoS and Multipoint support for Multimedia Applications over ATM ABR service”. *IEEE Communications Magazine*, 37(1):53–57, January 1999.

B. Vandalore, S. Fahmy, R. Jain, R. Goyal, M. Goyal ”A Definition of General Weighted Fairness and its Support in Explicit Rate Switch Algorithms”. Proc. Sixth International

Conference on Network Protocols 1998 (ICNP'98), pages 22–30, Austin, TX, October 13-16, 1998.

## FIELDS OF STUDY

Major Field: Computer and Information Science

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

Internet traffic engineering is defined as *the evaluation and optimization of the performance of operational IP networks* [38]. When dealing with traffic engineering in the context of IP networks, the important question is: what is the meaning of the *performance*? The performance of operational IP networks has very different yet related meanings for people with different perspectives. A user will define the performance in terms of the cost and the quality of the Internet service, which will be judged on the basis of the performance of the Internet based applications such as web browsing, file transfer, streaming audio-video etc. On the other hand, an *Internet Service Provider* (ISP) will determine the performance of the network under its control in terms of the cost of operating the network to support the agreed upon *quality of service* (QoS) requirements for its users. The QoS guarantees agreed upon with the users constitute the constraints in terms of which the provider tries to optimize the performance of their networks. The extent to which the provider succeed in this optimization determines the cost of QoS for the users.

This thesis deals with the evaluation and optimization of the performance of the operational IP networks. The first part of the thesis deals with the evaluation of the performance of the Internet from the user perspective while the second part of the thesis focuses on the evaluation and optimization of the network performance from the perspective of the internet service providers. As discussed above, the user and provider perspectives on the network *performance* are intimately related via the QoS expectations of the users. The service quality perceived by human users is influenced by a number of factors many of which are quantifiable objective metrics while others are largely qualitative subjective criteria. The quantifiable objective metrics consist of factors such as the cost and availability of the Internet service and whether the service meets the throughput, loss and delay (the so-called *network-level* QoS) requirements of the user applications. The subjective and qualitative criteria include factors such as the perceived *usefulness* of the service. Chapter 2 begins with a discussion of the previous findings on the factors that influence the service quality perception of the human beings and also contains a survey on the network-level QoS requirements of the user applications.

Relating user-level perception of service quality to the network-level QoS requirements of the applications may be useful in the scenarios where the user negotiates a custom-built *service level agreement* (SLA) with the provider. However, such scenarios are difficult to find in practice, where the norm is the service provider offering a few standard service options to the users. These service choices are associated with a particular SLA that specifies the network-level QoS guarantees available to the users. However, such guarantees make little sense to most of the users. A user would like to know how these QoS guarantees translate in terms of the performance of the Internet based applications - the download time for a particular web page, the file transfer times between specified locations, the quality of the Internet based audio-video applications such as video-on-demand or video conferencing etc. Translating network-level QoS into the performance of the user applications is a complicated task that can be attempted using two different approaches. The first approach involves straight forward measurement of the application performance on a per-user and per-application basis. This approach clearly suffers from the lack of scalability as the number of users and the applications increases. Further, since the performance of an application depends on multiple factors, the measurement based approach does not offer any help in pinpointing the reasons for bad performance. For example, the time it takes to download a web page depends on a number of factors such as the location of the user and the website, the prevalent loss rate on the links connecting the user to the website, the contents of the web page and the delays associated with the DNS/WWW server latencies. The user-perceived download time will be the combined effect of all these causes and hence simple measurement of the download times offer little insight into the reasons for poor performance. The measurement based approach for evaluating the user-level performance is made additionally complicated by the fact that the users will not be satisfied with periodic evaluation of a few fixed metrics, say the typical download times for few popular web pages or typical file transfer times for a set of file sizes and destinations. The users need real-time intimation of the network performance for the metrics of their choice. Clearly, providing such dynamic, real-time measures of Internet service performance on a per-user basis using an application measurement approach is not feasible for the providers.

The second approach for the evaluation of user-level Internet service performance is to develop a sound and reliable performance *prediction* framework. Such a framework can provide the Internet service performance evaluation on a per-user and per-application basis in a scalable manner. The scalability with respect to the number of users comes by splitting the network-level QoS characteristics into different domains - user to access router QoS, metropolitan area QoS and backbone QoS - and factoring out the common factors. The scalability in terms of the number of applications comes since the performance of most of the applications is determined by several common factors. For example, most of the applications are based on transport layer protocols such as TCP [5] and UDP [116] and hence the application performance depends to a large extent on the performance of these protocols. Thus, the performance prediction framework based on such common building-blocks can be used for a number of applications. The important challenge in building such a framework is its usability in an operational environment. Particularly, the input parameters required for the framework should be easily measurable in an operational network. In this thesis, we describe perhaps the most important building block for such a framework - translation of network-level QoS into an accurate prediction for long life TCP throughput. Chapter 3 deals with the development and evaluation of analytical models for TCP throughput prediction particularly in relation to their suitability to the user-level performance evaluation needs of the providers.

Over past several years, the Internet has become an important means for communication, commerce and entertainment all over the world. More emails are sent every day than the "snail mail" letters. Email is replacing the telephone as the preferred means of instant communication. Video conferencing has emerged as a viable alternative to travel. People prefer downloading an MP3 song than buying the CD or the cassette tape. The new advances in the Internet technology will soon allow people to download latest movies over the Internet and watch them in the comforts of their homes. Billions of dollars worth of business transactions take place daily with people buying and selling products/stocks/services over the Internet. Governments/businesses/hospitals all over the globe are connected to their databases via the Internet. Devices placed in remote and inaccessible environments are controlled over the Internet. The Internet is being used even to conduct remote surgery by the medical doctors. Since the Internet has become such an important requirement of our life, it natural that the people's expectations from the Internet have grown by leaps and bounds. One important aspect, where the user expectations have significantly changed is the *survivability* aspects of the Internet service. By survivability, we mean the ability of the network to maintain the connectivity and QoS guarantees in face of component failures within the network. In older days, the users could tolerate the unavailability of Internet service for a few minutes or even a few hours. The deterioration in the service quality from time to time even for significant durations was considered acceptable. However, today people depend on Internet so much that interruption in the Internet service or any deterioration in the service quality might prove disastrous. Today, people expect their Internet service to be available all the time with no deterioration in the service quality even if there are major facility failures in the network. Hence, it is imperative that the evaluation of the performance of the network takes into account the survivability requirements.

The second part of this thesis deals with the evaluation and optimization of the network performance from the perspective of a service provider taking in account the survivability requirements of the network. As mentioned earlier, from the provider's perspective the performance of the network is measured in terms of the cost of operating the network while providing the QoS guarantees agreed upon with the users. While there are many factors that determine the service quality from the user's perspective, the bandwidth availability constitutes perhaps the most important factor. The bandwidth requirements of the users also determine the capacity requirements for the links in the network, which is the most important factor in determining the cost of operating the network. When we consider survivability as an integral part of the user QoS requirements, the task of evaluating the network performance from the provider's perspective becomes considerably more complicated. The provider needs to ensure that the network has sufficient connectivity as well as sufficient redundant capacity on the links so that the user level service quality is not affected by the failures in the network. The extra capacity requirements clearly increase the cost of operating the network. In addition to understanding the cost of network operation in order to meet survivability needs, the provider has to ensure that the recovery from network failures is fast so that the user applications are barely affected by the failure in the network. Thus, the survivability requirements contribute the following dimensions to the task of evaluating the network performance from the provider's perspective:

1. Can the network survive common failures?

2. What is the cost of network operation for protection against such failures?

3. How soon can the network recover from the failures?

Since all three dimensions mentioned above are influenced by the "failures" in the network, an important question is: the network should be protected against what failures? A network can be struck with a range of failures comprising common failures like the failure of a link or a node to highly improbable failures such as the failures of all the nodes in the network. Clearly, it is not feasible to provide protection against all possible failures that might affect the network. The network needs to be protected against common failures and those probable failures that can significantly deteriorate the user level QoS performance. Single link and node failures fall under the first category while facility failures such as the failure of all the routers located in a building (some times called a *Point of Presence* or a PoP) constitutes the kind of failures comprising the second category. Protection against facility failures has assumed increased importance in recent times due to the concerns regarding a terror attack crippling important network facilities. In fact the facility failures can be considered simply as the node failures in a macro-level view of the network topology. In our analysis, we consider protection against all *single node failures* as the prime survivability requirement. Single node failure refers to the situations where only one node is in a state of failure at any given time. Protection against all single node failures automatically provides protection against all single link failures, thus covering most of the common failures, and can be easily adapted to mean protection against all single physical facility failures, thus providing protection against most of the probable disaster scenarios.

Our work on the network performance evaluation from the provider's perspective begins in Chapter 4, which deals with the cost of network operation for protection against all single node failures. This chapter begins with an evaluation of the survivability characteristics of several current IP backbone topologies. We illustrate that a large number of these topologies can not survive at least one single node failure. We present ways to improve the connectivity with in the network so that the topologies are no longer susceptible against any single node failure. After making the topologies single node failure resistant, we move on to evaluating the cost of network operation with and without survivability. This evaluation is performed for several survivability mechanisms such as traditional OSPF routing and modern MPLS based techniques. This is followed by an investigation into how to reduce the cost of network operation while still satisfying the survivability requirements. Finally the chapter deals with the relationship between the cost of survivability and a good balance in the link utilizations. It is shown that the load balancing and reducing the cost of network operation for survivability often turn out to be non-complimentary goals, i.e., mechanisms designed to improve the balance in link utilizations can drastically increase the cost of network operation for survivability and vice versa.

The remaining chapters in the thesis deal with the other important aspect of survivability, i.e. the speed of recovery from network failures. We examine the speed of recovery problem from two different perspectives. The first perspective deals with the problem at a macro time scale level and is concerned with how quickly can the network administrators detect the failure and take steps to correct it. The second perspective tackles the problem at a micro time scale level and is concerned with how quickly can the in-built network survivability mechanisms detect the failure and restore the user connectivity and service quality while the failure has not yet been corrected. Note that the first perspective is essentially a network management problem and can be viewed as the problem of dynamically tracking the network topology so that the network failures can be quickly and reliably detected by the network administrators. Chapter 5 discusses the problem of dynamic tracking of network topology and provides a detailed comparison between the two most appropriate alternative approaches to solve the problem. The second perspective dealing with the "speed of recovery" performance of the in-built network survivability mechanisms, is a complicated issue in itself. Modern communication networks have a multi-layer structure with each network layer providing its own survivability mechanisms. Thus, optical WDM layer, which is emerging as the ubiquitous choice for the base layer because of its high capacity characteristics, provides several alternative mechanisms to protect the layer against common failures occurring at the optical layer. Similarly, the IP layer, which resides ubiquitously at the top of the multi-layer network stack, has its own mechanisms dealing with failures occurring at the IP layer. The alternative survivability mechanisms at each layer basically provide tradeoffs between the cost of survivability and the speed of recovery. For example, *Automatic Protection Switching* (APS) at the optical layer provides very fast (less than 50 ms) recovery from the optical layer failures where as *shared mesh restoration* provides survivability at a much lower cost than APS but the speed of recovery can be much slower. A corresponding example at the IP layer is MPLS local recovery versus OSPF based recovery. MPLS local recovery can provide much faster failure recovery but at a much higher cost than OSPF. Our investigations in this regard are focussed on understanding if both goals - low cost of survivability and fast failure recovery - can be simultaneously achieved. For this investigation, we focus on improving the speed of recovery for the survivability mechanisms that are inherently economic in terms of the cost of survivability. Chapter 6 deals with investigating the factors that determine the speed of failure recovery with shared mesh restoration in optical WDM networks. Chapter 7 deals with investigating the factors that determine the speed of recovery with OSPF routing in IP networks. In each chapter, we analyze the operation of the investigated survivability mechanism, identify the factors having the most significant impact on the speed of failure recovery and make recommendations that will result in significantly improving the speed of failure recovery even in adverse network conditions.

In this thesis, we have attempted to understand the issues behind *the evaluation and optimization of network performance* from the perspectives of both the users and the providers. Given the magnitude of the problem, our work represents only a small contribution to the massive research efforts required in tackling all aspects of the problem. In Chapter 8, we recap the contributions made in this thesis and provide a blueprint for the immediate next steps required towards solving the *Internet Traffic Engineering* problem.

# CHAPTER 2

# QUAILITY OF SERVICE FROM A USER'S PERSPECTIVE

## 2.1 Introduction

Quality of Service has been defined as *the collective effect of service performances which determine the degree of satisfaction of a user of the service*[36]. It is a set of perceivable attributes expressed in user-friendly language with parameters that may be subjective or objective. Objective values are parameters related to a particular service and are measurable and verifiable. Subjective values are based on the opinions of the end-users [138]. We discussed in Chapter 1 how different perspectives on the performance of the network, as seen by the users and by the providers, are intimately related via the QoS requirements of the users. However, *what are the QoS requirements of the users?* The QoS requirements of the users are essentially determined by the user's perception of the service quality, which is shaped by several qualitative/subjective and quantitative/objective criteria. The qualitative/subjective criteria often originate from the human psychology while the quantitative/objective criteria are essentially network-level QoS requirements of the applications taking into account the qualitative/subjective aspects. Hence, the problem of QoS requirements of the human users can be sub-divided into the following related problems:

1. What psychological factors determine the service quality from a user's perspective?

2. What are the network-level QoS requirements of different user applications taking in account these psychological factors?

This chapter attempts to gain an understanding of these problems by taking a look at the previous research work.

## 2.2 Human psychological factors that determine the QoS from a user's perspective

In order to gain an understanding of the psychological factors that determine the service quality from a user's perspective we did a literature search on the topic and found that the following factors have been observed to be important in this respect:

1. *Utility*: "Quality in a product or service is not what the supplier puts in. It is what the customer gets out and is willing to pay for. Customers pay only for what is of use to them and gives them value. Nothing else constitutes quality"[34]. Each service has a utility associated with it and hence the service quality should be good enough to allow the user to get what she/he wants from the service.

2. *Predictability*: Users need a simple way to be able to predict the quality of service as well as the cost associated with that service. The user desires peace of mind, which describes the state of mind arising from users' ability to commit to a service and not experience unacceptable changes in price or QoS. User needs predictability of future QoS behavior and the associated cost because it allows risk assessment regarding the value of money used to buy a certain QoS level [18]. Predictability also plays a big role in determining the *dependability* of the service, which is a measure of whether the user can depend on the service to meet mission-critical needs [92, 66, 1].

3. *Security*: Security is the performance criterion that describes the degree of confidentiality and accuracy associated with the service [92, 66, 1].

4. *Cost*: The cost associated with a given service also plays an important psychological role in deciding the user perception of the quality of the service.

## 2.3 The network-level QoS requirements of different user applications

Having discussed the largely non-measurable psychological factors influencing the service quality perception of the human users, we turn our attention to understanding the network-level QoS requirements of some of the popular Internet based applications. Traditionally, the Internet has followed a best-effort service paradigm which provides no QoS guarantees. Hence the question "What are the QoS requirements of the applications?" did not receive the attention it deserves. Best-effort Internet led to the development of user applications that adapt themselves according to the resource availability. This is true not only for *data-centric* applications like FTP but also for many multimedia applications. "These applications essentially always work regardless of how congested the network is – they just work a lot better when the network is better" [133]. Even the current Internet infrastructure can not support sophisticated QoS requirements. This lack of proper infrastructure has hindered the development of standard QoS requirements for different applications. In the following, we present a discussion of the QoS requirements of several user applications as gathered from a literature search on the topic.

At a high level, the Internet applications can be classified as data applications and real-time applications. Data applications consist of traditional applications like file transfer, email, web browsing and remote login. File transfer applications (document transfer, MP3/MPEG downloads) generally adapt themselves to network conditions and do not demand special treatment from the network. However, the speed of their completion and the human perception of their quality will improve with bandwidth and loss rate guarantees. Typically, these applications employ reliable TCP as the transport layer protocol. Some versions of TCP perform better if the loss process in the network is random in nature rather than correlated (or bursty). Modern email applications comprise not only document email but also audio/video messaging and have pretty much same QoS requirements as the file transfer applications. Remote login come under the category of interactive applications and have small latency as the key requirement. With the advent of e-commerce, security in the exchange of information has also become an important requirement. Supporting services like LDAP [140], Domain Name Service [98], Session Initiation Protocol [59], Network Time Protocol [97], Dynamic Host Configuration Protocol [33] constitute another important category of applications requiring low latency along with availability as the key QoS requirements. With the advent of Internet as the main communication media, big and small corporations are increasingly relying on Internet to connect their distant locations via Virtual Private Networks (VPN) [54]. VPNs typically demand secure transmission with guaranteed bandwidth, loss rate and delay characteristics.

In comparison to data applications, the real-time applications have stricter QoS requirements in terms of bandwidth, loss rate, delay and jitter which if not met cause significant deterioration in the utility of the applications. Real-time applications have been defined as “the ones that impose an upper bound on the delay between sender and receiver, that is, a message should be received by a particular deadline” [125]. However, we consider many other applications like streaming voice/video as belonging to this category. Note that technically speaking, streaming audio/video only require the timing relationship between successive packets to be maintained at destination and do not impose a delay bound (so called *Continuous media* services [6, 7]). Some of the real-time applications under this loose definition are:

1. two-way interactive audio/video e.g. voice over IP.
2. collaborative virtual environments, video conferencing, Internet games, chat etc.
3. video and audio streaming at high fidelity, video on demand, distance education.
4. remote control of devices (e.g., telescopes and microscopes) , telemedicine.
5. Storage area networks connecting heterogeneous storage devices and servers .
6. T.120 based data conferencing

In the following, we discuss the key network-level QoS requirements for these services.

### 2.3.1 The network-level QoS requirements of real-time applications

**The bandwidth requirements**

The bandwidth requirements of the real-time services can vary widely. For example, real-time measurement data may only generate a few bits per second, communication-quality voice 8 to 32 kbps, conversational, toll-quality voice 64 kbps, video anywhere from a few tens of kbps (slow-scan, QCIF images of 144 lines by 176 pixels) to several tens of Mbps (HDTV), depending on image size, quantization and the frame rate. The bandwidth requirements for *Voice over IP* applications depend essentially on the coding scheme used. Table 2.1 lists some of the popular coding schemes and corresponding bandwidth requirements. Note that in Table 2.1 the IP bandwidth refers to the actual bandwidth required without any header compression or other such mechanisms. Typically, about 16 kbps of

| Coding algorithm | Bandwidth | Sample Time | IP Bandwidth |
|---|---|---|---|
| G.711 | 64 kbps | 0.125 ms | 80 kbps |
| G.723.1 | 5.6/6.4 kbps | 30 ms | 16.27/17.07 kbps |
| G.726 | 32 kbps | 0.125 ms | 48 kbps |
| G.728 | 16 kbps | 0.625 ms | 32 kbps |
| G.729A | 8 kbps | 10 ms | 24 kbps |

Table 2.1: Popular Voice Coding Schemes and their Bandwidth Requirements.[64]

bandwidth is required just to transmit the RTP, UDP an IP headers (40 bytes per packet and about 50 packets per second). Sample Time refers to how frequently a sample of the analog signal is taken.

Different coding methods vary in the levels of complexity and delay characteristics. The G.711 coding transmits voice at either 56 or 64 kbps and is well suited for high bandwidth environments. However, bandwidth is often a concern in a LAN/WAN setting, and the G.723.1 coding is becoming a popular option in H.323-protocol multimedia applications. The G.723.1 audio codec offers somewhat inferior audio quality to the G.711 codec, but offers much better compression, requiring only 5.6 or 6.4 kbps of bandwidth. Because of their low bandwidth requirements, it is expected that G.729A and G.723.1 will become prevalent in the Voice over IP arena. For a typical 8 kbps compression scheme, the packet header overhead can be as high as 16 kbps (or 200reduce this overhead. These include silence suppression, RTP header compression and RTP multiplexing. Similar to Voice over IP, the bandwidth required for video applications depends on the coding scheme used. The two popular coding schemes for video applications are H.261 and H.263. The H.261 coding generally provides less compression and therefore requires more bandwidth than the H.263 coding. The H.263 coding was designed to be more resilient to packet loss. It therefore

| Video Format | Image Resolution (Pixels) | H.261 | H.263 |
|---|---|---|---|
| Sub-QCIF (Sub-Quarter Common Intermediate Format) | 128 x 96 | | X |
| QCIF (Quarter Common Intermediate Format) | 176 x 144 | X | X |
| CIF (Common Intermediate Format) | 352 x 288 | X* | X* |
| 4CIF | 702 x 576 | | X* |
| 16CIF | 1408 x 1152 | | X* |

Table 2.2: Video Resolutions Supported by H.261 and H.263 Schemes (X: Supported,* meansOptional)[110]

provides better image quality in low bandwidth environments and also offers more video resolution options. JPEG (1-4 Mbps) [70], AVI, QuickTime, DVI, MPEG-1 (VHS quality, 1.5 Mbps) [71], MPEG-2 (theater quality, 3-60 Mbps, average 3-6 Mbps) [23, 24, 60], MPEG-4 [73], MPEG-7 [28] and J.81 [135] are some of the other coding techniques. In addition to the coding scheme, the video frame rate and the resolution used are the other factors important in determining the bandwidth requirements for video application. Table 2.2 describes the video resolution formats supported by the H.261 and H.263 schemes. In general, greater image resolution requires greater bandwidth to transmit and receive.

The H.323 [137] is ITU-T"s standard for multimedia conferencing over any packet based network. H.323 based video conferencing terminals must support G.711 audio and H.261 video coding schemes. The support for other audio (G.722, G.723.1, G.728, G.729) and video (H.263) schemes are optional. H.323-protocol clients could participate in a high quality video conference using the G.711 audio and H.261 CIF video. Each client would consume approximately 384 kbps of bandwidth in each direction (send/receive). The typical bandwidth required for a low quality video chat is 64 kbps in each direction.T.120-protocol data conferencing applications such as file transfer, white board, and application sharing, have widely varied bandwidth requirements. Unlike audio and video, which require consistent bandwidth performance, data applications tend to require bandwidth in bursts and can tolerate slow-downs in network performance.

**The delay and jitter tolerance**

The delay tolerance of real-time applications varies widely, from 12 ms end- to-end for voice without echo cancellation (G.164) to 400 ms (G.114) [100] if only conversational latencies and double- talk are an issue. User studies for the ITU indicate that telephony users perceive communication with round-trip delays of greater than about 300 ms more as simplex connections rather than duplex conversations. However, depending on the application more tolerant users were satisfied with delays of 300-800 ms [68]. Conversations with a round-trip delay close to a second cannot easily use “normal” social protocols for talker selection. For playback applications like video-on-demand, delays can be longer, maybe 500 ms or more, primarily limited by the responsiveness to VCR-type commands and the ability of the receiver to buffer the data [9]. From an application standpoint, a principal challenge in supporting interactive audio over a wide-area network is the need to provide synchronous playout of audio packets in the face of delay jitter due to queuing at switches and routers. This is typically achieved by buffering received audio packets (i.e., delaying their play-outs) for enough time so that “most” of the packets will have been received before their scheduled playout times. This additional artificial delay until playout can either be fixed throughout the duration of an audio call, or may vary adaptively during a call’s lifetime [27, 142]. Still, it is expected that the probability that jitter exceeds 20 ms is less than 10problem is that the sender clock and receiver clock are likely not synchronized, neither among themselves nor to a global clock. Even the small frequency deviation of a quartz crystal ( = 100 parts per million) would yield a drift of 360 ms per hour. Thus, for some services such as synchronous data or frame-synchronized MPEG it is necessary to have the receiver perform clock recovery from the data: To avoid data loss, the receiver has to estimate the receiver's true clock rate and adjust its own clock rate accordingly. For services which are "interrupted-continuous", as voice with silence periods, normal playout delay compensation during silence periods will usually take care of the clock drift without further effort. In order to guarantee user satisfaction, the desired synchronization skew between the audio and video streams was required to be 80 ms or lower. Perceptual studies [132] indicate that the skews within this boundary are acceptable to the user. Skews in the range between 80 and 160 ms are tolerable by the user. Skews larger than 160 ms are unacceptable and undesirable by the user.

**The loss tolerance**

The acceptable loss rate constitutes another important part of the QoS requirements. Note that for most of the real- time applications, packets that miss their deadline are considered lost (*late loss*), just as if they had been dropped at a switch or router. Within real-time services, we can (roughly) distinguish between hard real-time requirements, where missing deadlines or losing packets may have catastrophic consequences and soft real-time requirements where some delayed/lost packets can be tolerated. Applications such as remote control of a manufacturing process or telemedicine come under the category of hard real-time applications and interactive audio falls into the soft real-time category. User studies have shown that the human ear is by far more sensitive to erroneous audio transmission than defective video transfer. Typically, the use of Forward Error Correction mechanisms imply that certain amount of loss can be tolerated. Still, the QoS requirements for audio with respect to error liability are quite strict: the acceptable losses can not be more than 5ap-plications, the loss tolerance depends on the coding scheme but typically losses more than 1deterioration in perceptible quality [139]. In fact, it seems that for some coding schemes like MPEG-2, the perceived quality depends on both coding bit rate and experienced loss rates [139]. For a given loss rate, there is an optimal coding bit rate beyond which the quality suffers. Applications like Storage Area Networks have very small threshold for errors; the tolerable losses can not exceed about 1 error every 16 minutes [108].

# CHAPTER 3

# TRANSLATING NETWORK LEVEL QUALITY OF SERVICE CHARACTERISTICS TO USER LEVEL APPLICATION PERFORMANCE

## 3.1 Introduction

In the previous chapter, we examined the factors that determine the human perception of service quality and typical network-level QoS requirements of several user applications. We observed that user applications can be divided into two categories - data applications and real-time applications. While real-time applications have specific network-level QoS requirements, the data applications adapt themselves to the QoS level prevalent in the network. Streaming audio-video constitutes a prime example of real-time applications whereas e-mail, web browsing, file transfer and telnet are the main examples of the data applications. While timely delivery of information is very important for real-time traffic, the accuracy and reliability of data transfer is of at most concern to data applications. The data applications rely on TCP [5] protocol to achieve accurate and reliable transfer of traffic from the source to the destination. At present, the traffic from data applications, specifically TCP traffic, constitutes an overwhelming majority of the traffic on the Internet [1]. This trend

[1]TCP traffic constitutes about 83% of the packets and 91% of the bytes on the Internet. Source: www.caida.org

is expected to continue for several years in future too. Thus, from the user's perspective, the most important aspect of the service quality is the performance of the data applications. The data applications being elastic in nature do not demand hard QoS guarantees. However, the human perception regarding the performance of such applications, typically characterized by response or completion time, improves with better bandwidth availability and lower packet loss probability in the network. Most of the data applications can be categorized as either *file transfer* applications or *interactive* applications. These two categories are distinguished on the basis of the amount of information transferred between the source and destination. The file transfer applications involve transfer of significant amount of information whereas the interactive applications transfer only a few bytes at a time. As the data transfer speeds continue to approve, the boundary between the file transfer and interactive applications is become thinner. For example, the web browsing can quality as either file transfer or interactive based on the contents of the web page. Since file transfer applications involve transfer of significant amounts of data from a source to a destination, the performance of the underlying data transfer protocol TCP becomes the main factor in determining the completion time. On the other hand, for interactive applications, several factors including the latency associated with the TCP protocol determine the completion or response time.

While the previous chapter focussed on the evaluation of network-level QoS requirements of user applications, such an evaluation may not be very relevant in the context of data applications. Data applications are elastic in the nature and adapt themselves to the available QoS in the network. Hence, it is more meaningful to translate the available QoS in the network to the performance of the data applications than the other way around. In fact, since the Internet service options available to the individual consumers are severely limited at present, knowing the network QoS requirements of user applications - data or real-time - may not be useful at all. A present, the available Internet service options to home consumers consist of either narrowband (less than few hundred kbps) or broadband (about 1 Mbps) access to the ISP. The connection between the ISP and the home consumer - the so called *last mile* - essentially determines the bandwidth availability and loss rate suffered by the to and fro traffic from the user. Once past the ISP, the user traffic gets the *best effort* treatment in the Internet where the network QoS observed by the user traffic depends on the available resources on the links it travels through and the competition for resources from other traffic on the link. In the near future, the Internet might be able to offer different *levels of service* (or different *traffic classes*) to the consumers using the diffserv based mechanisms [17]. However, within a service level or traffic class, the user traffic will still get only the best effort treatment. Hence, knowing the network QoS requirements of the user applications - data or realtime - is somewhat futile since at present the Internet can not honor hard QoS requirements. The more relevant exercise from the user's perspective is to translate available network QoS into the performance of user applications. Translation of network QoS into the performance of user applications allows the user to make easy comparison between different service levels. Further, such a translation allows the user to effectively monitor if the observed service quality is commensurate with the promised service quality.

In this chapter, we focus on the problem of translating network-level QoS into the performance of data applications. As discussed earlier, most of the traffic on the Internet belongs to data applications and hence translating network-level QoS into the performance of data applications constitutes the logical first step in evaluating the performance of the Internet from the user's perspective. With in the data applications, we focus our attention on the performance evaluation of *file transfer* category of applications. As discussed before, the performance of such applications is calculated as the file transfer completion times. Translating network QoS characteristics such as loss rate and delay into the application level performance can be done in several ways. The simplest method will be to *measure* the application level performance and report it to the consumer. As discussed in Chapter 1, such an approach runs into several difficulties and is impractical from the perspective of the ISPs. The other alternative is to use a performance prediction framework made of building blocks that play a deciding role in determining the perceived performance of several applications. The idea is to predict the application performance by combining together the individual impacts of these building blocks on the application performance. For example, for web browsing, the main building blocks will be DNS response times, web server latencies and TCP throughput. It can be noticed that different applications will have several common building blocks and hence having accurate analytical models to translate the impact of these building blocks into application performance will help provide performance monitoring for several applications simultaneously. Further, the network QoS information required to predict application level performance for different users will have several common components. Clearly, such a performance prediction framework provides for a very scalable approach to application level performance monitoring. Moreover, the building block values will also help determine the reasons for less-than-satisfactory application performance. Since most of the data applications employ TCP protocol, arguably the most important building block for such a performance evaluation framework will be predicting the TCP throughput based on network QoS characteristics such as loss rate and delay.

In this chapter, we have focussed on the development and evaluation of analytical models that provide accurate prediction of TCP throughput based on easily measurable network loss and delay characteristics.

## 3.2 Issues in TCP Throughput Prediction

The problem of TCP throughput prediction can be decomposed into the following two problems:

1. How to obtain accurate estimates for network QoS characteristics as observed by individual TCP flows?

2. How to convert estimates for network QoS characteristics into accurate prediction for TCP throughput.

Among the two problems mentioned above, the first problem needs to be investigated from two different yet related angles: what to estimate and how to estimate. The second problem essentially involves using/developing an analytically sound model for predicting TCP throughput that captures TCP protocol's behavior in sufficient details as well as the impact of network QoS characteristics. The analytical model should take as input the QoS characteristics that can be easily and accurately estimated. This is because even the most accurate analytical model will be rendered ineffective if the required input parameter values can not be determined accurately. Finally the analytical model should provide good prediction of TCP throughput for a wide range of scenarios.

### 3.2.1 Estimating Network QoS Characteristics as Observed by Individual TCP Flows

The information needed to predict the throughput of TCP flows consists of *session level* information such as packet size, retransmission timer granularity [5], maximum *congestion window*[5] size, use of *delayed acks* [5], etc., and of *network level* information such as the sequence of successive *round trip times* (RTTs) and of losses. We are interested in the latter category, as session level information can be regarded as chosen a priori while defining representative flows. Further, analytic models of TCP performance use some simplified characteristic of the delay and loss sequences as predictors, rather than the sequences themselves. The important delay characteristics are the average RTT and the variation in RTT. While the RTT value has a direct influence on the achieved TCP throughput, the RTT variation is used in estimating the *retransmission timeout* (RTO) [115] duration. The most important loss characteristic is the average loss rate while the correlation in packet loss process can also have a significant impact on the TCP throughput achieved.

Now we consider how ISPs can estimate the network level QoS characteristics such as loss rate and delay as observed by the individual TCP flow. One simple approach will be to enable *per-flow* monitoring in the edge routers (by using features like *NetFlow* in Cisco routers [105]). We categorize such an approach as *invasive* since it involves actually examining the packets belonging to the flow in order to estimate the QoS characteristics for this and similar flows in the network. Invasive estimation of QoS characteristics suffers from the same problems as application performance measurements. Invasive estimation has serious privacy and security related implications and may even be impossible if the TCP flows are using security features like IPSec [77]. In this case, it will not be possible for the router to uniquely identify the packets belonging to one flow. Per-flow monitoring is computationally expensive and hence significantly slows down the operation of routers. Large scale use of per-flow monitoring will soon run into scalability problems. The above mentioned problems make it impractical to rely on invasive methods for estimating the QoS characteristics observed by individual TCP flows. The failure of invasive methods implies that *we need to estimate QoS characteristics for the TCP flows without looking at the flows themselves*. Such methods can be called *non- invasive* methods. The two most prominent non-invasive methods for estimating QoS characteristics for TCP flows are: Polling SNMP Management Information Base [131, 95] information maintained by routers in the network and probing end-to-end QoS characteristics of the TCP flows by using ping [78] based probe packets. The added advantage of using non-invasive estimation comes from the fact that the most of the IP networks already have a deployed network management infrastructure based on these methods. In the following, we discuss the characteristics as well as pros and cons of using these two mechanisms for non-invasive estimation of QoS characteristics as observed by the TCP flows. This is followed by a simulation based performance analysis of these methods in accurately estimating loss and delay characteristics of TCP flows.

## 3.3 Techniques for Non-invasive Estimation: Polling and Probing

As we discussed in the last section, rather than performing flow level monitoring to estimate QoS characteristics, we would like to instead rely on non-invasive estimation methods. This section focuses on the pros and cons of these non-invasive estimation procedures and their use in estimating parameters of interest in the context of TCP throughput prediction.

### 3.3.1 Probing

Probes can easily be implemented using existing mechanisms such as ping packets sent from ingress towards egress routers. The advantage of such “ping” probes, besides their simplicity, is that they sample the complete end-to-end path. As a result, they provide direct estimates of end-to-end loss probabilities and round-trip times. The disadvantages of ping probes include:

1. Ping packets do no receive same treatment in the network as the ordinary packets. Several routers give an inferior treatment to ping packets. Hence, the QoS characteristics determined with Ping packets may not be a good estimate for actual QoS characteristics observed by TCP flows. Often, ping packets sent from a source to a destination do not follow the same path as the ordinary traffic. Since ping packets have been many times used in *Denial of Service* (DOS) attacks, many network administrators disable any response to ping queries by routers in their network.

2. For any given pair of ingress and egress routers, producing accurate QoS estimates often requires a large number of samples (each probe packet provides one such sample). This can translate into a large overhead or slow convergence time. Further, probe loss estimates will be affected by the number of packets in a round of probing, the inter-probe delay, the frequency of probing rounds and the packet size used in probes.

3. Because probe packets do not necessarily sample the network queues in the same fashion as TCP data packets, the loss probabilities they encounter may also differ from those of TCP applications.

The last disadvantage mentioned above can be partly remedied by sending probe packets according to patterns that emulate the key features of TCP's packet transmission without consuming as much bandwidth as a TCP flow. We experimented with two such probing patterns that provide different degrees of emulation of TCP. The first, called *back to back* probing, simply attempts to capture the well known bursty nature of TCP by sending several ($n$) ping probes back to back in each round of probing. The second approach, called *ack paced* probing, tries to send its $n$ probe packets in each probing round in a manner that mimics TCP's *ack pacing* behavior. Specifically, while in congestion avoidance phase, a TCP flow generally sends 1 (or 2 in case of delayed acks) packet into the network in response to an incoming acknowledgement (ack). Thus, the time interval between arriving acks determines the time spacing between consecutive packets. In addition, after a window worth of acks has been received, the window is increased by one and two back to back packets are sent. Ack paced probing attempts to reproduce this behavior. Every probing round always starts with two back-to-back ping packets to capture the back-to-back transmissions that take place after each window increase. Also the spacing between the replies of these packets gives an idea of the minimum inter-packet spacing currently allowed by the network. In the initial probing round all the remaining probes are also sent back-to-back. However, in subsequent rounds, those probes are spaced based on the average of the current estimate of the minimum inter-packet spacing and the average spacing between replies for packets not sent back to back in the previous round.

### 3.3.2 SNMP Polling

SNMP Polling refers to the periodic querying of SNMP router MIBs to retrieve performance data such as number of transmitted or lost packets and bytes [95]. Polling SNMP

MIBs to estimate QoS characteristics experienced by TCP flows comes with following advantages:

1. Router SNMP MIBs automatically aggregate statistics over time and hence do not require the transmission of numerous (probe) packets into the network.

2. MIB polling is a routine activity in most service provider infrastructures.

The disadvantages of SNMP MIB polling are as follows:

1. SNMP MIB statistics are typically at the interface level, so that the performance measures they track are for the aggregate traffic crossing that interface, and as with probes, those can differ from what individual TCP flows experience.

2. SNMP MIBs provide information specific to a single router (interface), so that generating end-to-end statistics requires polling and combining MIB information from all routers on the path of a flow.

The overhead of MIB polling is primarily dictated by the frequency at which the polling occurs, and there is a trade-off between the associated message and processing overhead, and the accuracy with which changes in network parameters are being tracked.

## 3.4 *Non-invasive* Estimation of *Network level* QoS Parameters

In this section, we examine the capability and limitations of non-invasive methods in accurately estimating the loss and delay characteristics as observed by TCP flows.

### 3.4.1 Estimating *Network level* parameters: Round trip times and Retransmission Timeouts

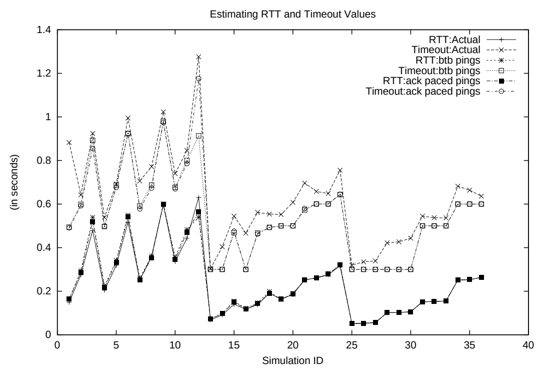
Each reply to a ping probe provides a sample for the round trip time. Simple average of these samples can be used as an estimate for the round trip time duration. The RTT samples can also be used to maintain *smoothed RTT (srtt)* and *smoothed mean deviation in RTT (rttvar)* values [115, 143]. These values along with that of clock granularity ($G$) can be used to estimate 'first' [2] retransmission timeout value ($T0$) using the well-known formula [115, 69]:

$$T0 = srtt + max(G, 4 \times rttvar) \tag{3.1}$$

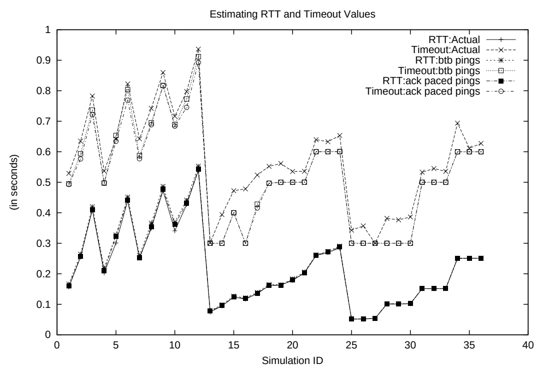
If the retransmission timeout value comes out to be less than 1 second, it is rounded up to 1 second [115]. Hence, it is not crucial to have a very good estimate for *rttvar* unless it will make significant difference to *T0* values.

Next, we present the simulation results regarding the estimation of round trip times and timeout durations. The simulations were performed for single congested router and two congested routers topologies. We direct the reader to sections 3.6.1 and 3.6.2 for details about the simulation and testbed scenarios. Here, we briefly explain some of the terminology we use while presenting the results and our presentation style. The *homogeneous* flow simulations refer to the simulations where all the TCP flows have the same round trip times. *Heterogeneous* simulations refer to the ones where the TCP flows were split in two sets each with different RTT. For 'heterogeneous' simulations, if we have not presented results for both sets of flows, the presented results refer to the flows with larger RTT. In two router simulations, the results are shown for the set of flows that pass through both the congested routers in the topology. In all the result figures, the $x$-axis of the figure corresponds to different simulation/testbed scenarios each one of which has a unique *ID* (details in sections 3.6.1 and 3.6.2).

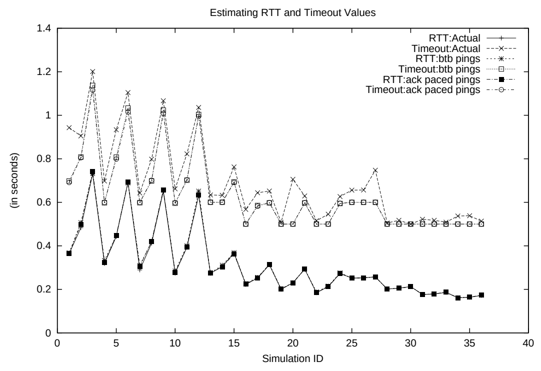
[2]By 'first' retransmission timeout duration, we mean the duration of the first timeout in a sequence of consecutive timeouts. In TCP, the timeout duration doubles with each successive timeout, until it becomes 64 times the original value.



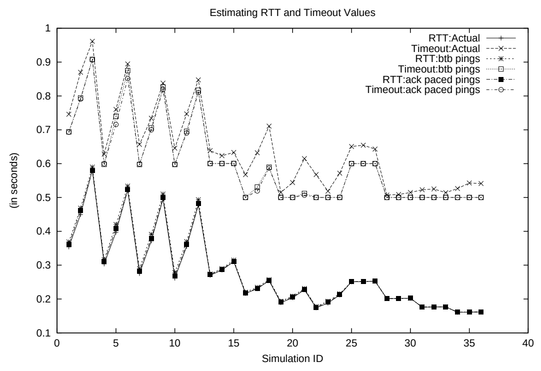
(a) Single droptail router, homogeneous flows Simulations



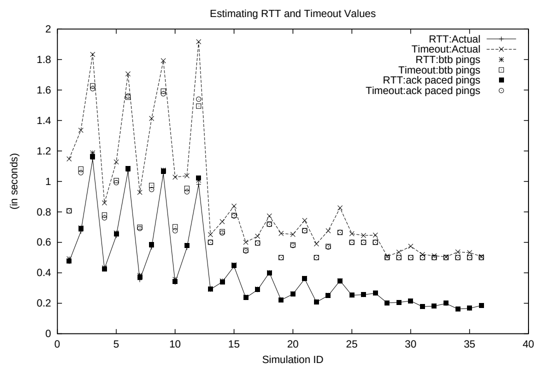
(b) Single RED router, homogeneous flows Simulations



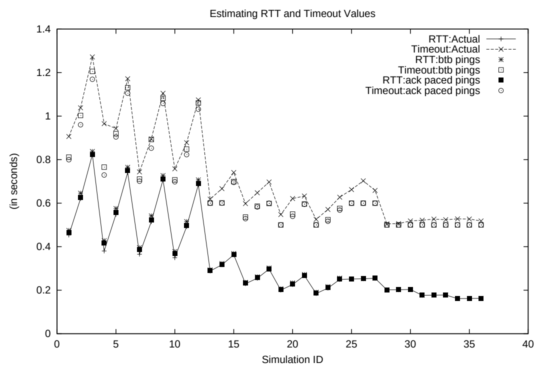
(c) Single droptail router, heterogeneous flows Simulations



(d) Single RED router, heterogeneous flows Simulations



(e) Two droptail routers Simulations



(f) Two RED routers Simulations

Figure 3.1: Simulation results for average round trip time and timeout duration estimation

Figure 3.1 compares across different simulation scenarios the actual values of RTT and $T0$, to their estimates obtained using *back to back pings* and *ack paced pings*. For the purpose of evaluating the effectiveness of ping probes in estimating the RTT variation, these simulations did not enforce 1 second lower limit on $T0$ values. The main conclusion is that in all cases, the estimates produced by either probing schemes[3] are very accurate for RTT and reasonably accurate for $T0$. Note that slight inaccuracy in estimating $T0$ values becomes inconsequential most of the time because the lower limit of 1 second on these values.

Finally, note that while standard MIBs can provide us with loss statistics, extracting estimates for round-trip times using just SNMP MIB data seems difficult except in small topologies. Such estimation will require knowledge of both propagation and queuing delays, and neither quantity is commonly tracked in MIBs. However, an estimate for the queuing delay at an interface can be obtained indirectly by assuming an average packet size and using the information about the length of the output packet queue and the link speed maintained in the *Interfaces* table of SNMP MIB-II[95]. Getting retransmission timeout estimates using SNMP MIBs is still more difficult since it involves tracking the variation in queuing delays on congested router interfaces across the path of the flow. The problem has two dimensions. First, tracking the queuing delay variation will require frequent polling. Secondly, how do we combine the variation on individual interfaces into end-to-end values? However, as noted earlier, often it is possible to assume 1 second as a reasonable estimate for "first" retransmission timeout duration.

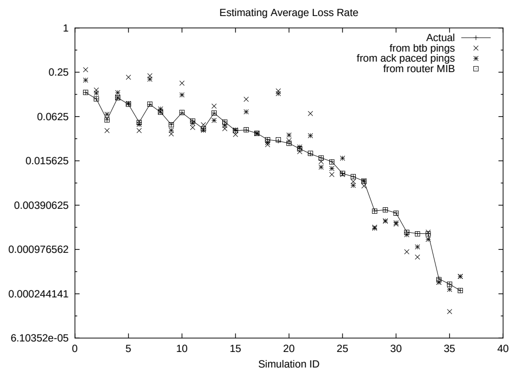
[3]The conclusion also held for several other probing schemes we experimented with, e.g., random probes.

### 3.4.2 Estimating *Network level* parameters: Loss Rates

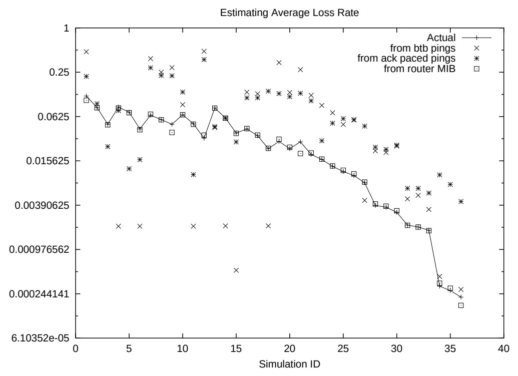
The nature of the packet loss sequence observed by the TCP flows depends on many factors like the path followed by the packets through the network and the competing traffic. With limited *non-invasive* means at our disposal, it is difficult to estimate anything but the simplest characteristics of the actual loss process suffered by a flow. One such characteristic will be the average loss rate suffered by flow. More ambitious endevours would aim at getting an idea about the burstiness or correlation in the packet loss process. In our study, we have focussed on understanding the issues involved in getting reasonable estimates for the average loss rate and a measure of the burstiness in packet losses observed by the TCP flows.

#### Estimating Average Loss Rate

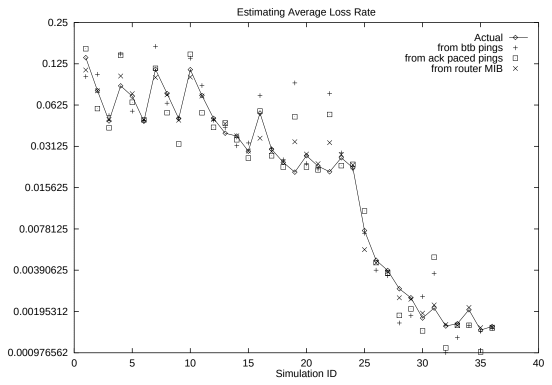
Figure 3.2 shows the sample simulation results illustrating the efficacy of different probing/polling means in estimating the average loss rates. In these simulations, the probing application sent 4 probes (of same size as the packets of TCP flows) in each round of probing and was frequent enough to consume 0.5% of the bottleneck link bandwidth. The most striking feature of these figures is that for the simulations where routers used *droptail* buffer management none of the estimation techniques seems to be working satisfactorily. When *Random Early Drop* (RED) [47] is the buffer management scheme used, the SNMP estimates are quite accurate but the same can not be said about ping probing estimates. In the following, we analyze these results along two dimensions. First, we discuss the difficulties involved in estimating loss rates for droptail routers in comparison with RED routers. After that, we compare the performance of SNMP and ping probing means in estimating average loss rates.



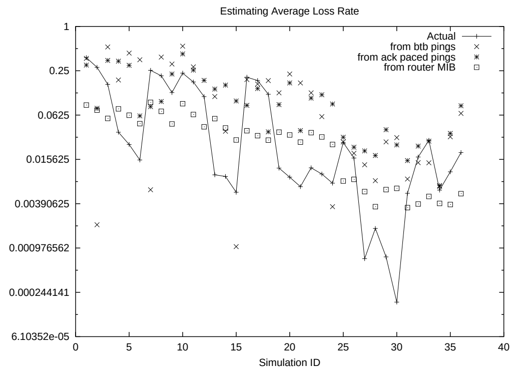
(a) Single RED router, homogeneous flows



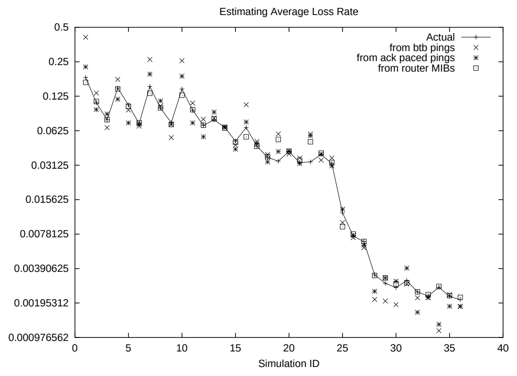
(b) Single Droptail router, homogeneous flows



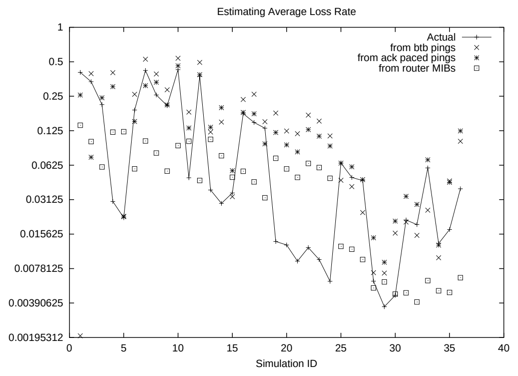
(c) Single RED router, heterogeneous flows



(d) Single Droptail router, heterogeneous flows

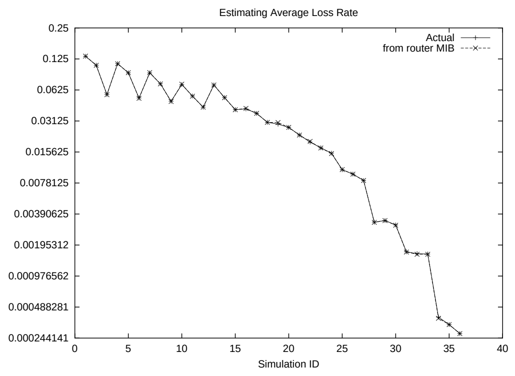


(e) Two RED routers

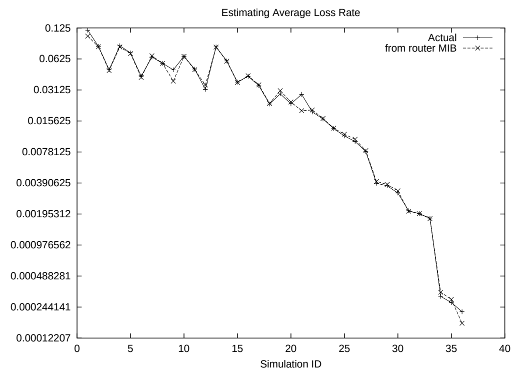


(f) Two Droptail routers

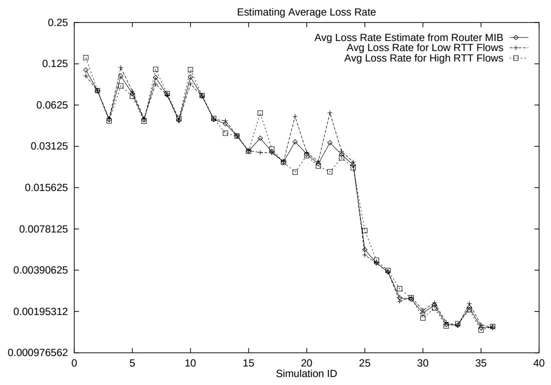
Figure 3.2: Simulation Results for Average Loss Rate Estimation.



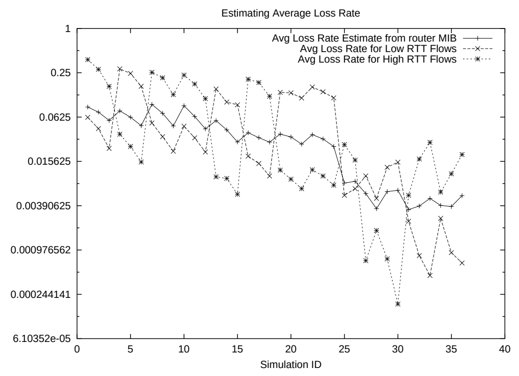
(a) Single RED router, homogeneous flows



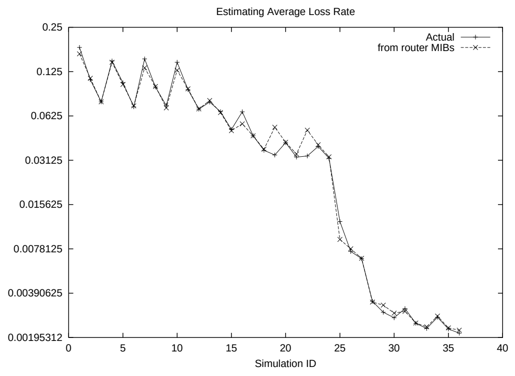
(b) Single Droptail router, homogeneous flows



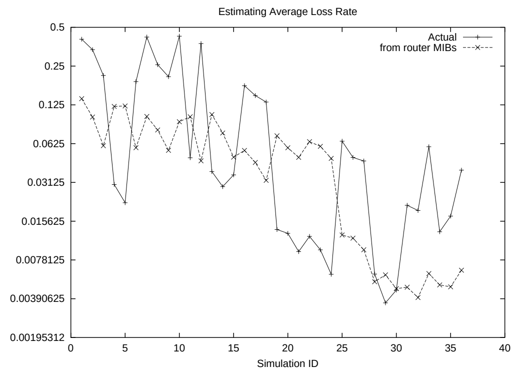
(c) Single RED router, heterogeneous flows



(d) Single Droptail router, heterogeneous flows

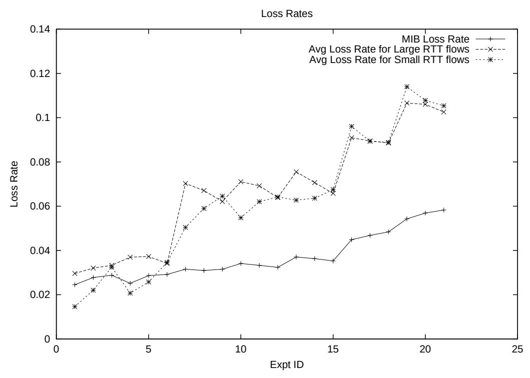


(e) Two RED routers

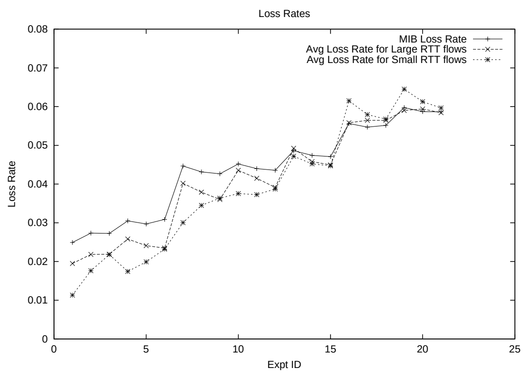


(f) Two Droptail routers

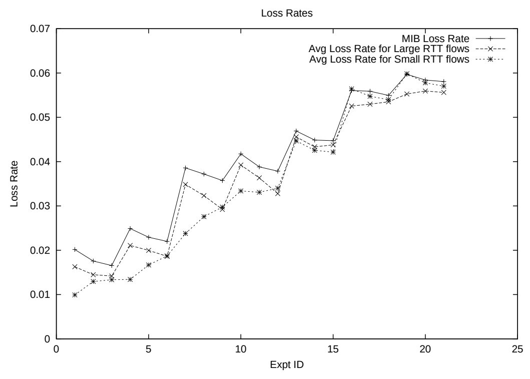
Figure 3.3: Simulation Results for Average Loss Rate Estimation using Router MIB.



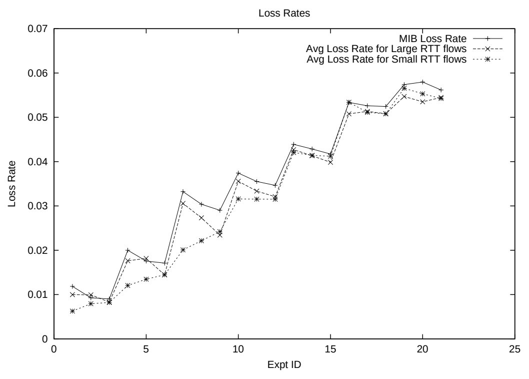
(a) Droptail Experiments (default buffer size: 40 packets)



(b) RED Experiments (buffer 100 packets)



(c) RED Experiments (buffer 500 packets)



(d) RED Experiments (buffer 1000 packets)

Figure 3.4: Testbed Results for Average Loss Rate Estimation using Router MIB.

Random Early Drop uses a moving average for the buffer occupancy to make its packet drop decisions. The arriving packets are dropped randomly with a probability that increases slowly with the increasing average buffer occupancy. On the other hand, Droptail interface does not drop any packet unless there is no more buffer available. The random drop nature of RED policy has the effect that with sufficient buffer sizes the average loss rate observed by an application does not depend too much on the way the application “samples” the interface. But for droptail policy, the “sampling” pattern of the application will have a deciding effect on not only the loss pattern but also the average loss rate. For example, an application sending packets in the bursts may suffer a very different average loss rate than a non-bursty application. As seen in the simulation results in figure 3.2 this translates into huge disparity between the actual loss rates and the estimates given by polling and probing means. In our experimentation with *droptail* policy, we found the problem of getting reasonable *non-invasive* estimates for the average loss rates experienced by TCP flows in a reasonable amount of time quite untractable. The SNMP MIBs provide the loss rates suffered by the aggregate of all the traffic passing through the interface which for *droptail* policy, we observed, are in general quite different from those experienced by individual flows. This was verified to be true by the results obtained from our testbed experiments (Figure 3.4). The estimates obtained from ping probing are also nowhere close to actual loss rates. Specialized probing techniques like the *back-to-back* and *ack-paced* did not help much in case of droptail interfaces. It seems that in spite of all its evils, the only way to get reasonable loss rate estimates for droptail scenarios in a reasonable amount of time is to actually start a long enough TCP flow between the given end points.

After having discussed the problem of estimating average loss rates in droptail scenarios, we turn our attention to RED scenarios where different *non-invasive* means do show some promise. Here we will essentially be comparing different polling and probing means for the accuracy of the loss rate estimates provided by them. Looking again at figure 3.2, we find that the loss rate estimates obtained from router MIBs are much better than those given by probing mechanisms. As can be expected, the *ack-paced* probing generally gives better estimates than the *back-to-back* probing. However, the performance is not quite as satisfactory as that of router MIBs. We experimented with many different probing frequencies and different number of probe packets in each round of probing with little improvement in estimation accuracy. Thus, our simulation results suggest that, for RED buffers, the aggregate loss rates given by router MIBs are much better estimates of actual application level loss rates than the estimates obtained by ping probing. The accuracy of router MIB loss rates in predicting the flow-level loss rates was verified with our testbed experiments (Figure 3.4). In figures 3.3 and 3.4, we have shown the actual loss rates alongwith the aggregate loss rate obtained from router MIBs for different simulations and testbed experiments. It is clear from these figures that when RED is the packet drop policy, the aggregate traffic loss rates given by router MIBs are a good estimate of actual loss rates experienced by long life TCP flows.

An additional issue with using MIB loss rate estimates is that the router MIBs provide information specific to a single router (interface) and generating end-to-end statistics requires combining MIB information from all bottleneck routers on the path of a flow. Generally a TCP flow's path on the Internet is characterized by just one or two bottlenecks (on access links to high capacity backbone). Hence the problem of obtaining end-to-end loss rate is not that severe. Further, in general, the two bottleneck links on the path of a flow will be quite far away from each other and the traffic causing congestion in these links will be quite unrelated. Thus, it may be assumed that the losses at different bottleneck routers on the path of a flow are independent in nature and can be accordingly combined to obtain end-to-end loss rates. Figure 3.2e shows the sample simulation results in this regard. Here the end-to-end loss rate estimate was obtained by combining the MIB loss rates by assuming that the losses at two congested routers are independent in nature. It can be seen that *independent loss* assumption provides quite satisfactory estimates of actual end-to-end loss rates.

**Estimating the Burstiness in Packet Losses**

The problem of estimating the burstiness in the losses suffered by the TCP flows is complicated by a number of factors. With increased use of RED in the routers in the Internet and the fact that most of the TCP flows are marked by small window sizes, it is not clear how bursty the loss process as observed by a single TCP flow actually is. Further, we are restricted to using only non-invasive means in estimating the burstiness. To our knowledge the current SNMP MIBs do not track any kind of *correlation* in the packet losses on an interface. Even if we introduce correlation tracking counters in the router MIBs, it is not clear how can we combine the individual *burstiness* indicators into end-to-end numbers. In the absence of any solution to the above problem, we are restricted to using only ping probing in order to get an estimate for the burstiness of the losses. A typical *correlated* loss model is characterized by two loss rate parameters $p$ and $p'$ applicable in *normal* and *congested* state of the network. We have seen earlier that probing applications do not perform satisfactorily even for estimating the simple average loss rate. They are likely to have lesser success when there is an additional parameter to estimate. One should also consider the fact that since the network will spend much less time in *congested* state than otherwise, the applicable loss rate $p'$ will be much harder to estimate than $p$. In addition to loss parameters $p$ and $p'$, one has to also estimate the duration of the *normal* and *congested* states of the network. Typically, the correlated loss models pursued in previous works make assumptions regarding one or more parameters of the model. e.g. Some times it is assumed that the network continues to be *congested* for the next RTT duration or for the time during which all the packets in the current window are transmitted. It is clear that none such assumption can reasonably be made if we are restricted to *non-invasive* means of estimation.

In spite of the difficulties mentioned above we developed a “correlated” loss model for which non-invasive estimation is theoretically feasible in the sense that the model does not use any *flow-level* characteristics as inputs or assumptions. We evaluated the available probing mechanisms for their accuracy in estimating model parameters. Then we used the loss model to obtain expressions for the steady state TCP throughput. Finally, we evaluated the TCP throughput prediction accuracy of the correlated loss model in comparison to the one using just average loss rate information. The details can be seen in Appendix D. Here we mention the main results obtained from this work:

1. As we suspected, the ping probes did not do a good job in estimating model parameters.

2. The improvement in throughput prediction accuracy by using the “correlated” loss model is not worth the immense trouble one will have to go through to get good estimates for model parameters.

On this somewhat pessimistic note, we conclude this section. The next section discusses our steady state TCP throughput prediction model that uses the average loss rate as the input parameter.

## 3.5 TCP Throughput Prediction Based on Non-Invasively Obtained Parameters

In this section, we start with a survey of different TCP throughput prediction models available in the literature. This is followed by a description of the *Amherst* model which is one of the most popular TCP throughput prediction model. Next we discuss why it is difficult to apply existing TCP throughput prediction models including the *Amherst* model, to the setting where network parameters are assumed to be obtained using only non-invasive procedures. This is then followed by the development of several modifications and enhancements to the *Amherst* model, which allow for reasonably accurate predictions of TCP throughput based only on information obtained through such non-invasive procedures.

### 3.5.1 Previous Work in Analytical Modeling of TCP Protocol

A large set of literature exists on analytic modeling of TCP behavior targeting different environments. The first empirical model for long-term TCP throughput prediction was presented in [94]. This model assumed random packet loss with constant probability $p$ and did not consider the possibility of retransmission timeouts or multiple packet loss leading to multiple reduction in TCP congestion window. This simple model captured the essential basic facts that the long-range TCP throughput is directly proportional to the *Maximum Segment Size* and inversely proportional to the Round-trip Time and the square root of the loss rate. Lakshman and Madhow [85] analyzed the performance of Tahoe [143] and Reno [143] TCP connections in a single bottleneck link environment with finite buffers and large bandwidth-delay product. Though a detailed mathematical analysis of single and multiple connection(s) scenarios was done, this analysis was primarily for the purpose of understanding the properties of TCP/IP rather than developing a throughput prediction tool. The effect of coarse granularity retransmission timeouts was ignored. The authors did point out that Reno TCP might undergo multiple window reductions in face of multiple packet drop caused by the same congestion event and assumed for their analysis that 2 packets are lost for each congestion event. Anurag Kumar [83] presented a stochastic model for the bulk throughput of various TCP versions in an environment where a host on a LAN is transmitting bulk data to a mobile host connected to the lan by a wireless link. The wireless link is characterized by uncorrelated random loss of packets. Padhye et. al. [114] presented a packet-level model, which we refer to as the *Amherst*[4] Model for bulk TCP throughput prediction based on an expression that involves several parameters including average RTT, first timeout duration and the *first packet loss* probability. This model has been shown to result in good prediction of TCP throughput and in fact provided the basic framework for our work. Casetti and Meo [22] analyzed TCP performance by developing separate models for the TCP connections and the network. The TCP source model describes the interaction between application behavior and the TCP protocol and the network model describes the network as a queuing network taking care of aspects like topology, buffers and the link capacities. The TCP source model influences the network model by generating an estimate for the aggregate traffic entering the network while the network model in turn influences the TCP source model by generating average packet loss probability and delays. The stationary behavior of the system is derived through a sequence of successive refinement steps where the two models tune each other. Altman et al. [72] presented a fluid model analysis of TCP throughput under a general loss process. They argued that the Internet exhibits different type of loss processes at different times and their model is able to capture any correlation or any distribution of inter-loss times. Abouzeid et

[4]After University of Massachusetts at Amherst where Padhye and his co-authors developed the model

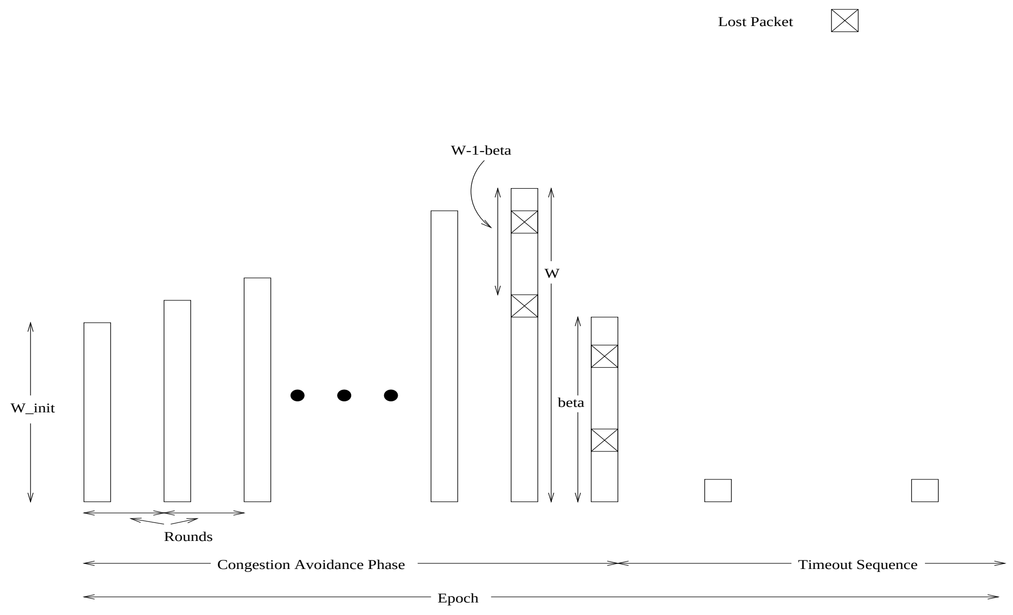


Figure 3.5: A TCP Epoch

al. [122] modeled the performance of single TCP session over a channel characterized by continuous-time two state Markov Chain alternating between a good state and a bad state, where packets are lost in good state with probability 0 and in bad state with probability 1. Cardwell et al. [123] included consideration for connection establishment and initial slow start phase into the model presented in the Amherst Model so as to reflect the behavior of short TCP connections. Yeom and Reddy [147] modeled the performance of TCP flows in differentiated services environment characterized by a reserved rate in addition to round trip time and packet loss probability.

### 3.5.2 Amherst Model for Predicting Bulk Transfer TCP Throughput

In this section, we briefly review the Amherst Model for predicting bulk transfer TCP throughput. The long-term behavior of TCP[5] may be modeled as a sequence of *epochs*

[5]We assume familiarity with TCP Reno Congestion Control [5, 143] and the terminology used therein.

consisting of a number of *rounds* (figure 3.5). An epoch consists of a congestion avoidance phase and a possible timeout sequence. A round is the time duration during which a flow sends a congestion window worth of packets. The receiver sends one ack for every $b$-th packet that it receives (delayed acks [143]), and a round gets over when the first acknowledgement (ack) for a packet sent during the round comes back. It is assumed that a TCP flow is able to send all its congestion window within a round and that duration of a round is equal to the average round trip time observed by the flow independent of the congestion window size. Further, the impact of slow start and connection establishment phases is ignored as the focus is on the long-term behavior of TCP.

During the congestion avoidance phase, the congestion window increases linearly with a slope of $1/b$ packets per round until the first packet loss occurs. With $p$ as the probability of losing a packet before a packet loss actually occurs, the expected number of packets sent in an epoch before the first packet loss is given by:

$$E[\alpha] = \sum_{k=1}^{\infty} (1-p)^{k-1} pk = \frac{1}{p} \tag{3.2}$$

After the loss of the first packet in the epoch, the flow sends $w-1$ more packets before a dup ack for the first lost packet comes back or the packet loss is detected via timeout. Here $w$ is the congestion window at the time of the first packet loss. Let $W = E[w]$ be the expected value of the congestion window size at the time the first packet is lost in an epoch. Then, on average $E[Y] = \frac{1}{p} + W - 1$ packets are sent by a flow during the congestion avoidance phase of an epoch. A simplifying assumption made by the Amherst Model is that irrespective of the number of packets lost in an epoch, the next epoch begins at half the window size reached at the end of the current epoch. With this assumption, an expression for $W$ can be obtained:

$$W = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} \tag{3.3}$$

Furthermore, by making the following two independence assumptions:

1. The maximum window size achieved in an epoch is independent of the number of rounds in the epoch,
2. The duration of a round is independent of the number of packets sent in the round;

an expression for the average duration of the congestion avoidance phase of an epoch, $E[A]$, can be obtained. It is given by $E[A] = (\frac{b}{2}W + 1)RTT$, where $RTT$ denotes the average duration of a round. Hence, in the absence of timeouts, the Amherst Model predicts the steady state throughput of a TCP flow to be:

$$B(p) = \frac{\frac{1-p}{p} + W}{E[A]} = \frac{\frac{1-p}{p} + W}{(\frac{b}{2}W + 1)RTT} \tag{3.4}$$

In the Amherst Model, a timeout will occur if 3 or more packets are not successfully transmitted after the lost packet. Because the model also assumes that after a packet is lost, all the subsequent packets in the round are also lost, it is possible to compute, for a given window size $w$, the probability $\hat{Q}(w)$ of a retransmission timeout:

$$\hat{Q}(w) = min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w}\right). \tag{3.5}$$

$\hat{Q}(W)$ is then taken to be equal to the average timeout probability.

The model next considers the possibility that the packet sent after the retransmission timeout itself gets lost. If this happens, the flow can go through a sequence of timeouts. Every consecutive timeout doubles the timeout duration until it reaches a value equal to 64 times the duration of the first timeout. The expected number of packets sent during the retransmission phase ($E[R]$) and the expected duration of a timeout sequence ($E[Z^{TO}]$) are then given by:

$$E[R] = \frac{1}{1-p} \tag{3.6}$$

$$E[Z^{TO}] = T_0 \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1-p} \tag{3.7}$$

where $T_0$ is the average duration of the first timeout in a sequence of one or more successive timeouts.

After taking into account both timeouts and the possible limit on the maximum congestion window size, $W_{max}$, the final formula for the steady state throughput of a TCP flow given by the Amherst Model is:

$$B(p) = \begin{cases} \frac{\frac{1-p}{p} + W + \hat{Q}(W)\frac{1}{1-p}}{RTT(\frac{bW}{2}+1) + \hat{Q}(W)T_0\frac{f(p)}{1-p}} & \text{if } W < W_{max} \\ \frac{\frac{1-p}{p} + W_{max} + \hat{Q}(W_{max})\frac{1}{1-p}}{RTT(\frac{b}{8}W_{max} + \frac{1-p}{pW_{max}} + 2) + \hat{Q}(W)T_0\frac{f(p)}{1-p}} & \text{otherwise} \end{cases} \tag{3.8}$$

where $f(p)$ is given by equation:

$$f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6 \tag{3.9}$$

### 3.5.3 Motivation for Developing New Models

As we saw previously in Section 3.5.1, there exists an extensive body of literature on analytic modeling of TCP behavior targeting different environments. A direct approach to our goal of converting non-invasive QoS estimates in to TCP throughput prediction would be to transform non-invasively obtained network data into input parameters of the chosen model. We ruled out several analytic models because of the infeasibility of estimating their input parameters. For instance, some models are tailored towards understanding the behavior of TCP protocol in different, many time restricted, environments; some others depending on loss correlations between successive packets of a TCP session can work only with a mechanism that inspects packets on a per-session granularity; finally some models use parameters such as *loss events* (the event of a TCP flow reducing its congestion window) that cannot be estimated from non-invasive network observables. Given such constraints, we tried instead to pick an analytically sound and well-tested model, and see how far we could go in retro-fitting network observables to model inputs. The Amherst model proposed in [114], turned out to be a good starting point for our investigations.

The Amherst Model predicts the throughput of a TCP flow based on an expression that involves several parameters, including average RTT, first time-out duration ($T0$), and more important for our purpose, the probability $p$ of *first* packet loss in an epoch. Correctly estimating this parameter is, therefore, key to an accurate throughput prediction. In [114], this estimation was carried out based on the number of loss indications (triple duplicate acks and time-outs) observed for the flow itself. Reference [46], which also uses the Amherst model, used the "loss event fraction" as its estimate for $p$. The loss event fraction was defined as the ratio of the number of loss *events* to the number of transmitted packets, where a loss event corresponds to one or more packet losses within a roundtrip time. Both of these estimation procedures are difficult to perform based on non-invasive procedures. The identification of loss indications requires flow level awareness, and the estimation of the loss event fraction is difficult because of its dependency on the RTT of an individual flow. Nevertheless, our first attempt was to determine if it was possible to obtain a reasonable estimate for the first packet loss used by the Amherst model, by using only non-invasive procedures such as the ones described in Section 3.4. The approach we took was to develop a procedure for computing the first packet loss $p$ needed by the Amherst model, from the measured overall loss rate $L$ obtained using non-invasive procedures.

**Estimating $p$ for the Amherst Model from the Overall Loss Rate $L$**

The *first* packet loss probability $p$ that the Amherst model uses can be derived from the overall loss rate $L$ by equating $L$ to the ratio of the expected number of packets lost to the expected number of packets sent during an epoch in Amherst Model.

Let $W$ be the final window size achieved in an epoch and $\hat{Q}(W)$ be the probability of the epoch ending in a retransmission timeout. The expressions for $W$ and $\hat{Q}(W)$ are given by [114]:

$$W = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2} \tag{3.10}$$

$$\hat{Q}(w) = min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w}\right). \tag{3.11}$$

Then, the expected number of packets sent during an epoch in Amherst model is given by:

$$E[Y] = \frac{1-p}{p} + W + \hat{Q}(W)\frac{1}{1-p} \tag{3.12}$$

The expected number of packets lost, $N$, in an epoch in the Amherst model is the sum of expected number of packets lost in the congestion avoidance phase plus those lost in the timeout phase. An expression for $N$ can be obtained that involves $p$ and uses the fact that the model assumes that the remaining packets in the round after the first loss are also lost. This gives:

$$N = \begin{array}{l} \left(\sum_{n=1}^{W} \left(W - n + 1 + p\sum_{i=0}^{n-1} i(1-p)^{n-1-i}\right) \frac{(1-p)^{n-1}p}{1-(1-p)^W}\right) \\ +\hat{Q}(W)\left(\frac{1}{1-p} - 1\right) \end{array} \tag{3.13}$$

The first expression in parenthesis corresponds to the average number of packets lost at the end of congestion avoidance phase, while the second expression corresponds to packets lost during timeout retransmissions. The above expression can be simplified to yield:

$$N = W - \frac{1-p}{p} + \frac{(1-p)(1+(1-p)^W)}{1-(1-p)^2} + \hat{Q}(W)\left(\frac{1}{1-p} - 1\right) \qquad (3.14)$$

As a result, the measured overall loss rate $L$ and the the first packet loss probability $p$ used in Amherst model are related through the following expression:

$$L = \frac{N}{E[Y]} \qquad (3.15)$$

It can be shown that the *right-hand side* of equation (3.15) is a monotonically increasing function of $p$. Hence equation 3.15 can be used to numerically compute the value of $p$ for any given $L$.

**The Performance of Amherst Model when *First* Loss Probability is derived From Overall Average Loss Rate**

Unfortunately, deriving *first* loss probability from average loss rate did not prove to be successful. Performing such a mapping required relating the observed number of losses to the number of lost packets (after the first loss) given by the loss model used in the Amherst model. Because the Amherst model assumes that, subsequent to the first packet loss, all the remaining packets in the window are also lost, the resulting *inversion* of the observed loss rate $L$ into a first packet loss probability $p$, is highly inaccurate. As a result and as shown in Figures 3.6 and 3.7, the resulting throughput estimates are also inaccurate. Note that this assumption regarding packet losses is of very limited consequence[6] in the environment

[6]As illustrated in [114], the Amherst model gives reasonably accurate TCP throughput estimates.

assumed by the Amherst model, i.e., when an accurate estimate is readily available for the first packet loss probability $p$. This is, however, not true when we need to derive an estimate for $p$ based on the overall observed loss probability. In such a setting, the loss model used for relating these two quantities is of significant importance.

### 3.5.4 A New Analytical Model for TCP Throughput Prediction with *Non-invasive* estimation of QoS Parameters

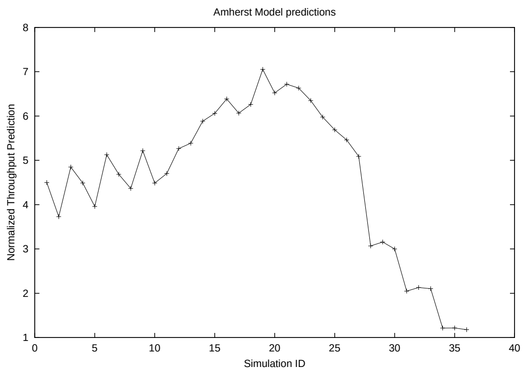
Given the failure of Amherst model to work with non-invasively determinable loss characteristics, our next step was to determine how to modify the Amherst model, in order to use different loss models, i.e., models that rely on parameters that can be estimated using non-invasive procedures, such as the one based on random Bernoulli losses (henceforth called *random loss model*).

In the random loss model, packets are lost randomly with a loss probability, which we denote as $p$. For such a loss model, the probability $P(i, W)$ that after the first packet loss in an epoch, a total of $i$ packets, including the first loss, are lost in the following window of size $W$ is given by:

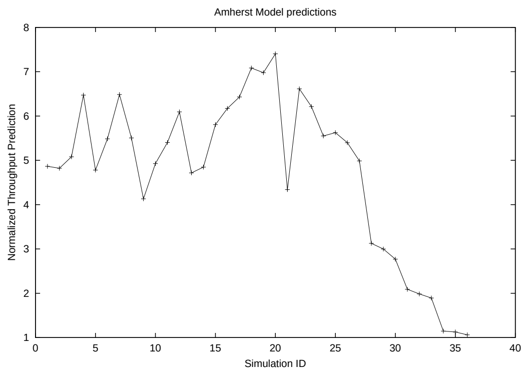
$$P(i, W) = \begin{pmatrix} W-1 \\ i-1 \end{pmatrix} p^{i-1}(1-p)^{W-i}, \tag{3.16}$$

where the window size $W$ is restricted to integer values.

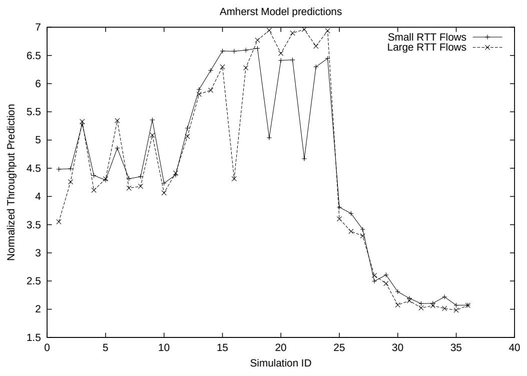
Before proceeding with the derivation of these models, we point to two important modifications we introduce to the approach used by the Amherst model. Our motivation for making these modifications was to further improve the accuracy of the models we were constructing. In particular, we felt that two simplifying assumptions used in the Amherst model might be responsible for some loss in accuracy in certain scenarios, e.g., in high loss environments.



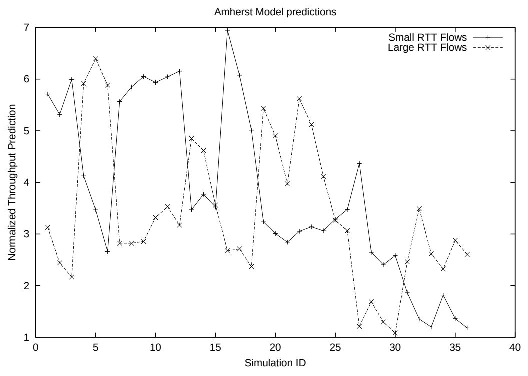
(a) Homogeneous RTT Flows Single RED Router Simulations



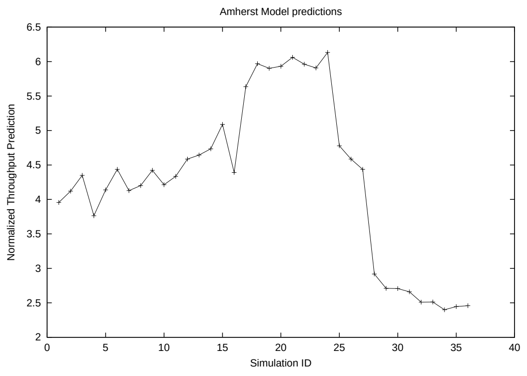
(b) Homogeneous RTT Flows Single Drop-tail Router Simulations



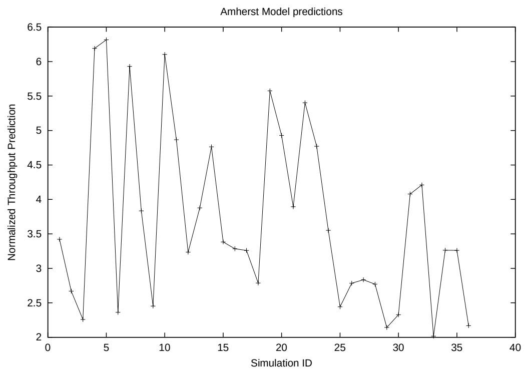
(c) Heterogeneous RTT Flows Single RED Router Simulations



(d) Heterogeneous RTT Flows Single Drop-tail Router Simulations

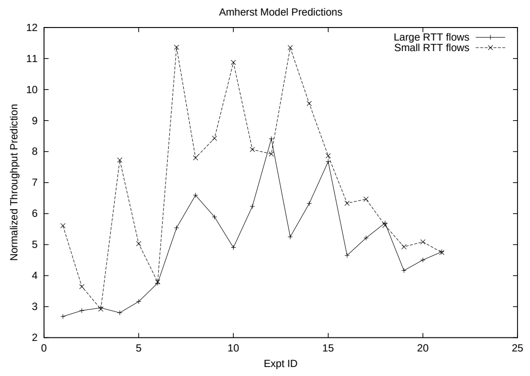


(e) Two RED Routers Simulations

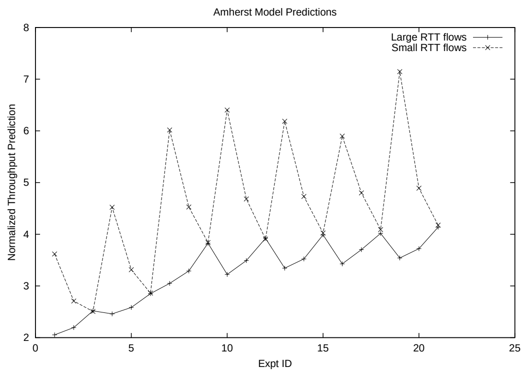


(f) Two Droptail Routers Simulations

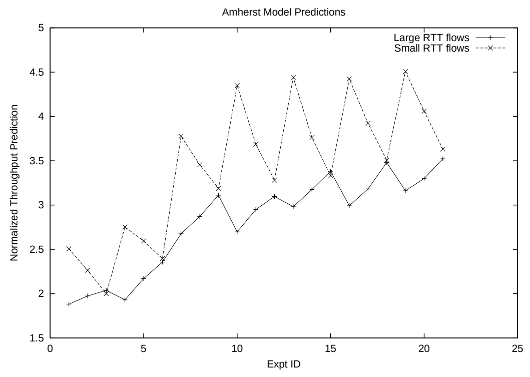
Figure 3.6: Performance of the Amherst model with first loss probability estimated from average loss rate. (Simulation results; The predictions normalized wrt to actual average throughput.)



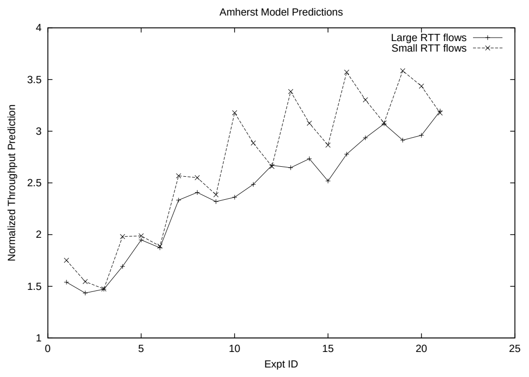
(a) Droptail (40 pkt buffer) Experiments



(b) RED (100 pkt buffer) Experiments



(c) RED (500 pkt buffer) Experiments



(d) RED (1000 pkt buffer) Experiments

Figure 3.7: Performance of the Amherst model with first loss probability estimated from average loss rate.(Testbed results; The predictions normalized wrt to actual average throughput.)

The first modification we targeted was in the computation of the probability of retransmission time-out. In the Amherst Model, a retransmission time-out is avoided if three duplicate acks are received after the first lost packet, and the corresponding probability is computed as the probability of being able to successfully transmit 3 or more packets after the first packet loss. Because of the particular loss model used by the Amherst model (after the first packet loss, all remaining packets in the window are assumed lost), this computation is somewhat inaccurate. Our goal was, therefore, to develop a more precise model for computing the probability of retransmission time-out. One that would incorporate both the *size* $W$ of the congestion window at the time of the first loss, and the likelihood of losing a certain number of packets after the first packet loss. This required a careful enumeration of the different possible loss scenarios for a given window size, and this is treated in details in Appendix A. The main result of this investigation is summarized in equation (3.17), which indicates that a loss of 3 or more packets in the window (typically) leads to a retransmission timeout.

$$P_{TO}(W) = \begin{cases} 1 & \text{if } W < 4 \\ 1 - P(1,W) & \text{if } W < 10 \\ 1 - P(1,W) - P(2,W) & \text{otherwise} \end{cases} \tag{3.17}$$

where $P_{TO}(W)$ is the probability that a timeout will take place when $W$ is the window size at the time of packet loss. As before, $P(n,W)$ represents the probability of losing $n$ packets out of a window of $W$ packets, starting with the first lost packet.

The second modification we introduce was in the determination of the initial congestion window at the beginning of an epoch. In the case of the Amherst Model, this initial window is taken to always be half of the window size at the end of the previous epoch, irrespective of the actual number of losses that trigger the end of the epoch. Such an assumption is likely not to be much of a concern for SACK [93] implementations of TCP stack. However, in TCP Reno implementations, which still comprise about $60\%$ of currently deployed implementations [4], the congestion window can be reduced by more than a half depending on the number of losses. This is significant, as it can be shown that for the *same* total number of packets transmitted in an epoch, the duration of the epoch, i.e., the number of rounds, is larger when the initial window size is smaller. This is illustrated in Figure 3.8. The figure displays two scenarios where the congestion window increases with a slope $1/b$ from an initial value $W_{i1}$ $(W_{i2} > W_{i1})$ to a final value $W_{f1}$ $(W_{f2} > W_{f1})$. The duration of the epochs in two scenarios are $X + Z$ and $Y + Z$, respectively, and showing that $X + Z > Y + Z$ amount to showing that $X > Y$. Because the numbers of packets sent in each epoch are assumed to be the same, $Area1$ is equal to $Area2$, which therefore implies that

$$\begin{aligned} Area1 &= Area2 \\ \Rightarrow \frac{X^2}{2b} + XW_{i1} &= \frac{Y^2}{2b} + YW_{f1} \end{aligned}$$

If $X < Y \Rightarrow \frac{X^2}{2b} < \frac{Y^2}{2b} \Rightarrow XW_{i1} > YW_{f1}$, which is impossible since $W_{i1} < W_{f1}$ and we assumed $X < Y$.

The implication of this result, is that always starting with the largest possible initial window size, as is done in the Amherst Model, can translate in over-estimates of the actual throughput, because it under-estimates the amount of time needed to transmit a given number of packets. Hence, it is desirable to develop a model that can relate the initial window size to the final window size in the previous epoch and the number of losses that ended the epoch.

For that purpose, we assume, as in the Amherst Model, that the steady state of a flow can be characterized by a sequence of similar epochs, with each epoch starting with a window

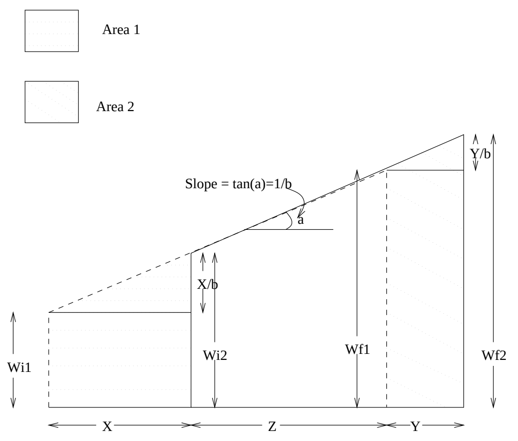


Figure 3.8: Higher initial window increases the number of rounds in an epoch

of size $W_i$ and ending with a window of size $W_f$. Let $Q(n, W_f)$ be the probability that $W_i$ equals $\frac{W_f}{2^n}$. We assume that $W_i$ never goes below $\frac{W_f}{8}$, so that as derived in Appendix B, we obtain the following expressions for $Q(n, W_f), n = 1, 2, 3$:

$$Q(1, W_f) = \begin{cases} 1 & \text{if } W_f < 4 \\ P(1, W_f) & \text{otherwise} \end{cases} \tag{3.18}$$

$$Q(2, W_f) = \begin{cases} 1 - Q(1, W_f) & \text{if } W_f < 10 \\ P(2, W_f) & \text{otherwise} \end{cases} \tag{3.19}$$

$$Q(3, W_f) = 1 - Q(1, W_f) - Q(2, W_f) \tag{3.20}$$

where as before, $P(n, W_f)$ is the probability of losing $n$ packets out of a window of $W_f$ packets starting with the first lost packet. Thus, the initial window size $W_i$, instead of always being set to $\frac{W_f}{2}$, is given by :

$$W_i = \sum_{n=1}^{3} Q(n, W_f) \frac{W_f}{2^n} \tag{3.21}$$

In the next section, we bring together the two modifications we have just outlined and the random loss model, to generate a new TCP throughput prediction model. As stated before, our goal is to develop a modified model that can operate on the basis of global network performance parameters that can be estimated using non-invasive procedures.

**A New Model for TCP Throughput Prediction**

In this section, we present a 'cyclical' model for bulk transfer TCP throughput prediction. By 'cyclical' we mean that the steady state of a Reno TCP flow can be characterized as a sequence of *epochs* during which the flow increases its congestion window linearly from an initial value $W_i$ to a final value $W_f$ with a slope of 1 packet per $b$ rounds. Here a *round* corresponds to the time during which the TCP flow sends a congestion window worth of packets and $b$ is the number of packets received before a TCP destination sends an ack back to the source. As in the Amherst Model, we assume that the duration of a round is independent of the congestion window size, and that an epoch consists of a congestion avoidance phase possibly followed by a timeout phase. For simplicity, we ignored packets sent during the slow start and fast retransmit phases, although the latter were taken into consideration when computing time-out probabilities.

The number of packets sent in the congestion avoidance phase of an epoch is determined by the packet loss probability $p$. Starting from the beginning of an epoch, let the $\alpha$th packet be the first one to be lost. The returning acks of the preceding successfully transmitted packets in the window will allow the TCP flow to send $W_f - 1$ more packets before the packet loss is detected. Thus, the total number of packets sent in the congestion avoidance phase of the epoch is given by $Y = \alpha + W_f - 1$. Now, the probability that $\alpha = k$ is equal to the probability that $k - 1$ packets were successfully transmitted before a loss occurs, which for the random loss model we consider is given by

$$P[\alpha = k] = (1-p)^{k-1}p, k = 1, 2, ... \tag{3.22}$$

Thus, the expected value of $\alpha$ is

$$E[\alpha] = \sum_{k=1}^{\infty}(1-p)^{k-1}pk = \frac{1}{p} \tag{3.23}$$

As a result, the expected number of packets sent in the congestion avoidance phase of an epoch is:

$$E[Y] = \frac{1-p}{p} + W_f \tag{3.24}$$

The number of packets sent in the congestion avoidance phase of an epoch can also be written in terms of $W_i$ and $W_f$. After the first lost packet, the TCP flow sends $W_f - 1$ more packets before the packet loss is detected. Some of these packets are sent in the same round as the first lost packet, and the remaining $\beta$ packets constitute what we term another *short round*. Therefore the total number of packets sent in the congestion avoidance phase of an epoch can also be written as:

$$Y = \frac{b}{2}(W_f + W_i)(W_f - W_i + 1) + \beta \tag{3.25}$$

The expected value of $\beta$ can be derived as follows. If $\beta$ packets are sent in the last short round, the position of the first lost packet in the penultimate round was $\beta + 1$. The probability that the first packet is lost at position $\beta + 1$ given that at least one packet was lost in the window of size $W_f$ is equal to $\frac{p(1-p)^{\beta}}{1-(1-p)^{W_f}}$. Therefore, the expected value of $\beta$ is given by:

$$E[\beta] = \sum_{i=0}^{W_f-1} i \frac{p(1-p)^i}{1-(1-p)^{W_f}} = \frac{1}{p} - \frac{1+(W_f-1)(1-p)^{W_f}}{1-(1-p)^{W_f}} \tag{3.26}$$

Thus, the expected number of packets sent in the congestion avoidance phase of an epoch can be expressed as:

$$E[Y] = \frac{b}{2}(W_f + W_i)(W_f - W_i + 1) + \frac{1}{p} - \frac{1+(W_f-1)(1-p)^{W_f}}{1-(1-p)^{W_f}} \tag{3.27}$$

Equations (3.24) and (3.27) provide two different ways for expressing the total number of packets sent in the congestion avoidance phase of an epoch. By equating these two expressions we get:

$$\begin{aligned} \frac{1-p}{p} + W_f &= \frac{b}{2}(W_f + W_i)(W_f - W_i + 1) + \frac{1}{p} - \frac{1+(W_f-1)(1-p)^{W_f}}{1-(1-p)^{W_f}} && (3.28) \\ \Rightarrow W_f &= \frac{b}{2}(W_f + W_i)(W_f - W_i + 1)(1-(1-p)^{W_f}) && (3.29) \end{aligned}$$

Equation (3.29) together with equation (3.21) can be used to eliminate the unknown $W_i$ and express $W_f$ as a function of $p$. It can be shown (Appendix C) that for a given $p$, the resulting equation admits a unique solution for $W_f$. Once $W_f$ is known, $W_i$ can be computed from equation (3.21), and the expected number of packets $E[Y]$ sent during the congestion avoidance phase can, therefore, be computed from equation (3.27). It now remains to compute the duration of an epoch in order to be able to predict the throughput of a TCP flow.

An epoch consists of a congestion phase possibly followed by a retransmission timeout phase. Let $X(W_i, W_f) = b(W_f - W_i + 1)$ be the number of rounds during which the congestion window increases from its initial value $W_i$ to its final value $W_f$ with a slope of 1 packet per $b$ rounds. Thus, after including the final short round, the duration of the congestion avoidance phase of an epoch is given by $(X(W_i, W_f) + 1)RTT$ where $RTT$ is the average duration of a round.

From equation (3.17), we know how to compute $P_{TO}(W_f)$, the probability that a congestion avoidance phase is followed by a retransmission timeout phase when the final congestion window is $W_f$. It, therefore, only remains to compute the number of packets that may be transmitted during this phase, as well as its duration. As in the Amherst Model, the probability that $R = k$ packets are sent in the timeout phase is same as the probability that the first $k - 1$ packets sent in the timeout phase are lost and the $k$th packet is successfully transmitted. Thus,

$$P[R = k] = p^{k-1}(1 - p) \tag{3.30}$$

Thus, the expected number of packets sent in the timeout phase is

$$E[R] = \sum_{k=1}^{\infty} kP[R = k] = \frac{1}{1 - p} \tag{3.31}$$

Similarly, the expected duration of the timeout phase can also be derived following the method put forth in the Amherst Model:

$$E[Z^{TO}] = T_0 \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1 - p} \tag{3.32}$$

where $T_0$ is the average duration of the 'first' timeout period.

Based on the above equations, we are now in a position to compute the steady state throughput of a TCP flow (in packets per second), based on the following expressions:

$$B(p) = \frac{E[Y] + P_{TO}(W_f)E[R]}{(X(W_i, W_f) + 1)RTT + P_{TO}(W_f)E[Z^{TO}]} \tag{3.33}$$

The remaining "final touch" required to our model, is to incorporate the impact of window size limitations. This is the topic of the last section, which follows essentially the steps outlined in [114] with minor differences to account for the modifications we introduced.

**The Impact of Window Size Limitation**

Let $W_{max}$ be the maximum permissible congestion window size and $W_u$ be the expected value of the maximum congestion window size achieved in an epoch without any limitation on the window size,. If $W_u < W_{max}$, we assume that $W_{max}$ has no impact on the long term average of the TCP throughput, and the formula given in equation 3.33 holds with $W_f = W_u$.

If $W_u \geq W_{max}$, we assume that $W_f = W_{max}$. In this case, the congestion avoidance phase consists of two parts. In the first part, the congestion window increases linearly from its initial value to $W_{max}$. In the second part, the congestion window remains constant at $W_{max}$ until the round where the first loss occurs. Let $U$ and $V$ be the expected number of rounds in these first and second parts, respectively. Since, the congestion window increases from its initial value to $W_{max}$ in $U$ rounds with a slope of 1 packet per $b$ rounds, and since the initial window has value $\sum_{m=1}^{3} Q(m, W_{max})\frac{W_{max}}{2^m}$, $U$ is given by:

$$U = b\sum_{m=1}^{3} Q(m, W_{max})(W_{max} - \frac{W_{max}}{2^m}) \tag{3.34}$$

The expected number of packets $Y_U$ sent during the first $U$ rounds is then given by:

$$Y_U = \frac{b}{2}\sum_{m=1}^{3} Q(m, W_{max})(\frac{W_{max}}{2^m} + W_{max} - 1)(W_{max} - \frac{W_{max}}{2^m}) \tag{3.35}$$

In the next $V$ rounds, the size of the congestion window remains equal to $W_{max}$, so that a total of $V \times W_{max}$ packets are sent. As before, this is followed by the final short round where $E[\beta]$ packets are sent. Since, the expected number of packets sent during the congestion avoidance phase is $\frac{1-p}{p} + W_{max}$, $V$ can be written as:

$$V = \frac{1}{W_{max}} \left( \frac{1-p}{p} + W_{max} - Y_U - E[\beta] \right) \tag{3.36}$$

Thus, the expected duration of the congestion avoidance phase in this case will be $RTT(U + V + 1)$ and equation (3.33) can be rewritten as:

$$B(p) = \frac{E[Y] + P_{TO}(W_{max})E[R]}{RTT(U + V + 1) + P_{TO}(W_{max})E[Z^{TO}]} \tag{3.37}$$

Before concluding the section, we reiterate the differences between the Amherst Model and the throughput prediction model derived above:

1. The required input loss rate for the new model is simply the average loss rate effective during the lifetime of the TCP Reno flow.

2. The new model calculates the retransmission timeout probability and initial congestion window size in an epoch based on the final window size in the previous epoch and the number of losses that ended the epoch.

Now that we have completed our derivation of a TCP throughput prediction model that relies on network parameters derived using non-intrusive estimation procedures, it remains to evaluate the accuracy achieved by this new model. This is the topic of the next section.

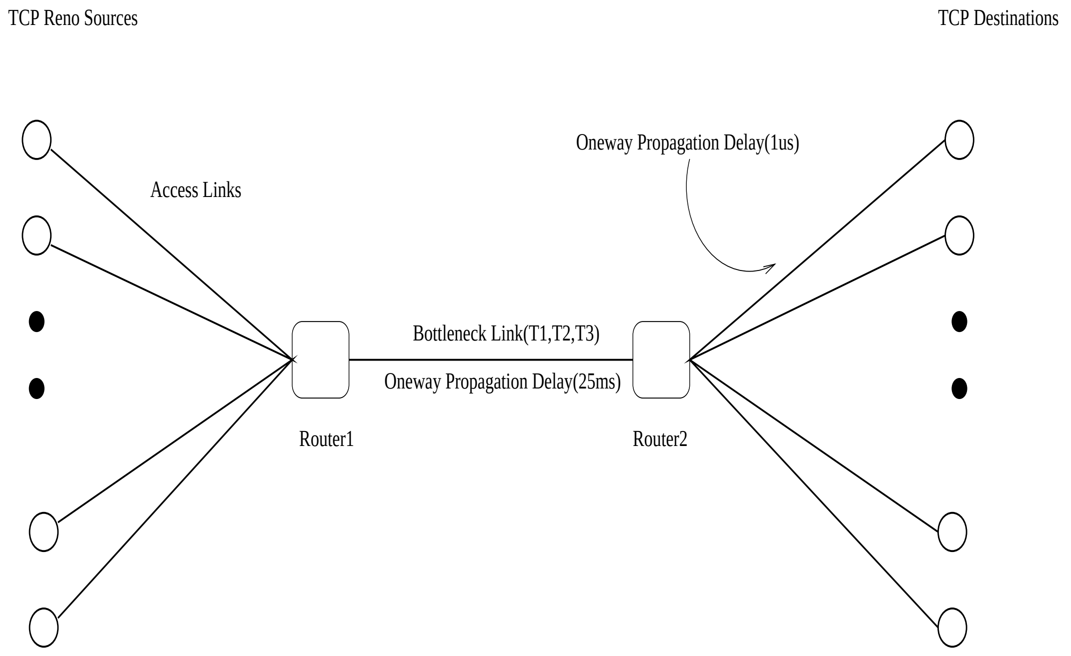
## 3.6 Performance Evaluation

The throughput prediction model described in the previous section was evaluated thoroughly for a wide range of scenarios using both NS2 [107] simulations and testbed experiments. Simulations allowed us to perform experimentation for a wide range of scenarios which is not possible in a resource restricted testbed. However, simulations do not always faithfully reproduce the “real world” behavior . Hence our approach was to perform comprehensive simulations for a wide range of scenarios and to verify the results with experimentation over a testbed. The consistency among the simulation and testbed results increased our confidence in the validity of the results in the “real world”. In the following paragraphs, we describe the simulation and testbed setups followed by presentation and discussion of results.

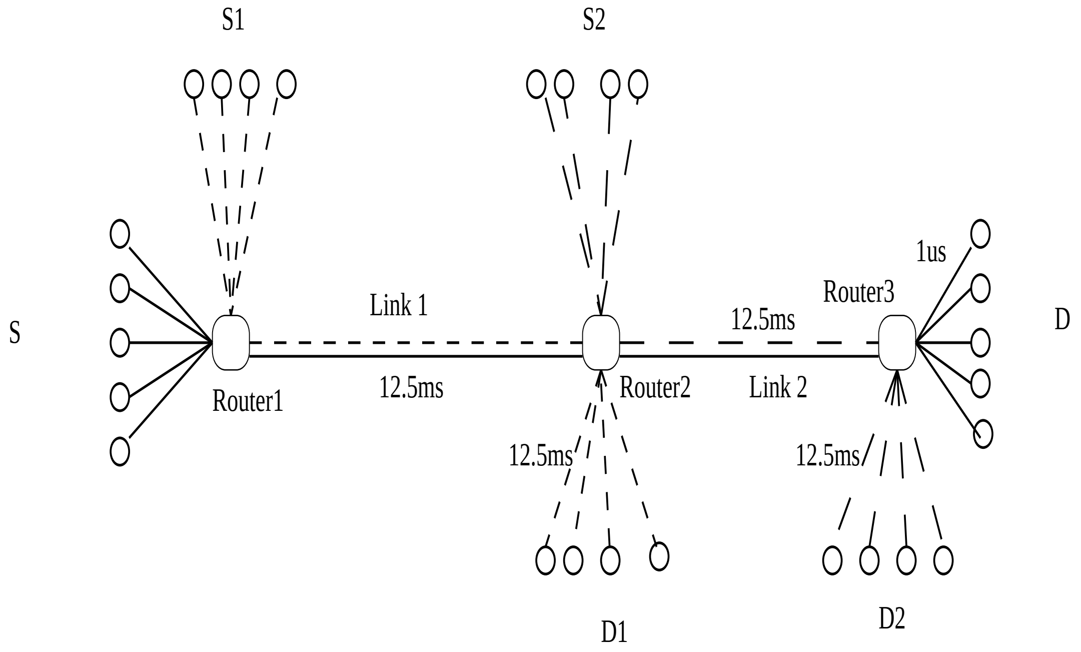
### 3.6.1 Simulation Experimental Setup

In these paragraphs we present the simulation scenarios and parameters used in our performance evaluation and motivate their choice in terms of the coverage they provide.

A first dimension that needs to be considered, in particular in the context of estimating loss probabilities, is the number of routers/links that are contributing to the overall loss probability. Typically, a TCP flow over the Internet will experience bottlenecks on one or both access links to the high capacity core. Thus, we expect single bottleneck and two bottleneck scenarios to represent the “common case” scenarios and use them in our simulations (Figure 3.9). Other simulation parameters that can be varied, include the number and composition of flows, i.e., whether all flows have same RTT (homogeneous) or not (heterogeneous), the link bandwidth, the propagation delays associated with each link, the buffer management policy and the size of the buffers at each router.


TCP Reno Sources
TCP Destinations
Access Links
Oneway Propagation Delay(1us)
Bottleneck Link(T1,T2,T3)
Oneway Propagation Delay(25ms)
Router1
Router2

(a) Single Congested Router



(b) Two Congested Routers

Figure 3.9: Simulation Configurations

The simplest configuration involves a single congested router with homogeneous flows. It represents a baseline case for which any model under consideration needs to perform well. The remaining variables for this base model are the number of TCP flows, the bottleneck link bandwidth, the propagation delay, and the buffer size at the router. The speed of the access links were set to the same value as the bottleneck link, as this additional parameter was found to have hardly any impact. In particular, a higher speed access link will not have much effect, as the transmission of TCP packets is typically triggered by the receipt of an ACK, and ACKs are paced based on the speed of the congested link.

The number of TCP flows was fixed at 48[7] and the bottleneck link bandwidth was set to either one of three possible values T1 (1.544 Mbps), T2 (6.132 Mbps) and T3 (44.736 Mbps). All flows were TCP Reno flows with default congestion window sizes (i.e. 64 packets) and a timer granularity of 100 ms. Further, TCP destinations acked all packets as soon as received. The round trip time values used in the simulations were 50, 100, 150 and 250 ms, and the buffer size was chosen to be either 50, 100, or 200 packets. All flows were assumed to have the same packet size of 500 bytes. This choice of values provides a reasonably broad coverage of the possible range of parameters that one can expect in practice. In order to facilitate the presentation of results for the different possible variations of parameters, simulations were numbered according to their combination of parameters. This numbering is summarized in Table 3.1. Based on this numbering, the simulation parameters for a particular simulation can be ascertained from its ID. e.g. the simulation ID 31 refers to the simulation with bottleneck bandwidth T3, round trip propagation delay of each flow 150 ms and a buffer of 50 packets at the congested router.

[7]All the simulations (homogeneous/heterogeneous single congested router and two congested routers) were repeated for 96 flows to make sure that the number of flows in the simulations are not a factor influencing the results. The 96 flow results were similar to 48 flow results.

We performed simulations with both Droptail and RED as the buffer management policy. The *minth* and *maxth* thresholds chosen for RED simulations were 0.2 and 0.8 times the total buffer size, respectively. Similarly, the maximum packet drop probability was set to 0.1, and the weight used to calculate the average queue length was taken to be 0.002.

The first extension to this set of base scenarios was to introduce differences in RTTs across flows. In particular, the 48 flows were partitioned in two equal sets, with each set being given a different RTT (*RTT1* and *RTT2*). The pair of values used for *RTT1/RTT2* were 140 ms/160 ms, 125 ms/175 ms, 100 ms/200 ms and 50 ms/250 ms. The main motivation behind using different RTT values was to evaluate the performance of the throughput prediction models and of the parameter estimation techniques, as the RTT disparity between the two groups of flows increased.

The next extension was to consider the two congested routers scenario of Figure 3.9b, for which most of the previous combinations were repeated. In the two congested routers scenario, a set *S-D* of 24 Reno TCP flows with the same RTT crosses the two congested router links, *link1* and *link2*. Both *link1* and *link2* remain congested as a result of traffic generated by flow sets *S1-D1* and *S2-D2*, respectively. Each one of these sets consists of 24 Reno TCP flows, again with the same RTTs. In the scenarios with heterogeneous RTTs, flows within the set *S-D* were assigned an RTT equal to *RTT2*, while flows in the sets *S1-D1* and *S2-D2* were assigned the same RTT of *RTT1*.

Finally, all simulations were run for a period of 256 seconds. The basic reason for selecting this duration was that we wanted all flows to achieve their steady state behavior, And it was observed that after 256 seconds of simulation run, the standard deviation in the throughput of different flows with the same RTT became negligible (Section 3.6.5).

| Simulation ID | Bottleneck bandwidth | Round-trip Propagation Delay | Buffer at Congested Router |
|---|---|---|---|
| 1-12<br>13-24<br>25-36 | T1<br>T2<br>T3 | 50,100,150,250 ms | 50,100,200 packets |

Table 3.1: IDs for Single Congested Router, Homogeneous RTT Simulations

| Simulation ID | Bottleneck bandwidth | Round-trip Propagation Delay RTT1/RTT2 | Buffer at Congested Router |
|---|---|---|---|
| 1-12<br>13-24<br>25-36 | T1<br>T2<br>T3 | 50/250 ms, 100/200 ms,<br>125/175 ms, 140/160 ms | 50,100,200 packets |

Table 3.2: IDs for Single Congested Router, Heterogeneous RTT / Two congested routers Simulations

In addition, in order to avoid synchronization among flows, flows were started at random times within the first second of simulation run.

### 3.6.2 Testbed Experimental Setup

Figure 3.10 shows the testbed setup. The setup consisted of two Cisco 3660 routers (running IOS 12.0) connected by a 100 Mbps link, an Agilent Router Tester box for generating background traffic and two PCs running FreeBSD 5.0 and Linux 2.4.2 acting as the source and destination for TCP flows. An interface on Linux machine was assigned two IP addresses (10.0.0.66 and 10.0.0.67) which acted as two destinations for the TCP flows starting from FreeBSD machine. The FreeBSD machine ran *Dummynet* [120] which allowed introduction of delays in the path of packets going to a specific destination. Thus the flows going to 10.0.0.66 and 10.0.0.67 addresses had propagation delays of R1 and R2

| | |
|---|---|
| TCP Flavor | Reno |
| Max CWND, Recv Window | 64 packets |
| Delayed Acks | No |
| Timer Granularity | 100 ms |
| Simulation Time | 256 s |
| Packet Size | 500 bytes |
| Bottleneck Bandwidth | T1(1.544 Mbps),T2(6.132 Mbps), T3(44.736 Mbps) |
| Round Trip Propagation Delays (R1/R2) | 50/250 ms, 100/200 ms 125/175 ms, 140/160 ms |
| Buffer Size (x) at Congested Router | 50,100,200 packets |
| Buffer Management Policy | RED (0.2x/0.8x/0.1, exp weighing const 0.002) |
| Access Link Bandwidth | Same as Bottleneck |

Table 3.3: Common Simulation Parameters

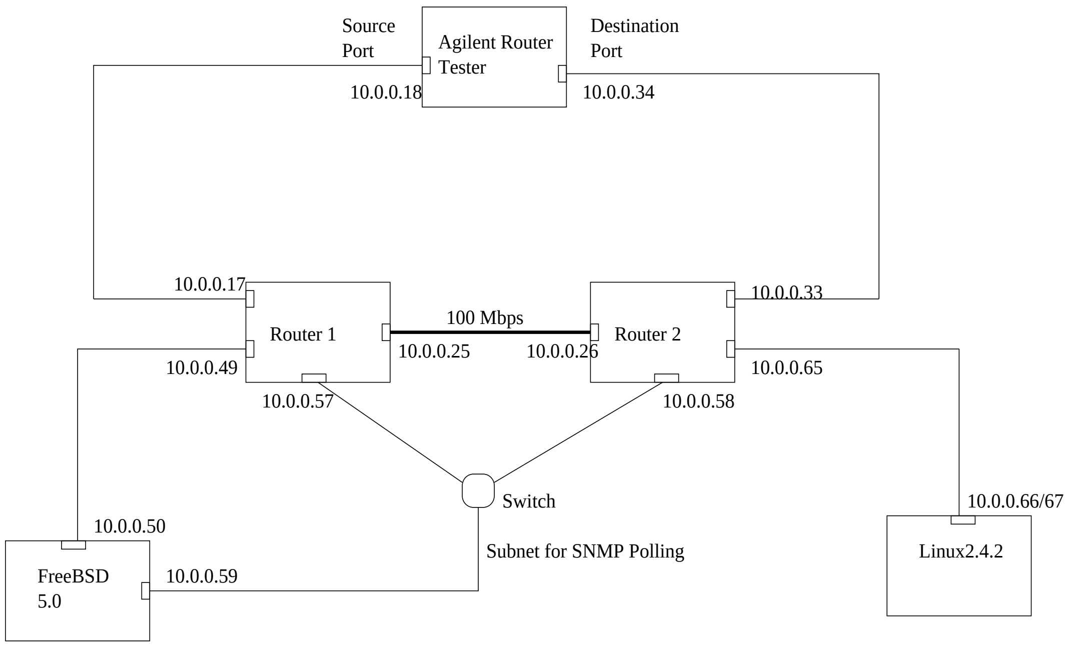


Figure 3.10: Testbed Setup

ms respectively. The R1/R2 values used in the experiments were 100/20, 80/40, 60/60 ms. The two PCs were kept time synchronized with an NTP server. In each experiment, 10 FTP flows were started from FreeBSD machine to the Linux machine for each destination address. The number of flows was chosen so that the PCs could easily handle the processing overhead of the experiments. Each flow transferred a 180 MB long file from FreeBSD machine to different directories on Linux machine. The results shown here correspond to the duration (always greater than 15 minutes) when all the 20 flows were active. *tcpdump* ran on each machine to record all the TCP packets sent and received. The FreeBSD machine used Reno as the TCP protocol and 1460 bytes as the packet size. The Linux machine used delayed ack mechanism and had a *recv window* of 12 packets. While the experiments ran, the FreeBSD machine queried the MIB-2 Interfaces table on two routers after every 3 seconds over a subnet unaffected by congestion. We experimented with both droptail and RED buffer management policies. For buffer size x, the RED min and max thresholds were 0.5x and x with the max drop probability being 0.1 and exponential weighing constant for calculating average buffer occupancy being 1/512. Agilent router tester was used to generate 7 different background traffic mixes to get a good range of the loss rates. Each traffic mix had a bursty component consisting of bursts of 1000 packets every second of 60, 576 and 1460 byte packets each and a CBR component. The CBR component (with load values of 50, 60, 70, 80, 85, 90 and 95 Mbps) distinguished different traffic mixes and consisted of 50 parts each of 576 and 1460 byte packets and 5 parts of 60 byte packets (to represent TCP ACKs).

| Simulation ID | Background CBR Load (in Mbps) | Round-trip Propagation Delay R1/R2 |
|---|---|---|
| 1-3 | 50 | 100/20 ms, 80/40 ms, 60/60 ms |
| 4-6 | 60 | |
| 7-9 | 70 | |
| 10-12 | 80 | |
| 13-15 | 85 | |
| 16-18 | 90 | |
| 19-21 | 95 | |

Table 3.4: IDs for Testbed Experiments

### 3.6.3 Performance With Accurate Loss Rate, RTT and Timeout Duration Information

In this section, we evaluate the accuracy of the TCP throughput prediction model we have just derived. In order to provide a benchmark reference point of its performance, we first evaluate its ability to accurately predict TCP throughput given *actual* average values for the different parameters it relies on, i.e., RTT, time-out duration, loss rate, etc. Clearly, this represents a best case scenario for the model, but is useful in assessing the impact that estimation errors have on its ability to accurately predict TCP throughput. Evaluations in more realistic scenarios, i.e., using estimated network parameters, are carried out in next section.

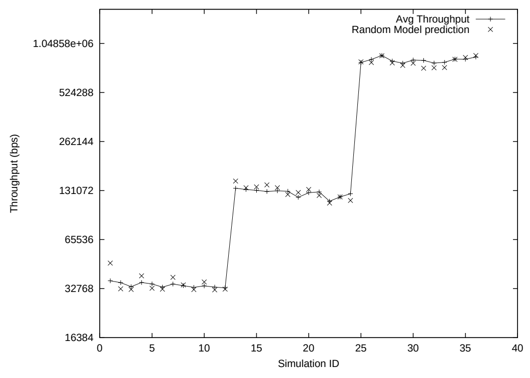
Figure 3.11 presents the average throughput values achieved in different simulations as well as the throughput predictions made by our model using actual average values for the input parameters. Figure 3.12 shows the same results as figure 3.11 after normalizing them against actual average throughput values. For reference purposes, the average loss rates observed in these simulations are shown in Figure 3.13. An overview of the throughput prediction results indicates that our model performs excellently in RED scenarios and reasonably well in droptail scenarios. Excellent performance in RED environment demonstrates that our model captures the TCP dynamics very well in the environment characterized by randomized losses. In the droptail simulations, the model's throughput predictions are remarkably well especially in the knowledge of the fact that packet loss process in droptail simulations is quite bursty in nature. Thus, even for bursty loss scenarios, if a good estimate for average loss rate can be obtained, our model provides a reasonably good estimate for the TCP throughput. The only significant deviations are observed for T3 simulations where the per-flow throughput and hence window size values are pretty high. This coupled with droptail nature of the buffers result in bursty packet loss sequences for TCP flows which our model based on the assumption of uniform loss rate does not capture very well. This is the price one has to pay for restricting estimation procedures to non-invasive ones. Also note that our model seems to underestimate the TCP throughputs in these cases. This suggests that the prevalent loss rates in these environments are quite less than the average loss rates. This also means that most of the packet losses are lumped together and the resulting increase in retransmission timeout probability does not effect throughputs as much as the reduced prevalent loss rates. This looks reasonable because the simulation TCP code does not enforce the lower limit of 1 second on the retransmission timeout values.

Next, we present the results obtained from our testbed experiments. Figure 3.14 shows the actual average throughput values and the predictions made. The same results are shown in Figure 3.15 after normalizing throughput predictions against actual average values. Figure 3.16 shows the average loss rates observed in these experiments. Again, the throughput predictions for RED experiments are excellent confirming the simulation results. The droptail experiment results are mostly reasonable with a few overestimates. A careful look tells us that the overestimates have been produced for flows with small (20 ms) round trip propagation delays. A small round trip time means that all the packets in the window travel close to each other which tends to produce bursty losses in droptail environments. With retransmission timeout having a minimum value of 1 second, the resulting higher timeout probability (when compared to non-bursty loss scenario) clamps down the actual throughput achieved and hence the overestimate in the predicted throughput.

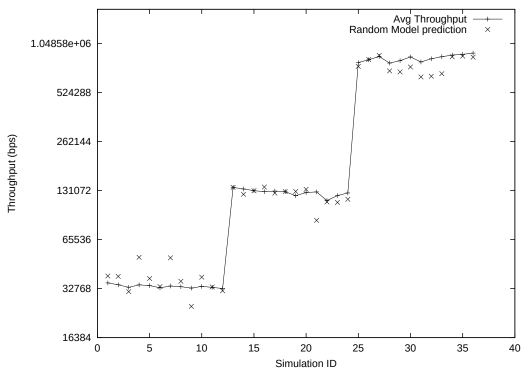
Overall, the simulation and testbed results indicate that our throughput prediction model based on simple average loss rate performs reasonably well in a wide range of scenarios when the accurate loss rate, RTT and timeout duration information is fed to the model as input. In the next section, we examine how well the model does when only the *non-invasive* estimates of input parameters are available.

### 3.6.4 Performance With Non-Invasively Determined Estimates for Input Parameters

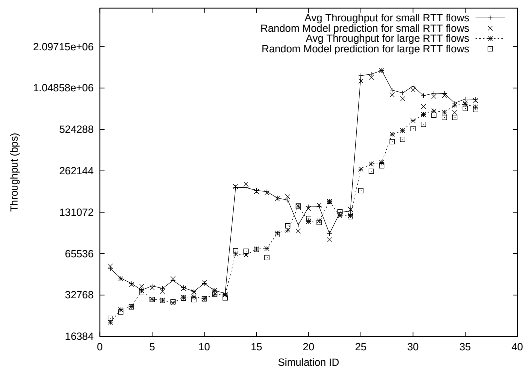
In this section, we evaluate the performance of our throughput prediction model when used with *non-invasively* obtained estimates of the loss rates, RTTs and timeout durations. Earlier in section 3.4, we have seen the performance of the different SNMP polling and probing based estimation techniques and concluded that router SNMP MIBs provide reasonably good loss rate estimates (for RED routers) while the ping probing is quite effective in RTT and timeout duration estimation. The simulation results presented in this section have used the aggregate loss rates observed at the routers as the estimate for the flow level loss rates while *ack-paced* probing was used to obtain RTT and timeout durations. The testbed results also used MIB loss rates as the flow-level loss rate estimates. For testbed



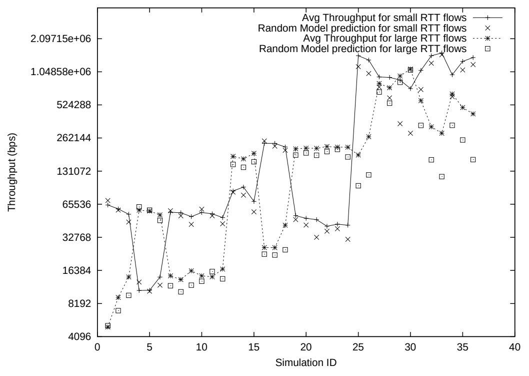
(a) Single RED Router, Homogeneous RTT Flows



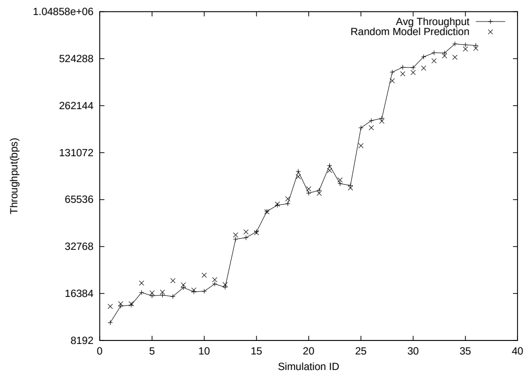
(b) Single Droptail Router, Homogeneous RTT Flows



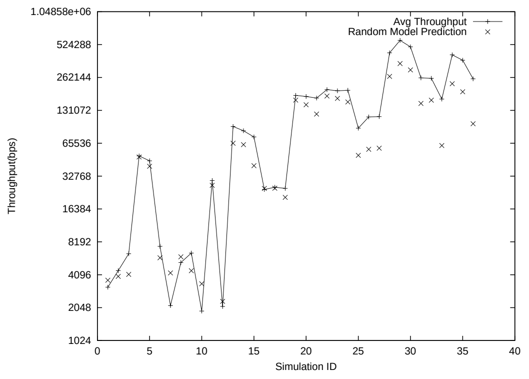
(c) Single RED Router, Heterogeneous RTT Flows



(d) Single Droptail Router, Heterogeneous RTT Flows

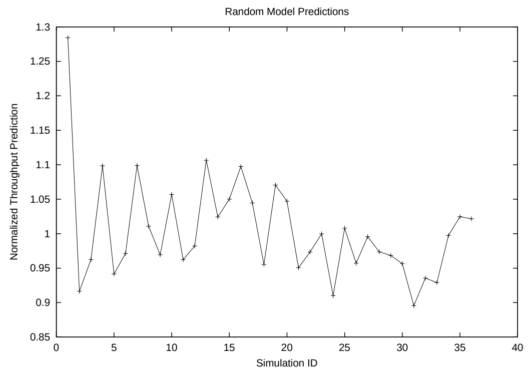


(e) Two RED Routers

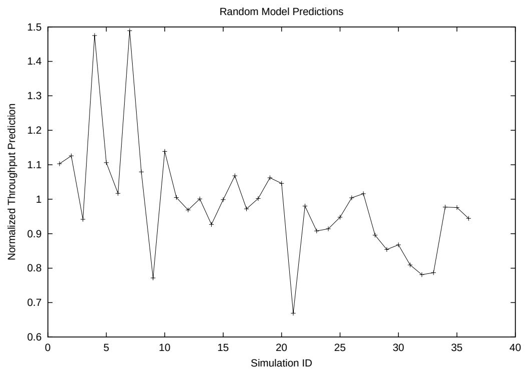


(f) Two Droptail Routers

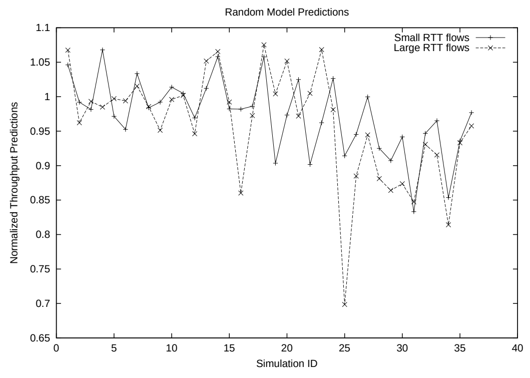
Figure 3.11: Simulation results for throughput prediction with actual network parameters.



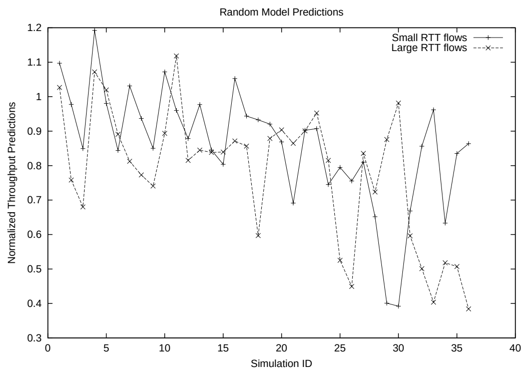
(a) Single RED Router, Homogeneous RTT Flows



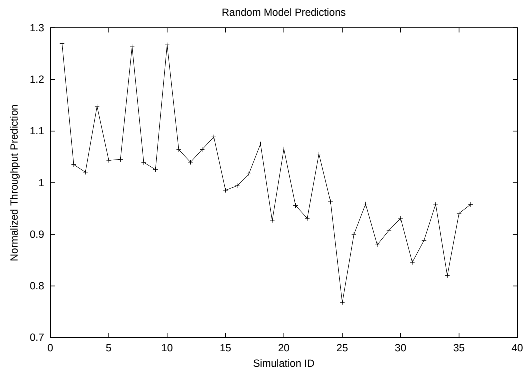
(b) Single Droptail Router, Homogeneous RTT Flows



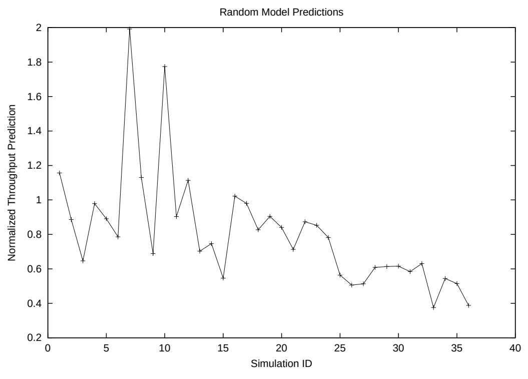
(c) Single RED Router, Heterogeneous RTT Flows



(d) Single Droptail Router, Heterogeneous RTT Flows

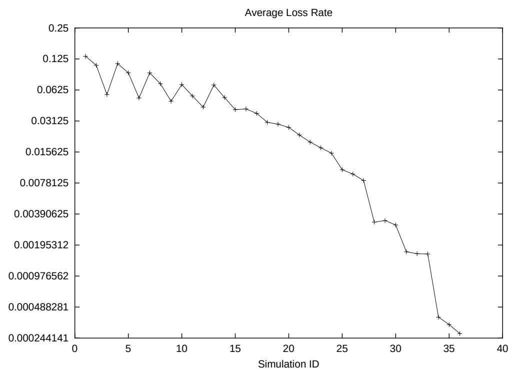


(e) Two RED Routers

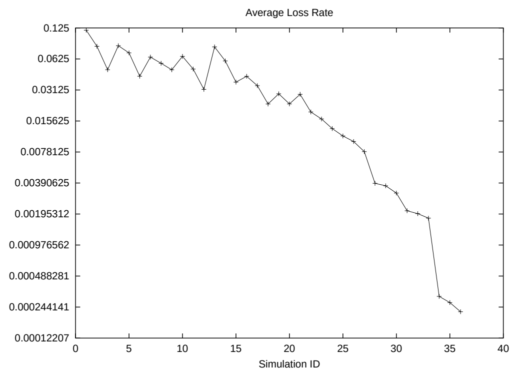


(f) Two Droptail Routers

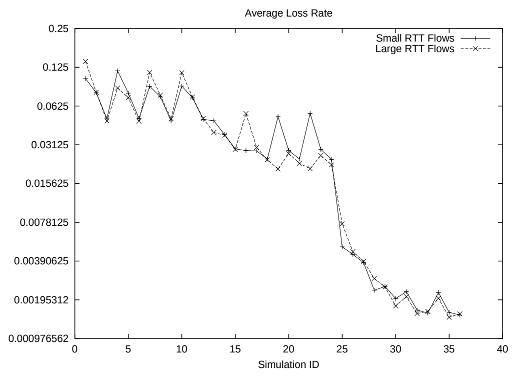
Figure 3.12: Simulation results for throughput prediction with actual network parameters.(Normalized wrt actual avg throughput values)



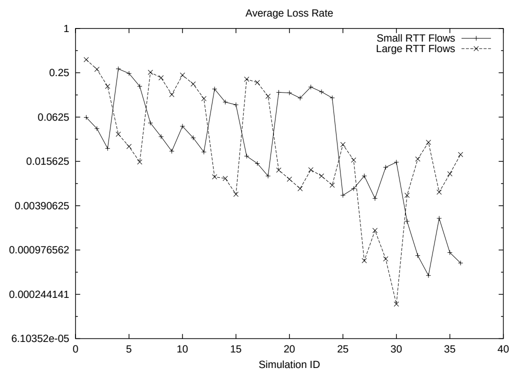
(a) Single RED Router, Homogeneous RTT Flows



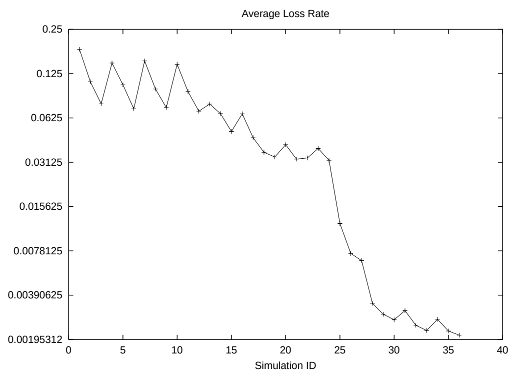
(b) Single Droptail Router, Homogeneous RTT Flows



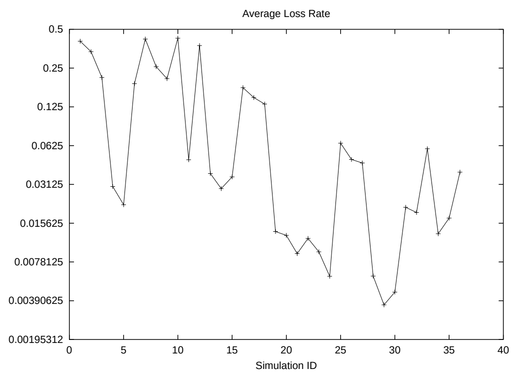
(c) Single RED Router, Heterogeneous RTT Flows



(d) Single Droptail Router, Heterogeneous RTT Flows

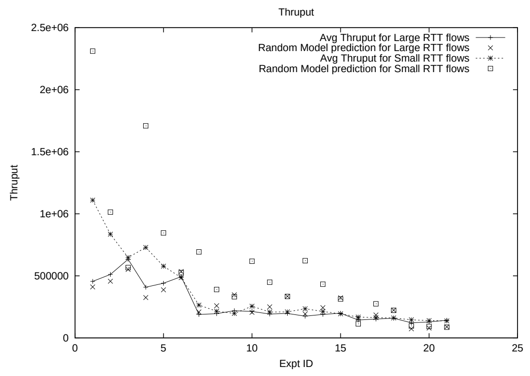


(e) Two RED Routers

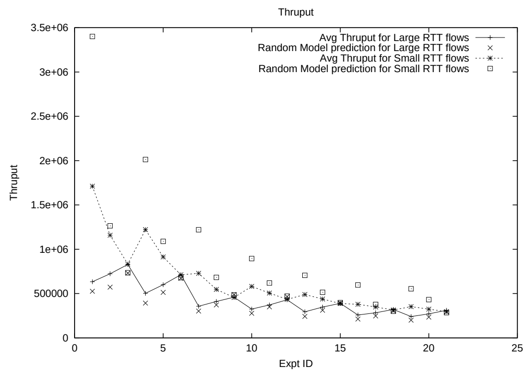


(f) Two Droptail Routers

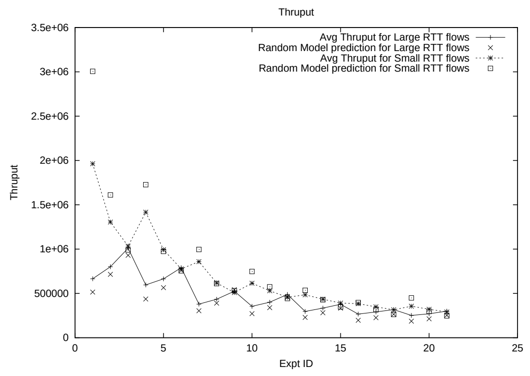
Figure 3.13: The actual average loss rates observed in simulations



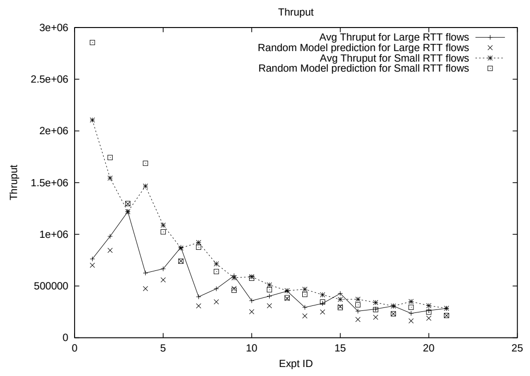
(a) Droptail (40 pkt buffer) Experiments



(b) RED (100 pkt buffer) Experiments

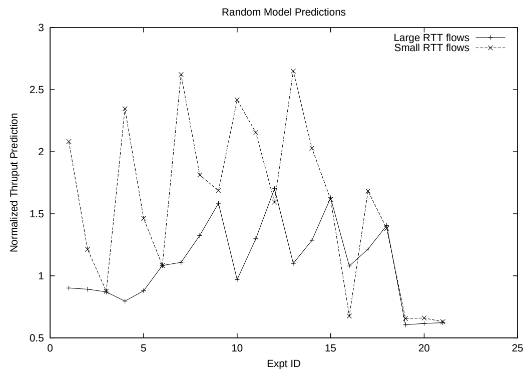


(c) RED (500 pkt buffer) Experiments

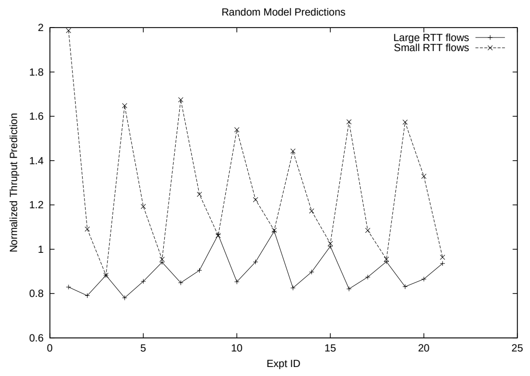


(d) RED (1000 pkt buffer) Experiments

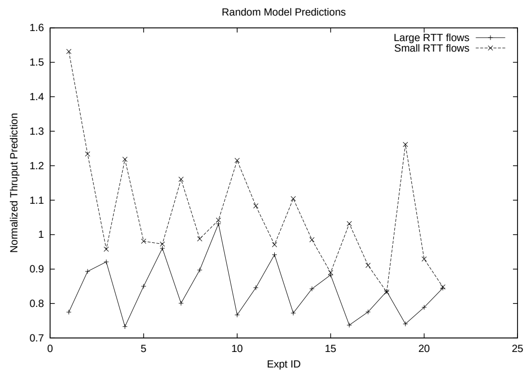
Figure 3.14: Testbed results for throughput prediction with actual loss rates.



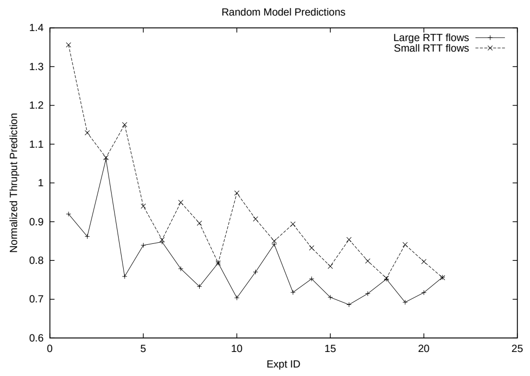
(a) Droptail (40 pkt buffer) Experiments



(b) RED (100 pkt buffer) Experiments

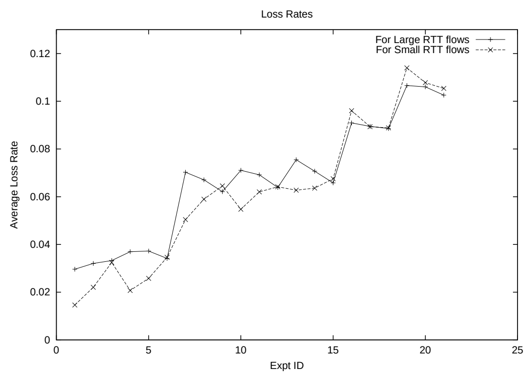


(c) RED (500 pkt buffer) Experiments

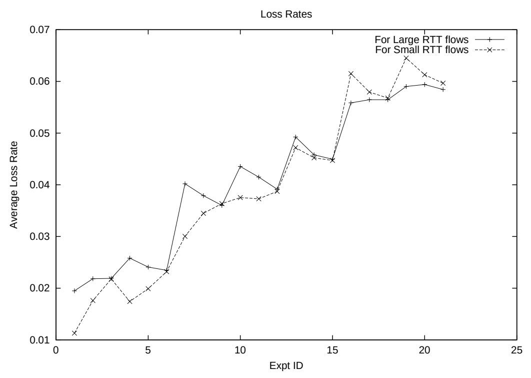


(d) RED (1000 pkt buffer) Experiments

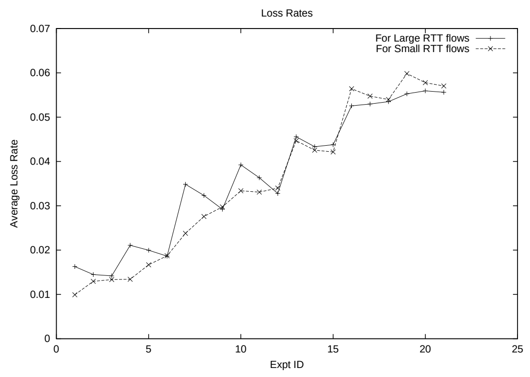
Figure 3.15: Testbed results for throughput prediction with actual loss rates.(Normalized wrt actual average throughput values.)



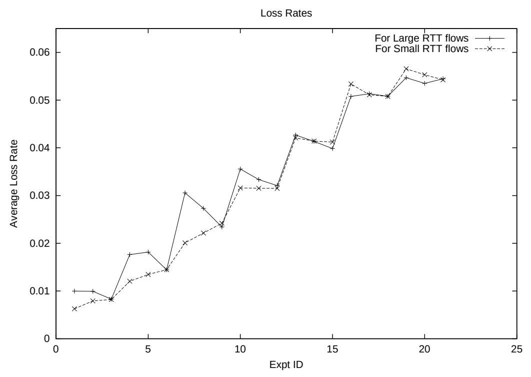
(a) Droptail (40 pkt buffer) Experiments



(b) RED (100 pkt buffer) Experiments



(c) RED (500 pkt buffer) Experiments



(d) RED (1000 pkt buffer) Experiments

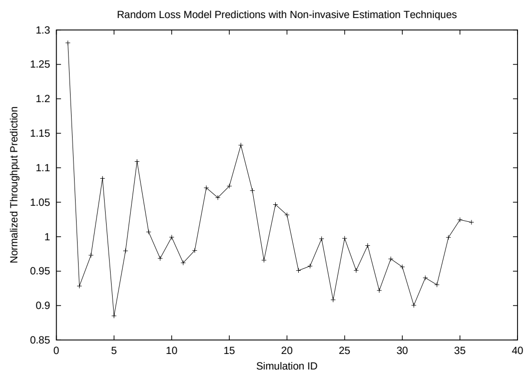
Figure 3.16: Actual average loss rates observed in testbed experiments.

experiments, there was no need to estimate the 'first' retransmission timeout duration as it was observed to be 1 second almost always. An estimate of the flow RTTs was obtained by adding the average queuing delay at the congested router interface to the round trip propagation delays. The queuing delay was calculated using the output packet queue information maintained in the *Interfaces* table of SNMP MIB-II [95] and an average packet size computed from known proportion of packet sizes passing through the interface.

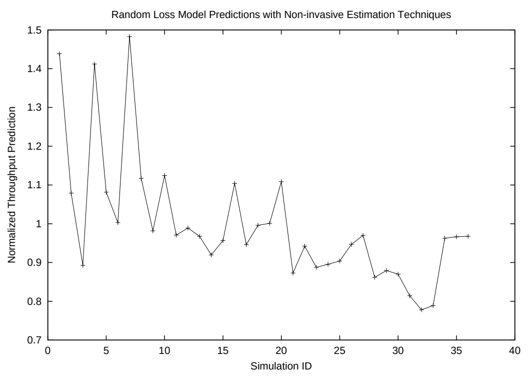
Figures 3.17 and 3.18 show the performance of our throughput prediction model using *non-invasively* estimated input parameters. For RED simulations and experiments, the excellent predictions obtained with actual network parameters are maintained even with non-invasively determined estimates of network parameters. As already discussed, for two router simulations, we used the "independent losses" assumption to combine individual router loss rates into an end-to-end loss rate. We have already seen that the resulting estimate for end-to-end loss rate is pretty good for RED buffers. Here, we notice that throughput prediction made on the basis of such an estimate is also quite good. Looking at droptail simulations and experiments we observe that the throughput predictions are not satisfactory many times. The simplifying assumption of uniform loss rate used in the throughput prediction model coupled with inability to accurately estimate average loss rates renders the predicted throughputs to be off the mark many times.

### 3.6.5 How Soon Is Steady State Reached?

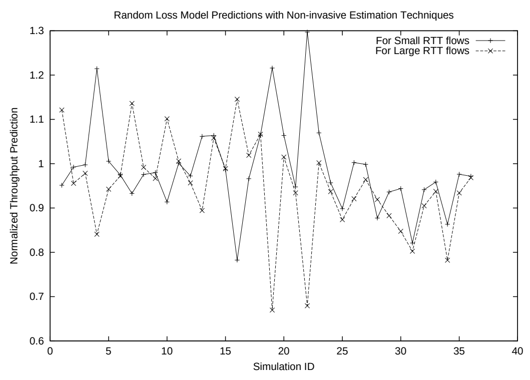
In this section, we address one last aspect that is of importance in practice. Specifically, we have been concerned so far with developing models for predicting *"steady state"* throughput of TCP flows. However, most TCP flows are relatively short-lived, and this



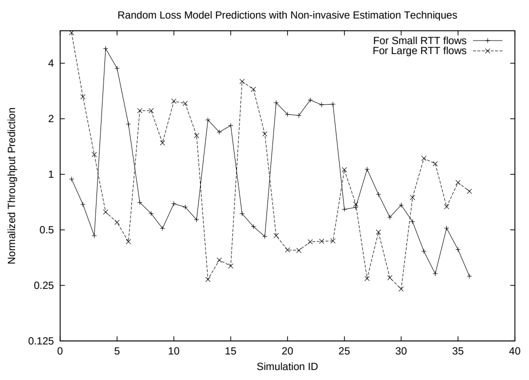
(a) Single RED Router, Homogeneous RTT Flows



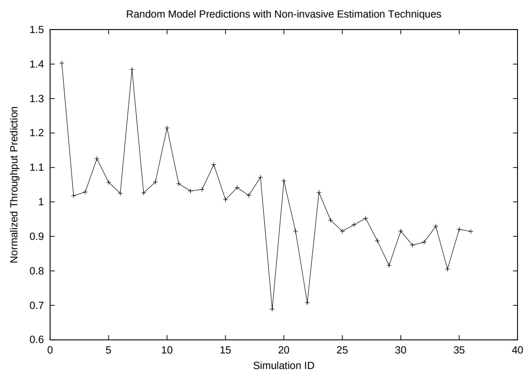
(b) Single Droptail Router, Homogeneous RTT Flows



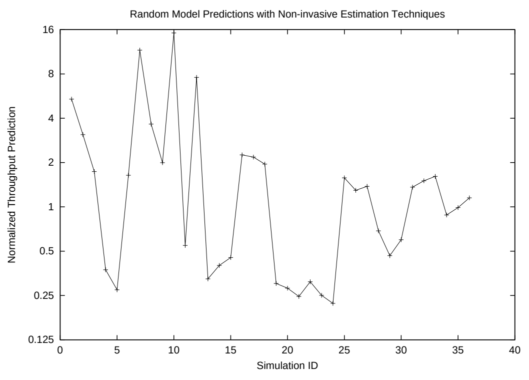
(c) Single RED Router, Heterogeneous RTT Flows



(d) Single Droptail Router, Heterogeneous RTT Flows

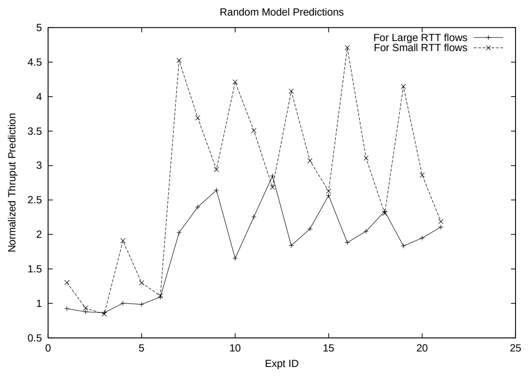


(e) Two RED Routers

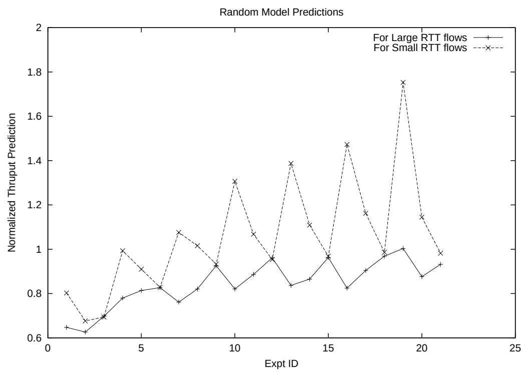


(f) Two Droptail Routers

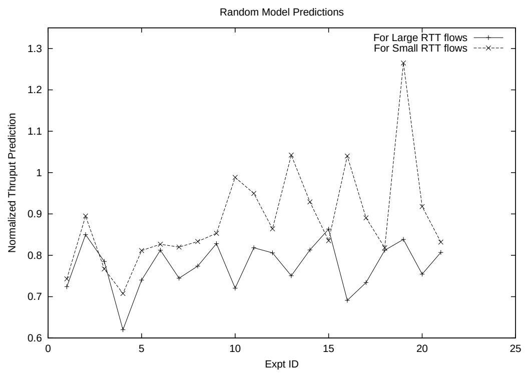
Figure 3.17: Simulation results for throughput prediction with non-invasive estimation techniques.(Normalized wrt actual average throughput values)



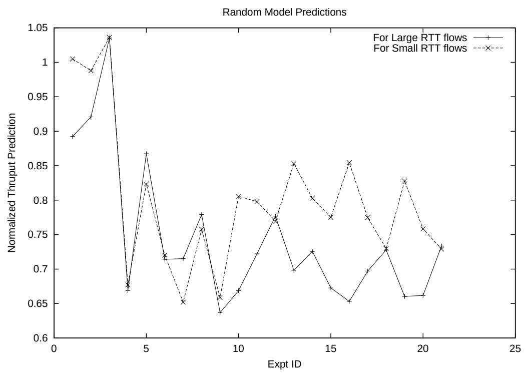
(a) Droptail (40 pkt buffer) Experiments



(b) RED (100 pkt buffer) Experiments



(c) RED (500 pkt buffer) Experiments



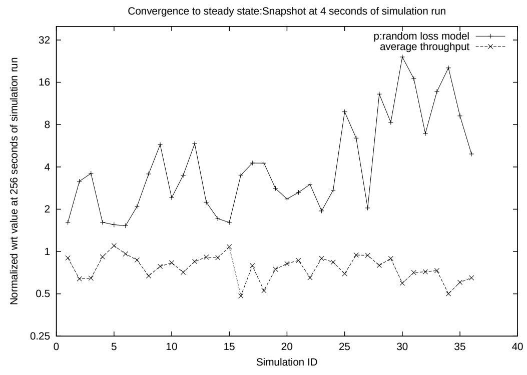
(d) RED (1000 pkt buffer) Experiments

Figure 3.18: Testbed results for throughput prediction with non-invasive estimation techniques.(Normalized wrt actual average throughput values)

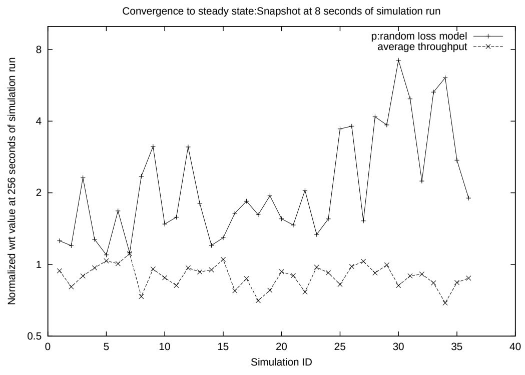
therefore begs the question of how soon the steady state throughput is reached. Investigating this issue is the topic of this section, and results are reported in Figures 3.19 and 3.20. Figures 3.19 plots the evolution over time of throughput and loss rate mean values, while Figure 3.20 gives the same information but using standard deviation. In both cases, the different values have been normalized with respect to final (steady state) values at the end of the simulation. Note that TCP throughput tends to achieve its steady state value very quickly, in many cases as early as after 4 seconds of simulation run, while the loss rate takes much longer (around 32 seconds) to stabilize. The main conclusion from the figures, is that, at least in the scenarios being simulated, TCP throughputs and loss rates appear to converge to their steady state value within 50 seconds of simulation run. This corresponds to about a couple of hundred RTT periods. These numbers give us some indication of the time range for which steady state throughput prediction models are applicable.

## 3.7 Summary

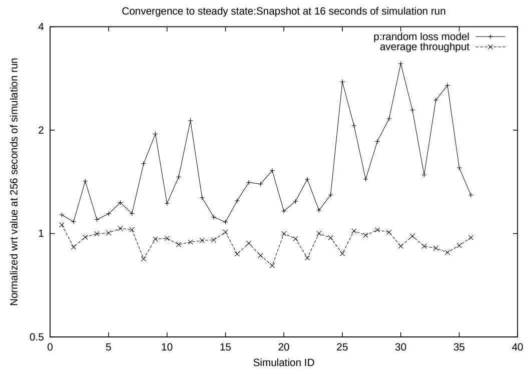
This chapter has been concerned with developing and evaluating models for predicting the steady state of representative TCP flows between fixed end-points, based on non-invasively obtained network data. The motivation for this work is to enable the transformation of data readily-available to service providers into user and application specific metrics. As existing TCP models do not readily fit the task at hand, we chose the popular Amherst model as a starting point and developed a modified cyclical model to suit our purposes. We also investigated the appropriateness and limitations of polling and probing as techniques for generating suitable inputs for our model. Our TCP throughput prediction model has



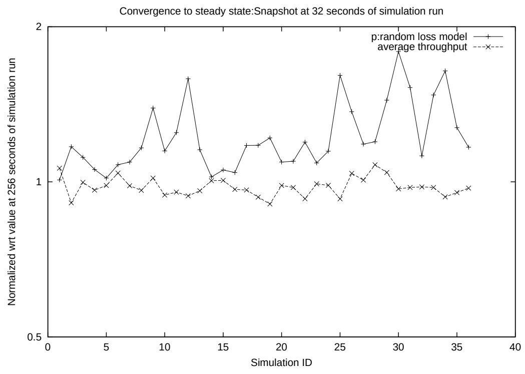
(a) Snapshot at 4 seconds



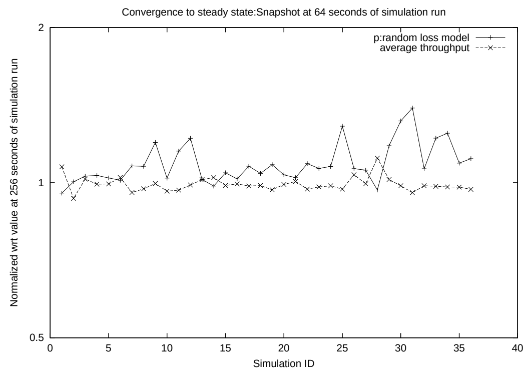
(b) Snapshot at 8 seconds



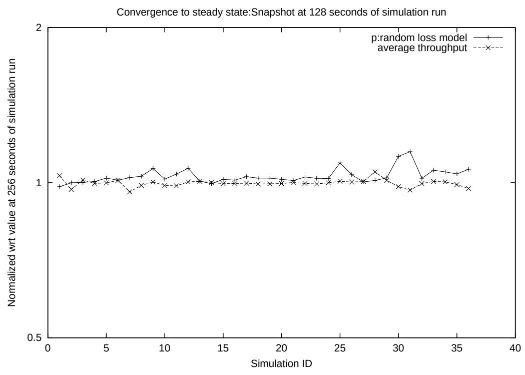
(c) Snapshot at 16 seconds



(d) Snapshot at 32 seconds

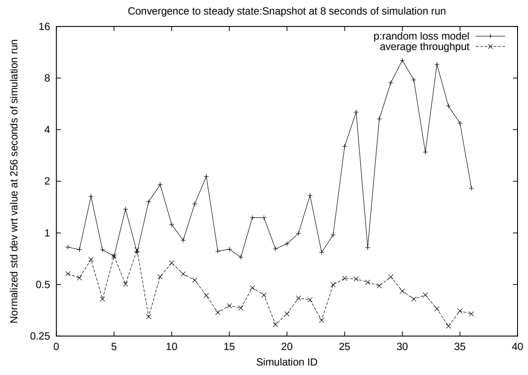


(e) Snapshot at 64 seconds

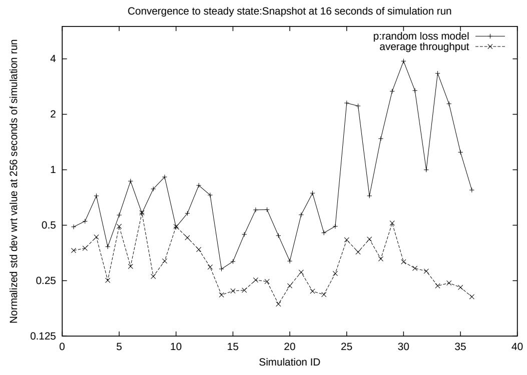


(f) Snapshot at 128 seconds

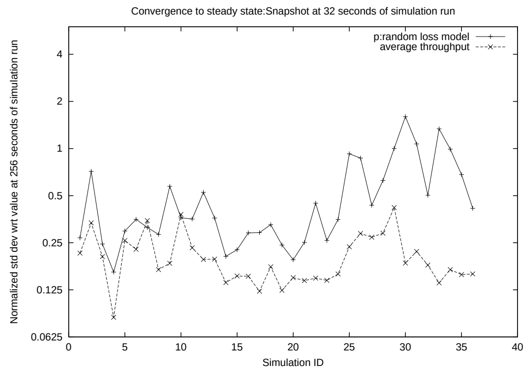
Figure 3.19: Convergence of average values of various parameters with time for simulations with single RED router and heterogeneous RTT flows.



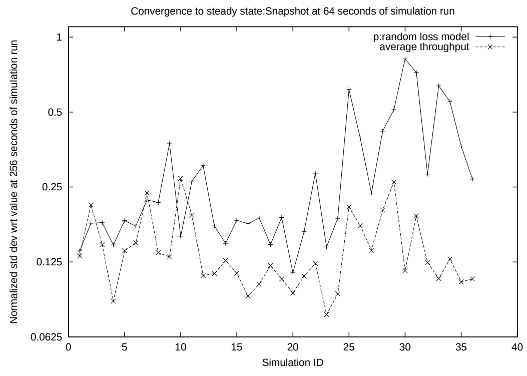
(a) Snapshot at 8 seconds



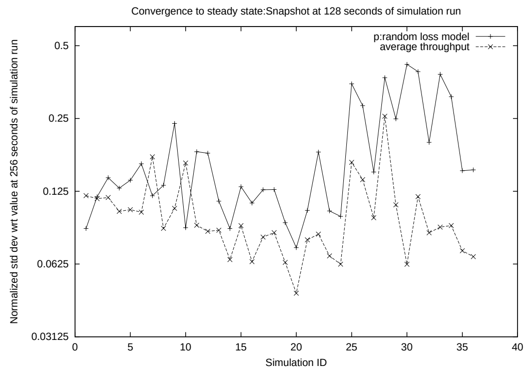
(b) Snapshot at 16 seconds



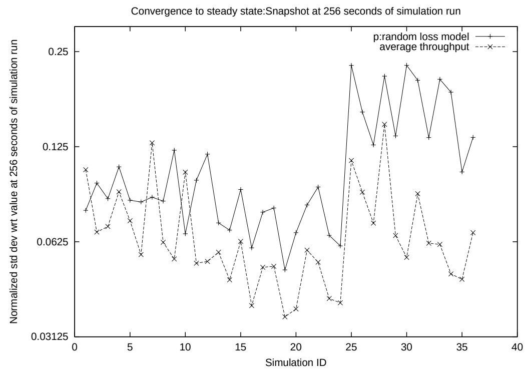
(c) Snapshot at 32 seconds



(d) Snapshot at 64 seconds



(e) Snapshot at 128 seconds



(f) Snapshot at 256 seconds

Figure 3.20: Snapshots of standard deviation of various parameters at different times for simulations with single RED router and heterogeneous RTT flows.

been validated using simulations and as well as testbed experiments under a variety of scenarios. In the context of TCP throughput prediction based on non-invasive estimates of QoS, there are several directions in which more research work can be done:

1. Our work focuses on long-lived TCP Reno flows. Metrics based on short-lived flows, as well as other TCP flavors need to be investigated [123]. For predicting the throughput of short-lived TCP flows, the important challenge is to design reliable non-invasive methods to estimate required input parameters - a potentially difficult task given the short life time of flows.

2. It will be interesting to examine if throughput prediction can be improved by modeling the effect of varying loss rates during the life time of a flow.

3. Our work assumes that one can determine the congestion points affecting flows between two end-user sites. This assumption is valid when network access points are the principal bottlenecks. However, more sophisticated techniques for determining congestion points need to be coupled with our techniques, in order to select network samples relevant to a user.

4. Non-invasive estimation of loss rates for droptail buffer management policy remains a big challenge. In general, significant more research is required for the development of non-invasive techniques that can reliably estimate the network characteristics as observed by the end-user applications. The important network characteristics requiring immediate attention include the average loss rates and a measure of correlation in packet losses.

5. The next logical task for the future is to work on other building blocks identified in the beginning of this chapter (such as DNS lookup times and web server latencies)

to obtain a complete framework for for predicting the performance of common data applications (Web downloads, FTP transfer times).

Creating a framework for predicting the application level performance using easily available network QoS observables is an ambitious goal and our work so far can be viewed as a small first step in this direction. On the basis of the experience gained so far, we have recognized three important dimensions of the problem : characterizing the network, the application and relating the two. We conclude this chapter with a brief description of these dimensions.

Characterizing the network (i.e. evaluating network QoS) is a difficult task because of several reasons. We are interested in determining how the network behaves from the perspective of the application traffic. This requirement some times makes the problem easy and at other times presents un-surmountable difficulties. For example, in our TCP throughput prediction work, we realized that for a single flow, the loss process can be assumed to be “bernoulli” even though it is “correlated” (or bursty) in nature for aggregate traffic. Bernoulli loss process is much simpler to model than a correlated loss process. Further, determining the characteristics of the Bernoulli loss process is much simpler than doing the same for a correlated loss process. So, in this respect, the requirement to evaluate network QoS from the perspective of application traffic played a beneficial role in the performance evaluation process. However, this was not always the case. For example, we realized, to our dismay, that aggregate loss rate given by router MIBs is many times not a good estimate for loss rate observed by a single TCP flow. The task of measuring network QoS characteristics is made more difficult by the lack of infrastructure for measurement. Often, this requires access to individual network devices which is not possible. Even the network owner companies do not understand the nature of their networks because of either lack of suitable measurement infrastructure or the impossibility of analyzing the available data. The compromise will be a network model sophisticated enough to capture the important characteristics and yet simple enough to be easily measurable. The task of characterizing an application can become very complex. While it may be straight forward to identify the building blocks that determine the application level performance, the complexity is introduced by the often unpredictable interaction between different building blocks. Understanding the interaction between different factors determining the application performance for popular applications is an open research problem and needs considerable progress before satisfactorily levels of understanding are achieved and useful models emerge.

Even if detailed informational models are available for the network and the application, the exact mapping between the two is often difficult to achieve and apply because of mathematical complexities involved. Typically, simplifying assumptions are required to achieve a useful mapping between the network and application models. Often the choice is between mathematical simplicity and the correctness of the mapping. Our experience with mapping between long term TCP throughput and the average loss rate indicates that if the predictions are generally within 20% of the actual values, the mapping should be considered accurate.

# CHAPTER 4

# THE COST OF SURVIVABILITY IN IP NETWORKS

## 4.1 Introduction

In the previous chapters, we focussed on the performance of Internet from a user's perspective. Now, we turn our attention to the performance of the networks from the perspective of Internet Service Providers. While a user determines the performance of the Internet on the basis of his/her perception of service quality offered by the Internet, the providers determine the performance of the network in terms of the cost of operating the network to support the agreed upon service quality requirements of the users. As discussed in Chapter 1, the survivability against failures has become an integral part of the service quality requirements of the users. In the rest of the thesis, we focus on the evaluation and optimization of the performance of the networks from the provider's perspective taking in account the survivability requirements. We initiate the discussion by focusing on the cost of network operation for survivability in the current chapter. The remaining chapters in the thesis focus on the speed of recovery from the failures.

Today, the Internet has become a key for communications and commerce in the world. Private network infrastructures, also based on IP, whether they are distinct networks or virtual private networks on a larger shared IP network, are vital for business enterprises.

Customers, especially enterprises expect a highly available network service, and their service level agreements with service providers specify the availability and reliability needs of the infrastructure. Service providers have to design and provision their networks so that they can survive failures. Maintaining connectivity in face of failures in the network, and provisioning links to have sufficient capacity to carry the additional network traffic coming their way in the event of failures, is expected by the users of the network. It is vital that the networks can withstand even major facility failures (such as the loss of a PoP - the Point of Presence[8]) without leading to lack of connectivity or degradation of quality of service. In our analysis of several network topologies belonging to both commercial and educational ISPs, we observed that most of these networks can not maintain connectivity between all source-destination node pairs in face of certain single node failures (we represent a PoP as a single node in the network topology). The main reason for the observed lack of survivability is the sparse connectivity within the network. In addition to poor survivability characteristics, sparse connectivity in the network offers little opportunity for traffic engineering since there are often no additional available data paths between nodes in the network.

The need for the network to maintain connectivity and the quality of service in the face of failures raises a lot of issues that need to be addressed. Since protecting the quality of service (minimally assuring availability of capacity) in face of network failures requires redundant resources, what is the cost of survivability? Since the traffic matrix observed on a network can vary significantly over both short and long term time scales, how does the cost of survivability vary with the traffic matrix? The desire for flexible traffic engineering has led to design of MPLS based networks where the route calculation may be based on

[8]the group of *access* and *core* routers located in the same facility such as the same room or the same building.

considerations such as existing traffic load and resource availability, in addition to static link weights. MPLS protocols provide failure recovery either by local re-tunneling of the affected traffic around the failed nodes (local recovery) or by switching the affected traffic to a different end-to-end backup path that is node disjoint with the original path (end-to-end recovery). What is the relative cost of survivability for modern MPLS based schemes when compared to traditional OSPF routing? Since the cost of survivability is determined by the routes taken by the traffic through the network, how much can the cost of survivability reduced by adjusting traffic routes in an intelligent manner?

Traditional OSPF based routing as well as some of the popular MPLS routing algorithms depend directly or indirectly on "static" link weights to make routing decisions. Adjusting the link weights provides a natural handle to reduce the cost of survivability. Often, owing to sparse connectivity in the network, there are only a few choices for possible routes, and in such scenarios weight adjustments may not offer significant savings. We examine how the careful addition of few new links in the network can significantly reduce the cost of network operation for survivability. New links in the network can offer new shorter paths for traffic that would otherwise travel over multiple hops. We examine the effectiveness of link additions and weight adjustments together in terms of reducing the cost of survivability. Since, the traffic matrix observed on a network can undergo significant changes in both short term and long term time scales, any weight adjustments and link additions have to necessarily be insensitive to traffic matrix variations. A service provider would likely consider making topology modifications only if the benefits persist even after drastic changes in the traffic matrix. We examine the sensitivity of the suggested topology modifications to the traffic matrix.

Traffic engineering has traditionally been associated with load balancing i.e., adjusting the routes such that traffic loads move from highly utilized links to links with lower utilization. Recently, the link weight adjustments have been identified as a practical and effective means of achieving better balance in link utilizations [49, 48, 50]. Load balancing through adjustments of the link weights can have serious implications for the network survivability, because a previously failure-resistant network might become susceptible to failures after route changes performed during load balancing operations. Further, the cost of survivability may increase. We study the relationship between the cost of survivability and the link load balance. We examine the approaches that attempt to achieve both goals - reduce the cost of survivability and improve the balance in link utilizations.

In this chapter, we provide empirical answers to the issues raised above , based on experiments with several real IP backbone network topologies assuming a variety of realistic traffic matrices. The rest of the chapter is organized as follows. The next section provides an overview of the previous research on network survivability. Section 4.3 begins with an examination of several IP backbone topologies belonging to commercial and educational ISPs, illustrating their poor survivability characteristics, and the need to improve connectivity within the network. This is followed by the application of a heuristic based approach to add links to the topologies so that they can survive any single node failure. We examine the capacity requirements with and without single node failure protection on these topologies for several realistic traffic loads to get an understanding of the cost of survivability. This examination is performed for both OSPF and MPLS based failure recovery mechanisms. Section 4.4 deals with reducing the cost of network operation for survivability using weight adjustments and link additions. We show that the cost can be significantly reduced by weight adjustments and careful addition of a very limited, key links to the topology.

The section next examines how sensitive are topology modifications (weight adjustments and link additions) to the variations in traffic matrix. Section 4.5 examines the relationship between load balancing and the cost of network operation for survivability. We examine if load balancing can be achieved without increasing the cost and vice versa. Finally, we conclude the chapter with a summary of the new understanding gained by our work and outline future work.

## 4.2 Related Work

The existing literature on network survivability has explored the issue from several dimensions. Survivability has been defined as “the capability of a network to maintain service continuity in the presence of faults” [12]. While the intuitive meaning of survivability can be easily understood, quantifying the survivability characteristics of a network is not straight forward. [136, 96, 31, 106, 90] offer different approaches to measure survivability, robustness towards failures and the impact of a failure. Fast recovery from network failures and reducing the cost of survivability constitute the two most important aspects of network survivability. Fast failure recovery has received significant attention in the context of ATM [8], IP [3, 26, 56] and optical [119, 88, 113, 57] networks and continues to be an area of active research. Regarding the cost of survivability, many studies, over past several years, have confirmed that end-to-end recovery results in lower extra capacity requirements for protection against failures than local recovery [8, 76, 103, 67, 144, 118, 19]. Several of these works [103, 67, 144, 118] performed the comparison on the basis of optimized capacity requirements for local and end-to-end recovery. [103, 144] concluded that the relative gains obtained with end-to-end recovery depend significantly on the network topology: for sparsely connected topologies, end-to-end recovery provides significant capacity savings over local recovery; for well connected topologies, there is not much difference between the capacity requirements of local and end-to-end recovery. Most of these works based their analysis on the case of protection against single link failures. Significant attention has been given to the development of algorithms for routing traffic connections in a manner that optimizes certain criterion (e.g. the spare capacity requirements, the network delay etc.) while satisfying certain constraints (e.g. survivability requirements, hop limits etc.) [62, 61, 30, 25]. Other works have focused on developing or understanding the performance of simpler/heuristic based algorithms, many times with limited objectives, that give close to optimal performance[35, 58]. Significant attention has also been given to the development of *dynamic* QoS routing algorithm that do not assume prior information about connections in the network and the traffic matrix [79, 80, 112, 81, 74, 87].

Survivability in optical networks has also received significant attention in recent years [118, 119, 19, 37, 121, 63, 30, 25, 61]. Modern communication networks are multi-layer networks involving interaction of several technologies such as IP, ATM and WDM. From the survivability point of view, this has given rise to a number of questions regarding achieving the coordination among the actions of different network layers such as: how to identify the original cause of failure, how to determine which layer should be responsible for which failure, how to assign capacity for failure protection in different layers [40, 53, 146, 145, 124, 39]. The multi-layer nature of the network also means that the higher layer topology can be seriously affected by the failures at the lower layer. Thus, it is not sufficient that the higher layer connections have link/node disjoint primary and backup paths. This is so because a lower layer failure can cause components of both primary and backup paths to fail simultaneously. Such a situation can be avoided if the higher layer route calculation protocols are aware of lower layer failures that can disrupt the higher layer links. Two or more higher layer links that can fail simultaneously because of a lower layer failure are said to belong to the same *shared risk link group* (SRLG). A higher layer connection can be considered protected only if its primary and backup paths are not only link/node disjoint but also SRLG disjoint. If SRLG information is not available to the higher layer, it is important that the higher layer topology is constructed in a manner such that a lower layer failure does not cause partitions in the higher layer topology [29, 99]. Some work has also been done on developing dynamic routing algorithms that take into account the combined topology and resource information at IP and optical layers [82].

Load balancing via link weight adjustments has recently received significant attention [49, 48, 50]. [49] showed that it is possible to achieve close to optimal load balance with shortest path based algorithms such as OSPF and IS-IS. It was shown that a local search can often provide link weight settings that allows OSPF/IS-IS protocols to perform almost as effectively as optimal general routing. [50] suggested that optimization over key traffic matrices allows the link weight settings to provide good performance over all traffic matrices dominated by convex combination of these key traffic matrices. [141] suggested that the shortest path routing can be tailored to achieve any desired loop-free traffic routes by an appropriate setting of link weights. [104] proposed reconfiguration of logical IP level topology to achieve load balancing in face of dynamic traffic matrix for IP over WDM networks. Other innovative approaches to load balancing have been suggested by [21, 16].

## 4.3 Connectivity and Cost of Survivability

Survivability against failures requires redundancy in the network in terms of connectivity and capacity. It is necessary that the network be well-connected so that the failure of a part of the network does not lead to a partitioning of the network. Similarly, the links in the network should have sufficient capacity so that they can accommodate the additional traffic load coming their way after a failure in the network. In this section, we attempt to understand the connectivity and capacity requirements associated with protection against single node failures.

The *link* and *node* failures constitute the two main types of network failures that are typically observed in a service provider network [84]. The link failures can be caused by either the *interface* failures in a router or cable/fiber cuts. The node failures can be associated with either the failure of a single router or a complete PoP (comprising multiple routers) depending on the level of granularity associated with the topology. Router failures may occur when they are brought down for maintenance. In a large network, this could be quite frequent. Routers can also fail due to hardware and software problems, or due to a power failure. A complete PoP may fail due to power failures, weather and terror-strikes and other disasters. Most of the previous work regarding survivability has focussed on providing protection against single link failures. However, recently, protection against single node failures (including facility failures that may impact a complete PoP) has become particularly relevant, given the potentially catastrophic impact of such failures on the network services. Also relevant is the increased concerns that a physical facility where a PoP is located could be brought down due to a disaster. Moreover, protection against single node failures also provides protection against single link failures.

### 4.3.1 Connectivity to Improve Survivability

The survivability analysis presented in this chapter is based on the examination of 28 real network topologies used as the IP backbones by commercial and educational ISPs [91]. Most of these topologies are at the PoP level. That is, a node in our representation of the network topology is often a PoP. Figure 4.1 illustrates the characteristics of these topologies in terms of the number of nodes and links in the topology. Figure 4.2 shows the degree of connectivity in the topology i.e., the average number of directly connected neighbors for a node, calculated as the ratio of the number of unidirectional links to the number of nodes. We measured, for each topology, the percentage of *fatal* single node failures, i.e., the single node failures that will lead to a network partition (where a node is not reachable from at least one other node in the network.) The results are shown in Figure 4.3. Notice the high percentage of fatal single node failures even for topologies with a large number of nodes. The results are especially remarkable for MPLS end-to-end recovery. For MPLS end-to-end recovery, a topology can survive all single node failures if it is possible for each *connection*[9] to be assigned a primary path and a node disjoint backup path. Hence for MPLS end-to-end recovery, with the existing topologies, we see that either all single node failures are fatal or none is. Figure 4.3 shows that for MPLS end-to-end recovery, only 2 of the 28 topologies could provide node-disjoint primary and backup paths for all the connections even during failure-free operation[10].

The poor survivability characteristics of a topology can be related to its poor connectivity. As illustrated in Figure 4.4, a remarkably high percentage of nodes have only one directly connected neighbor. If this neighbor fails, the network will be partitioned. Clearly, there is a need to improve the connectivity in these topologies if survivability against single node failures is desired. A simple strategy will be to introduce new links in the topology so that each node is directly connected to at least two other nodes. Accordingly, we connected

[9]We use the term "connection" to refer to the traffic flowing between a source node and a destination node in the network.

[10]We used simple shortest path routing to calculate the primary path. The backup path was calculated after removing all the nodes on the primary path, except the source and destination, and re-running the shortest path algorithm.

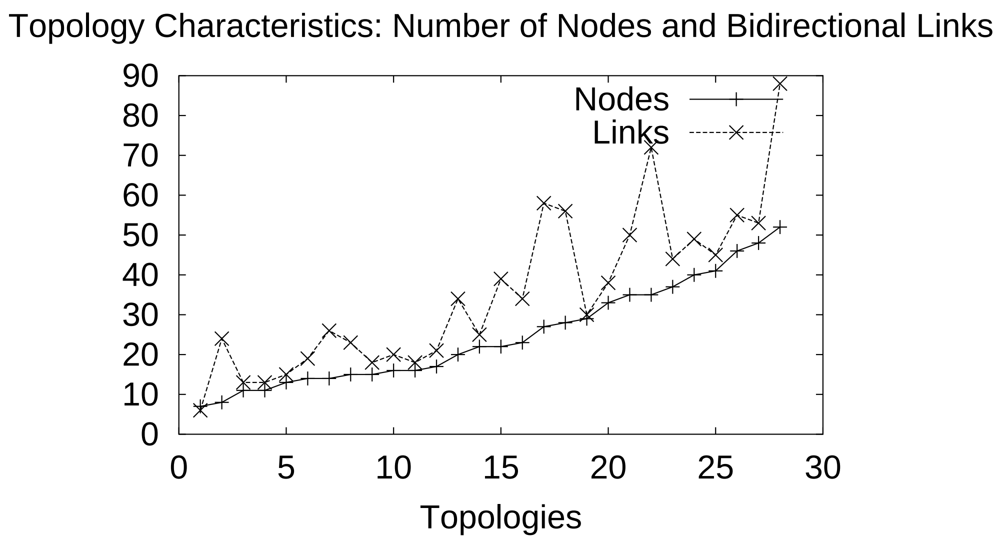


Figure 4.1: Topology characteristics: number of nodes and links

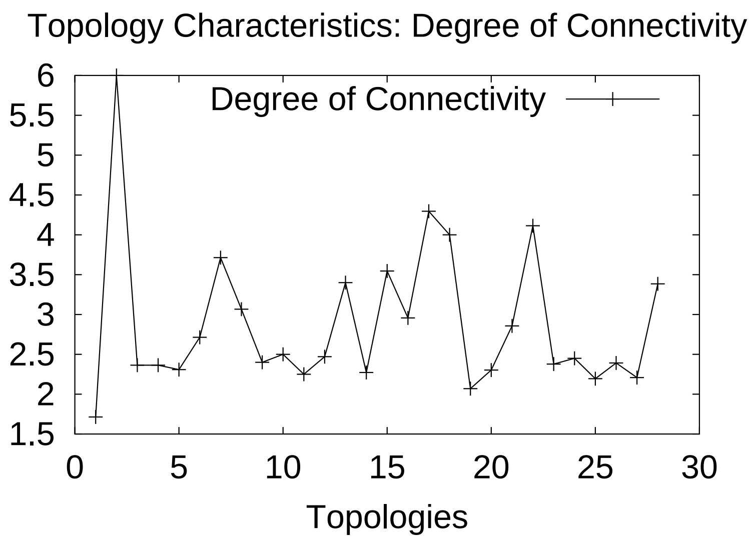


Figure 4.2: Topology characteristics: degree of connectivity

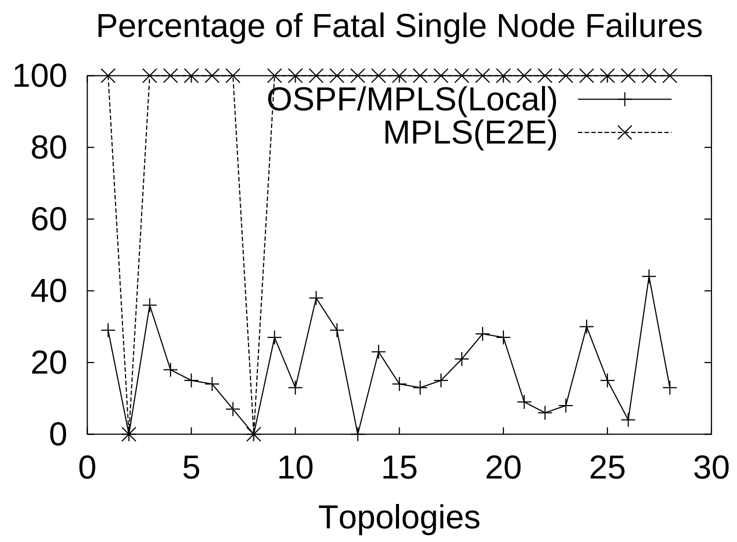


Figure 4.3: Percentage of single node failures resulting in topology partition

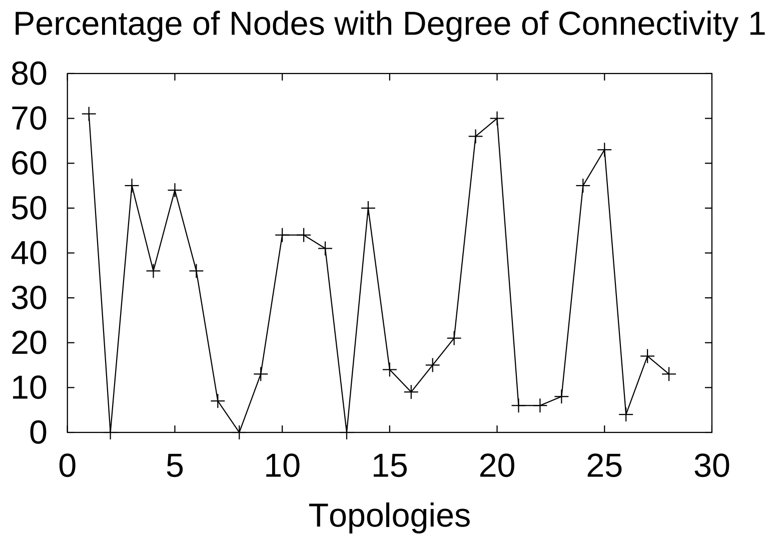


Figure 4.4: Percentage of nodes with degree of connectivity 1

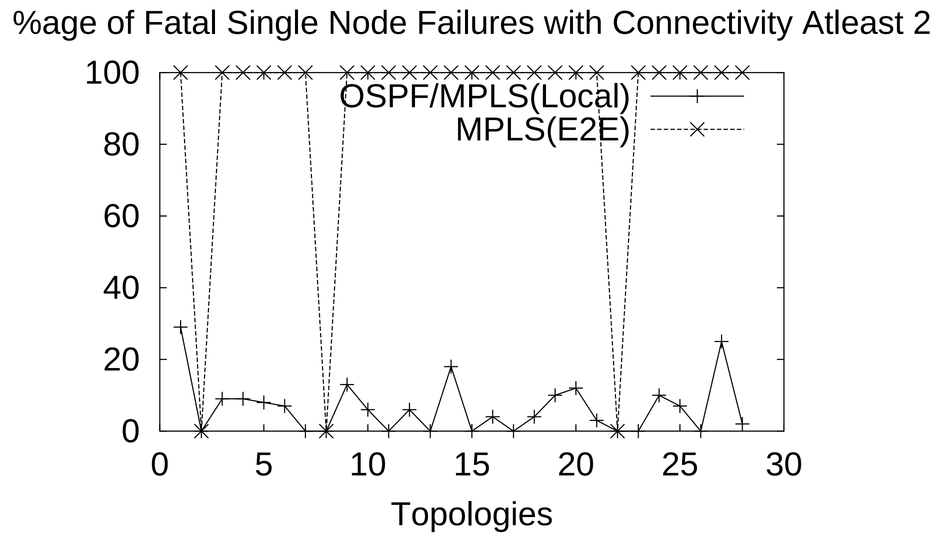


Figure 4.5: Connecting each node to at least 2 other nodes is not sufficient to avoid fatal failures

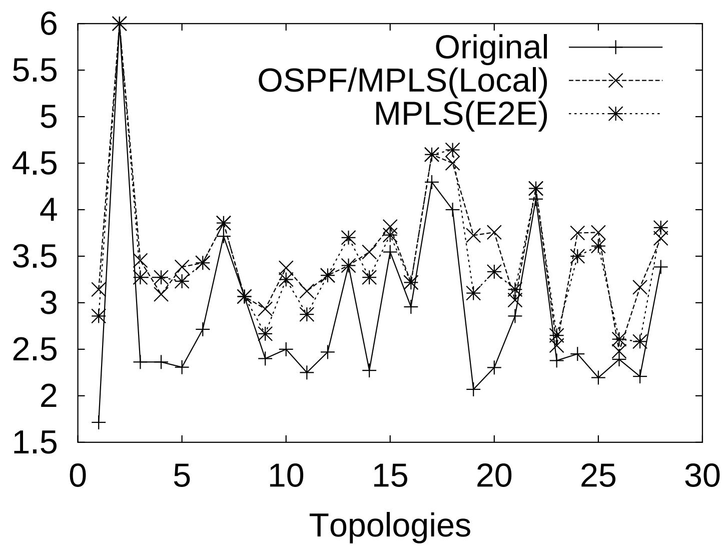


Figure 4.6: Average connectivity after link additions to achieve single node failure protection for each topology

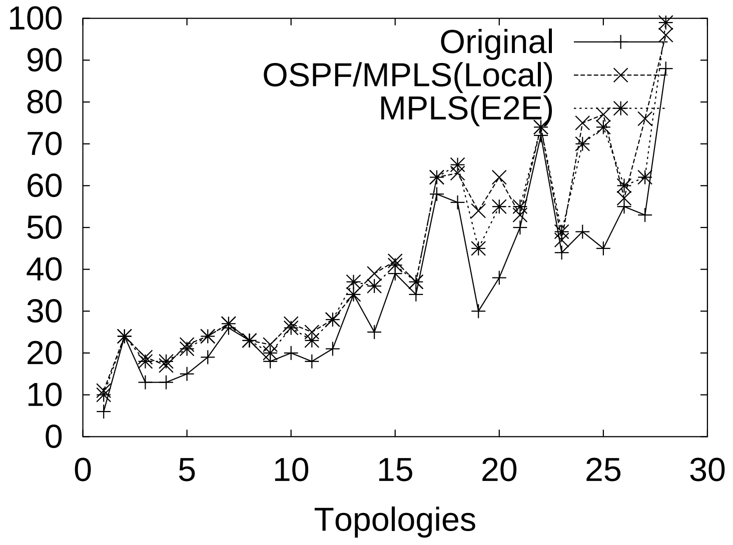


Figure 4.7: Number of links in topologies before and after link additions to achieve single node failure protection.

each node with connectivity 1 to the closest node which is not already connected to it. The distance between the nodes was calculated using the latitude and longitude information about the node locations. Figure 4.5 shows the resulting decrease in the percentage of fatal failures. As Figure 4.5 illustrates, increasing the minimum node connectivity to 2 decreases the percentage of fatal failures, but does not eliminate them. In fact, with MPLS end-to-end recovery, this step only succeeds in making one additional topology resistant against single node failures. For the remaining topologies, it is still difficult to find node-disjoint primary and backup paths for all the connections for MPLS end-to-end recovery.

An alternative strategy to achieve protection against all single node failures is to continue adding links to a topology until it can survive all single node failures. For MPLS end-to-end recovery, this means adding randomly selected links to the topology till all connections can be assigned node-disjoint primary and backup paths. After the link additions step, we remove those newly added links that are no longer required to provide node-disjoint primary and backup paths to all the connections. For OSPF and MPLS based local recovery, the link additions to achieve single node failure protection followed the following procedure:

```
for each node in the topology
{
   fail the node;
   calculate the number of “disconnected” node pairs;
   while there are “disconnected” node pairs
   {
      add a link between randomly selected neighbors
      of the node;
      calculate the number of “disconnected” node pairs;
      retain the new link if “disconnected” node pairs
      reduce in number;
   }
   for each newly added link
   {
      remove the link if it does not cause any node pair
      to become “disconnected”;
   }
}
for each newly added link to the topology
{
    remove the link if it is not necessary to avoid
    "disconnected" node pairs for any single node failure;
}
```

Figure 4.6 compares the resulting average connectivity of the topologies with the original average connectivity values and Figure 4.7 shows the actual number of links in the original and modified topologies. It can be seen from the Figure 4.6 that the topologies with low average connectivity required a substantial number of new links to achieve protection against single node failures. Note that OSPF and MPLS based local recovery can use the same set of new links for protection against single node failures, whereas a different set is required for the MPLS based end-to-end recovery mechanism. Further, the average connectivity levels required for single node failure protection in OSPF and MPLS local recovery are comparable to those required for MPLS end-to-end recovery.

### 4.3.2 Cost of Survivability

Having achieved survivability against single node failures by adding links to the topologies, we proceed to estimate the cost of survivability in terms of the extra capacity requirements. The cost of network operation is influenced by a number of parameters including the initial investment required in establishing nodes/links and the continuing expenditures involved in maintaining the network. [19] discusses one particular cost model relevant to optical WDM networks. The focus of this work lies in improving the survivability characteristics of the existing IP backbone networks. In our analysis, we have used a cost model that considers only the ongoing cost of network maintenance and ignores the one-time costs of establishing nodes/links. Moreover, we assume that the continuing cost of network operation primarily consists of cost of operating the links, i.e., the cost of leasing capacity for the backbone links over underlying fiber networks. The experiments and analysis presented in this chapter rests on such a model of the cost of network operation.

While calculating the cost of network operation, it is important to consider both the capacity required on a link to accommodate all the traffic coming its way for all possible failure scenarios as well as the characteristics of the link such as the distance spanned by the link. For example, a trans-continental link between Los Angeles and New York City is potentially more expensive than a much shorter link of the same capacity between Washington DC and New York City. A number of such considerations may determine the actual cost of the capacity required on a link. In this chapter, we use the distance spanned by the link as the scaling factor for the cost of the link, i.e.

$$\text{link cost} = \text{link capacity} \times \text{link distance} \tag{4.1}$$

While the actual distance traversed by a link depends on the underlying fiber network, a rough estimate can be obtained by calculating the geographical distance between the end nodes using their latitude and longitude values. We have assumed that the required capacity on a link in a given scenario is simply the sum of the average traffic load of all the “connections” (i.e., traffic belonging to a particular source-destination pair) passing through the link. [11] Therefore, the required capacity on a link for protection against all single node failures is the maximum of the capacities required on the link for every failure scenario as well as for failure free operation.

[11]We verified with packet level simulations that, when the number of connections passing through a link is large enough (about 20 or more), *statistical multiplexing* ensures that a link capacity equal to the sum of average traffic loads of all the connections is sufficient to accommodate the variations in the traffic load of individual connections even for very bursty traffic load distributions.

The overall cost of network operation is calculated as follows:

$$\text{cost} = \sum_{\forall i} \text{capacity}_i \times \text{distance}_i \tag{4.2}$$

where $i$ is a link in the network. The cost of survivability is calculated as the ratio of cost of network operation, defined above, with and without the protection against failures, i.e.,

$$\text{surv\_cost} = \frac{\text{cost}_{protection}}{\text{cost}_{no\_protection}} \tag{4.3}$$

Henceforth, we use the term *surv_cost* to refer to the cost of survivability defined above for protection against single node failures and the term $cost_{protection}$ to refer to the cost of network operation (as defined in Equation 4.2) for protection against single node failures.

We calculate the *surv_cost* metric for different average traffic loads between source-destination node pairs. Previous work on the nature of network traffic indicates that the average traffic loads between source-destination pairs in a network follows a zipf-like distribution [44] which means that the average traffic load ($load_r$) of the $r$the largest connection will be inversely proportional to its rank $r$, i.e., *load*$_r = 1/r^a$ , for some $a$. In our experiments, we use a zipf distribution (with $a = 0.4$) to determine the average traffic loads between the source-destination pairs and employ five random assignments (using different seed values) of these zipf-distributed values to the source-destination pairs. To make sure that our results are generally valid, we repeat the experiments with average traffic loads determined according to a uniform distribution and assigned to the source-destination pairs randomly using five different seed values.

In order to compare the *surv_cost* values associated with different protection mechanisms, it is important to not only use the same traffic matrix, but also the same network topologies. Hence, we augment the links in the topologies using the strategies mentioned earlier so that each topology can survive failures for both OSPF and MPLS based protection mechanisms. Figure 4.8 compares the average degree of connectivity in the original topologies with new topologies that can survive single node failures irrespective of the protection mechanism. Henceforth, we refer to these topologies as *basic failure-resistant* topologies. Note that the cost of network operation without any survivability requirements will be the same for both OSPF and MPLS based mechanisms since all of them use shortest path routing to calculate the primary paths of the connections[12]. Also, it should be noted that all the *basic failure-resistant* topologies use *unit* link weights. This choice was primarily dictated by the the lack of information regarding the actual link weights used on different topologies. Unit link weights mean that the calculated shortest paths are actually *minimum hop* paths. Hence, the $cost_{protection}$ values calculated with unit link weights actually provide a good base line for the subsequent discussion (in Section 4.4) regarding reducing the $cost_{protection}$ value for a topology via link weight adjustments.

Before discussing the experimental results regarding the *surv_cost* values with different protection mechanisms, we briefly describe the two different path selection methods used in our experiments for MPLS based end-to-end recovery. Both methods employ the shortest path between source and destination nodes as the primary path of a connection.

1. In the first method, the backup path is selected by excluding all the nodes on the primary path (except the source and destination) and then calculating the shortest path from the source to the destination using the given link weights. We refer to this method as *MPLS end-to-end recovery with shortest path routing* (or *E2E_SPR* in short).

[12]OSPF splits traffic among multiple shortest paths and hence the cost of network operation for no failure protection may not be exactly equal for both the OSPF and MPLS based approaches. However, in our experiments, we found this difference to be very small.

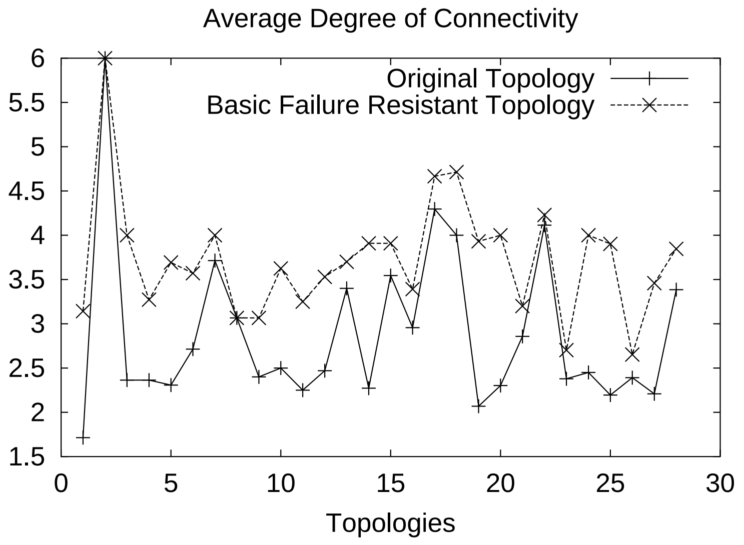


Figure 4.8: Average degree of connectivity in original topologies and basic failure resistant topologies obtained after link additions to achieve single node failure protection.

2. The second method is based on a scheme proposed by Li et. al. [87] and is designed towards reducing the extra capacity required for backup paths. In this scheme, before calculating the backup path for a connection, the link weights are adjusted in the following manner:

   (a) A link is assigned weight $\infty$ (i.e., a very large number) if the either end of the link lies on the primary path of the connection (excluding the source and destination nodes);

   (b) A link is assigned weight $\epsilon$ (i.e., a very small positive number) if it has enough capacity to accommodate the connection's traffic load in case of any failure on the primary path of the connection (in addition to accommodating other traffic that will be routed towards this link in those failure scenarios);

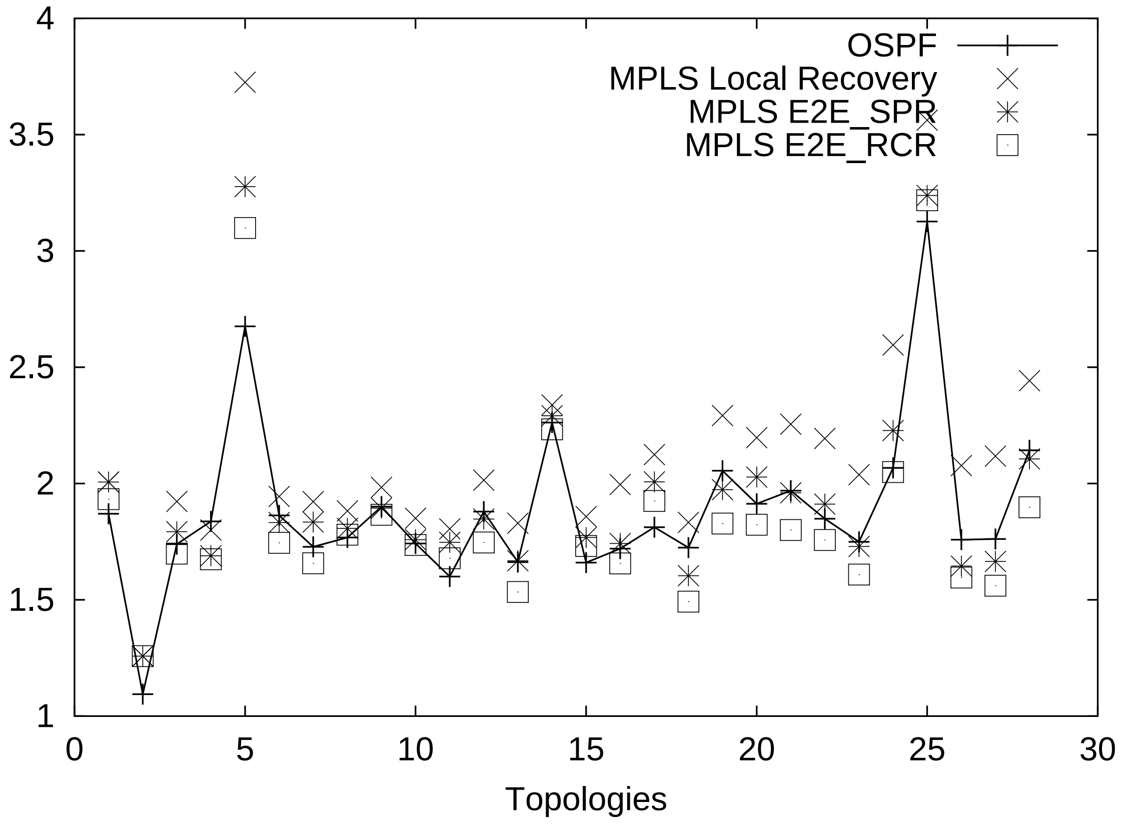


Figure 4.9: Cost of survivability for different protection mechanisms for zipf traffic loads

(c) The remaining links are assigned weights equal to the original weight of the link scaled by the *maximum extra capacity* required on the link to accommodate the traffic load of the connection in the case of any failure along the primary path of the connection.

After assigning the link weights in this manner, a shortest path from the source to the destination node is calculated and used as the backup path of the connection. We refer to this method as *MPLS end-to-end recovery with reduced capacity requirements* (or *E2E_RCR* in short).

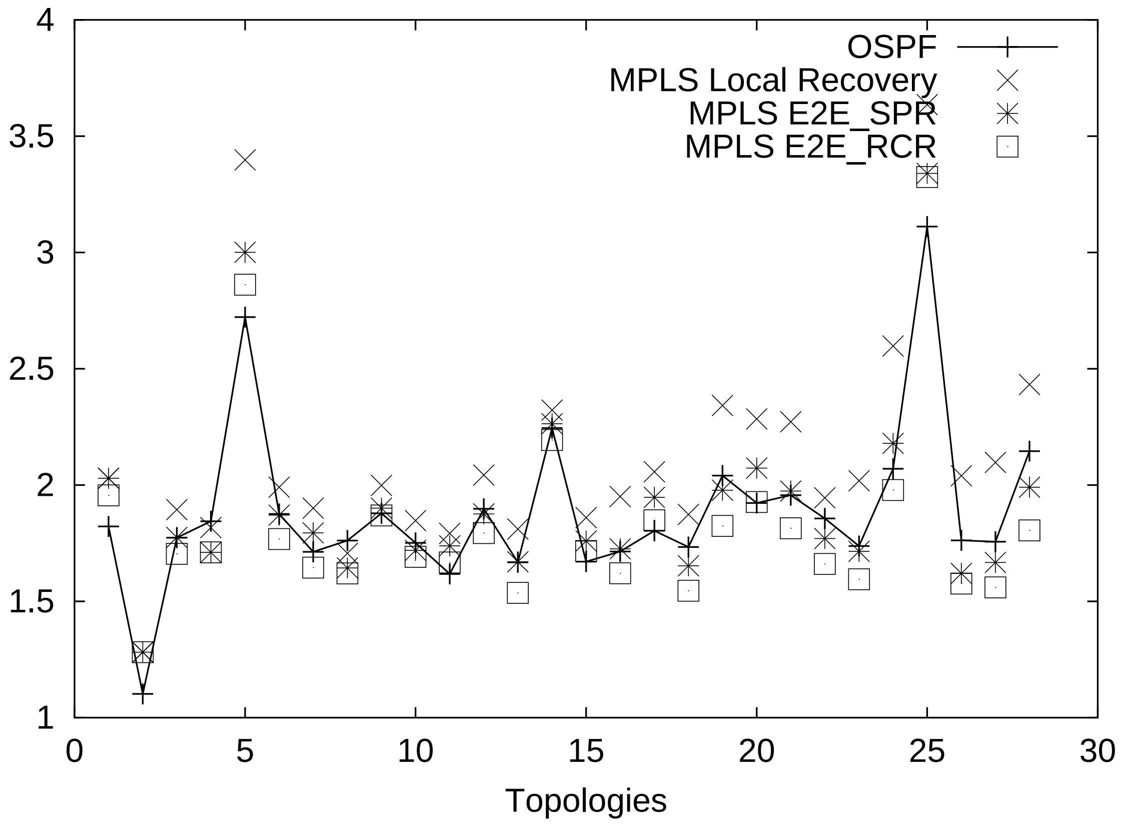


Figure 4.10: Cost of survivability for different protection mechanisms for uniformly distributed random traffic loads

We begin the cost of survivability discussion by comparing the *surv_cost* values for different protection mechanisms. Figure 4.9 illustrates the survivability costs (*surv_cost*) for average traffic loads calculated using the zipf distribution and Figure 4.10 shows the *surv_cost* values for uniformly distributed random traffic loads. The values shown in Figures 4.9 and 4.10 are the average *surv_cost* values obtained from five random assignments to connection traffic loads from the zipf and uniform distributions. The points showing the *surv_cost* values for OSPF based recovery are connected together so as to allow for easy comparison of the OSPF *surv_cost* with other protection mechanisms.

In the following we discuss some of the salient observations made from the Figures 4.9 and 4.10 regarding the cost of survivability as measured by *surv_cost* metric:

1. MPLS based local recovery is always observed to incur a higher cost of survivability than any other protection mechanism.

2. *E2E_RCR* leads to a somewhat lower cost of survivability than *E2E_SPR*.

3. No conclusion can be drawn regarding the comparative cost of survivability for OSPF based recovery and and two MPLS end-to-end recovery mechanisms (*E2E_RCR* and *E2E_SPR*).

4. The cost of survivability depends essentially on the topological characteristics. The actual traffic loads between source-destination pairs have little impact on the cost of survivability.

Let us examine these results in more detail. As expected, MPLS based local recovery is more expensive than the other protection mechanisms we studied here. With MPLS based local recovery, the nodes in the neighborhood of the failed node tunnel the affected traffic around the failed node. These tunnels are typically the shortest paths between the neighbors of the failed node. However, the effective end-to-end backup path taken by a connection in MPLS local recovery may turn out to be much longer than the backup paths used in either OSPF based recovery or any of the two MPLS end-to-end recovery schemes. However, MPLS based local recovery can potentially provide much faster failure recovery than either OSPF or MPLS based end-to-end approaches.

Among the two MPLS end-to-end based schemes, we observed that, depending on the topology, the *E2E_RCR* resulted in 0% to 10% reduction in the *surv_cost* values over *E2E_SPR*. However, these numbers are by no means conclusive since the performance of *E2E_RCR* depends on a number of factors not considered in these experiments, such as the order of arrival of the connection requests. The *E2E_RCR* scheme may also do better with other topologies that have a larger choice of alternate paths between source-destination pairs. Note that using the *E2E_RCR* scheme requires detailed information about the traffic load on each link in various failure (and failure free) scenarios. It has been proposed that traffic engineering extensions to OSPF protocol (OSPF-TE) will make the required information available throughout the network via special TE LSAs [75]. Thus, the usability of *E2E_RCR* depends on the deployment of protocols, such as OSPF-TE, in the network.

Now we discuss the comparative cost of survivability for OSPF based recovery and the two MPLS end-to-end recovery schemes. In OSPF based recovery, there is no need for each connection to have capacity reserved along both a primary path and a node-disjoint backup path. The new path(s) followed by a connection after the failure is the new shortest path(s) between the source-destination node pair in the topology after the failure. Thus, in OSPF, some of the links (and hence the capacity) belonging to the connection path before the failure can be reused after the failure. This should help in reducing the capacity requirements for survivability in OSPF based recovery when compared to MPLS end-to-end recovery. However, in many cases the need for capacity reservation along two paths in MPLS end-to-end recovery may not be that expensive, especially if the same backup capacity can be shared across multiple connections that do not fail simultaneously. Moreover, MPLS end-to-end recovery allows for the use of innovative schemes like *E2E_RCR* which can manipulate link weights to improve the sharing of backup capacity among connections in the network, thereby reducing the capacity requirements for survivability.

The experimental results regarding the comparative cost of survivability for OSPF based recovery and two MPLS end-to-end recovery mechanisms (*E2E_RCR* and *E2E_SPR*) did not indicate a clear trend. In 17 out of 28 topologies, OSPF based recovery had a lower *surv_cost* value than *E2E_SPR* scheme while in 20 out of 28 topologies, *E2E_RCR* had a lower *surv_cost* value than OSPF based recovery. A deeper analysis of the experimental results (Figure 4.11) showed that, for all but two of the largest 11 topologies, *E2E_RCR* resulted in 5% to 13% savings in the *surv_cost* value over OSPF based recovery. However, for smaller topologies, *E2E_RCR* can result in as much as 16% higher *surv_cost* value than OSPF based recovery. Regarding the comparison between *E2E_SPR* and OSPF based recovery, we observed that *E2E_SPR* could reduce the *surv_cost* value by as much as 8% or increase it by as much as 22% over the value with OSPF based recovery. Though the results are inconclusive, it is clear that MPLS end-to-end recovery is not a clear winner over OSPF based recovery as far as the cost of survivability is concerned. This issue has to be considered when making a choice between using the largely OSPF based routing infrastructure prevalent in current IP backbone networks versus using MPLS. These results also point out to the need to understand, more deeply, the topologies and scenarios where MPLS based

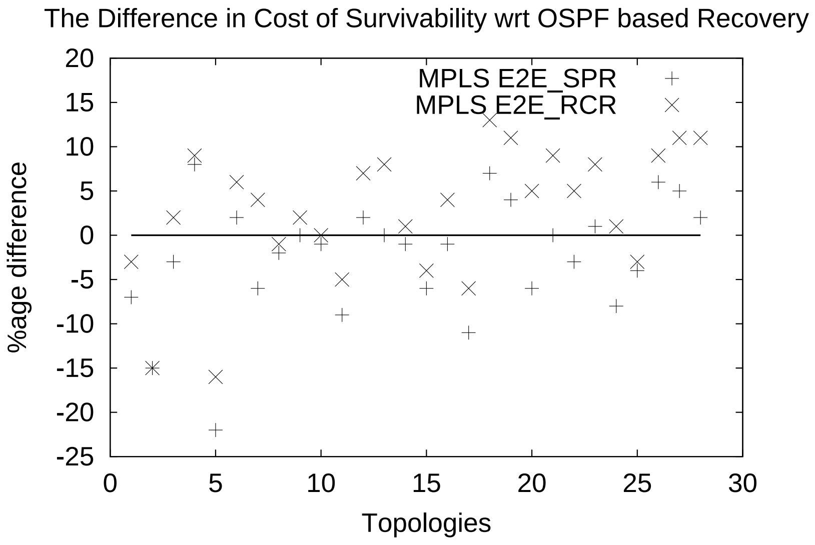


Figure 4.11: Comparing cost of survivability for OSPF based recovery and MPLS end-to-end recovery (zipf traffic matrix, actual link distances)

end-to-end recovery results in a lower cost of survivability than OSPF based recovery or vice versa.

**Sensitivity to Traffic Matrix Variation**

The most remarkable result observed in these experiments is the insensitivity of the cost of survivability to the connection traffic loads. That is, changes in traffic matrix cause only a minor change in the cost of survivability. Note that the capacity requirements for survivability indeed depend on the traffic matrix, however, the cost of survivability (the *surv_cost* value as defined in Equation 4.3) seems largely not to be impacted by the traffic matrix. The first evidence of the marginal influence of the traffic matrix variations on the *surv_cost* values emerged when changing the connection traffic loads from one random assignment of zipf distribution (or uniform distribution) values to another random assignment. This change resulted in only a small change in the *surv_cost* value for the topology. This was

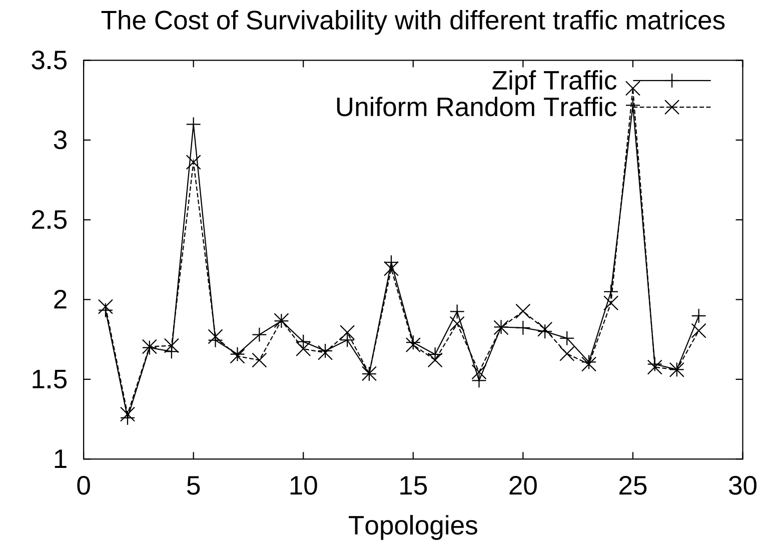


Figure 4.12: The cost of survivability does not change significantly as the zipf traffic matrix is replaced with uniform random traffic matrix (results with MPLS E2E_RCR routing).

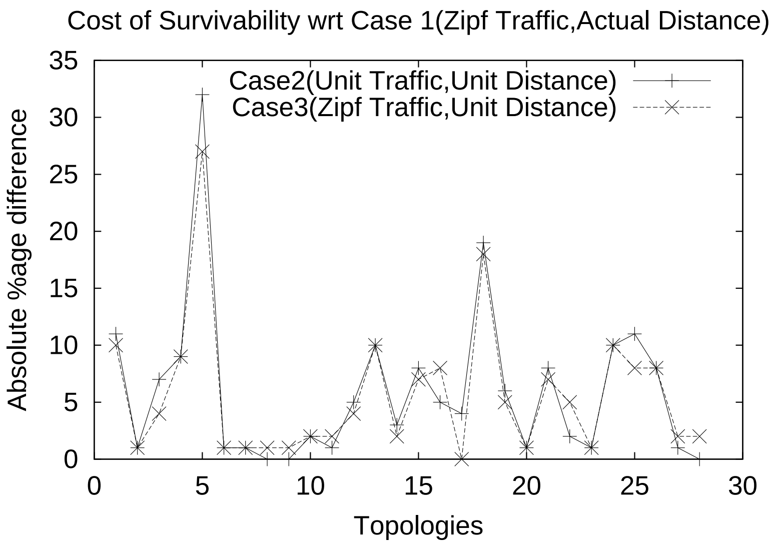


Figure 4.13: Evaluating the impact of traffic matrix and link cost factor on the cost of survivability (results with MPLS E2E_RCR routing).

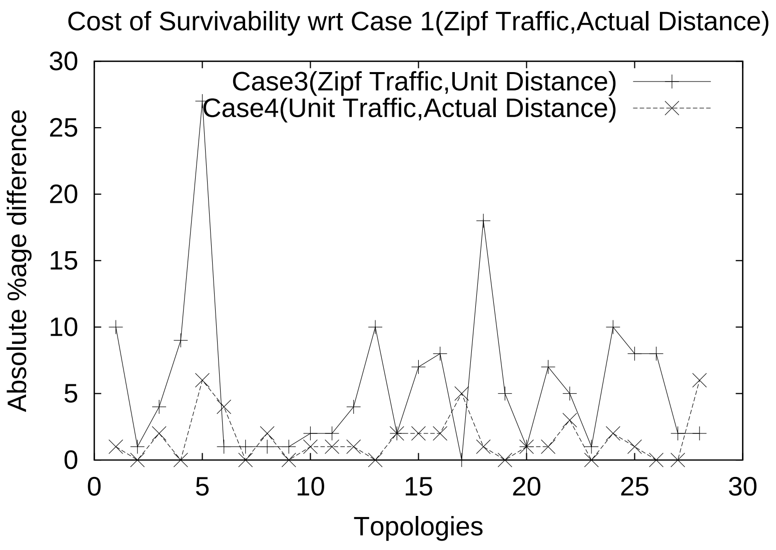


Figure 4.14: Evaluating the impact of traffic matrix and link cost factor on the cost of survivability (results with MPLS E2E_RCR routing).

observed to be true irrespective of the protection mechanism used. More significantly, the cost of survivability for a topology changed only a little when the zipf distributed traffic loads are replaced by uniformly distributed random traffic loads. Figure 4.12 illustrates these results for MPLS end-to-end recovery using *E2E_RCR* scheme. Similar results were observed for other protection methods as well.

It appears that the structure of the topology and the cost of operating a link (in our case the link distance) are the main determinant factors for the cost of survivability. For verification, we calculated the *surv_cost* value for all the protection mechanisms in the following four cases:

1. Case 1, where the connection loads are assigned randomly from a zipf distribution (the same five load assignments were used as reported earlier) and the link distances are determined according to the latitude/longitude information;

2. Case 2, where each connection has a unit load and each link has a unit distance;

3. Case 3, where the connection loads are zipf distributed (same assignments as in Case 1) and each link has a unit distance;

4. Case 4, where each connection has a unit load and the link distance are determined according to the latitude/longitude information.

Comparing the cost of survivability in the four cases should help us understand the relative impact of the topology, the cost factor (the link distance) and the traffic matrix on the the cost of survivability. In Figure 4.13, we illustrate the *absolute* percentage difference in the *surv_cost* values for Cases 2 and 3 compared to Case 1 for MPLS end-to-end recovery with *E2E_RCR* scheme. Similar results were obtained for other protection methods as well. In Case 2, since we have minimized the impact of traffic matrix and link cost factors by having unit values for connection traffic loads and link distance, the comparison between Case 1 and Case 2 should capture the impact of topological structure on the cost of survivability. It can be seen from Figure 4.13 that for most of the topologies, Case 2 has a *surv_cost* value within 10% of the *surv_cost* value in Case 1. Hence, it appears that the structure of the topology is the biggest factor in determining the cost of survivability.

To ascertain the impact of traffic matrix variations on the cost of survivability, we look at the curves showing comparative *surv_cost* values for Case 2 and Case 3 with respect to Case 1. It can be seen in Figure 4.13 that the *surv_cost* value for Case 3 is not much different from the *surv_cost* value for Case 2. Thus, changing the traffic matrix from unit connection loads(Case 2) to zipf distributed connection loads (Case 3) does not seem to affect the cost of survivability or move it closer towards the cost of survivability for Case 1. The limited impact of the traffic matrix is confirmed by examining Figure 4.14. In Figure

4.14, we illustrate the *absolute* percentage difference in the *surv_cost* values for Cases 3 and 4 compared to Case 1 for MPLS end-to-end recovery with *E2E_RCR* scheme. Figure 4.14 shows that Case 3 with zipf traffic loads and unit distances is much farther from Case 1 *surv_cost* values than Case 4 which uses unit traffic loads but actual link distances. Clearly, traffic loads are not as important in determining the cost of survivability as the link cost factor (the actual link distance, in our case). The results shown in Figures 4.13 and 4.14 are for *E2E_RCR* experiments. However, similar results were obtained for other protection mechanisms as well.

In conclusion, our experiments indicate that the topological structure has the biggest impact on the cost of survivability. The particular cost model in use can also influence the cost of survivability although not as much as the topological structure. The traffic matrix on the network has only a minimal impact on the cost of survivability.

This section illustrated the need for better connectivity in the network in order to achieve the survivability goals and examined the cost of survivability for different protection mechanisms. The question that arises next is whether we can reduce the cost of network operation for survivability ($cost_{protection}$ as defined in Equation 4.2). In the next section, we examine different alternatives to reduce this cost, such as adjusting link weights and carefully introducing new links in the topology.

## 4.4 Reducing The Cost of Network Operation For Survivability

In this section, we examine ways to reduce the cost of network operation for protection against single node failures, beyond what was observed in the previous section. In the previous section, we observed that MPLS based local recovery is more expensive in terms of the cost of survivability when compared to other schemes and *E2E_RCR* scheme has a lower cost of survivability than *E2E_SPR*. Therefore, in this section, we choose to focus on two approaches: OSPF based recovery and MPLS based end-to-end recovery using *E2E_RCR* scheme. Recall that the "cost of network operation" for single node failure protection is defined as: $cost_{protection} = \sum_{\forall i} capacity_i \times distance_i$, where $i$ is a link, whose capacity is $capacity_i$ as determined for protection against single node failures.

The cost of network operation ($cost_{protection}$) can be reduced by controlling the traffic routes either via link weight adjustments or by adding new links to the topology to create additional possible routes. Link weight adjustments can be used to make low cost ( i.e. short distance) links more attractive to the traffic than high cost (i.e. long distance) links. However, because the topology may be sparsely connected, often long and circuitous routes are the only possible choices. In such scenarios, link weight adjustments may not be useful. Careful introduction of a few new links to the topology can provide direct connections between key transit points which can significantly reduce the overall cost of network operation. Changes to the traffic routes can possibly alter the survivability properties of the topology, i.e., a a hitherto failure-proof topology might become susceptible to failures. The weight adjustment and link addition processes should take precautions against such a possibility.

### 4.4.1 Weight Adjustments to Reduce Cost of Network Operation for Desired Survivability Levels

We use an iterative heuristic based search to determine the link weights that will result in a reduced cost of network operation. The heuristic used in the search is to increment the weight of the costliest link so as to make it less attractive for use in either primary or backup paths. In this regard, the cost of link is calculated using Equation 4.1. In each iteration, the link with maximum cost is identified and has its weight increased by a unit amount. The weight adjustment is made permanent if it results in reducing the $cost_{protection}$ value. Otherwise, the link weight adjustment is undone and the link is “marked” so that we do not attempt to modify its weight in a future iteration. The search process ends when all the links in the topology have been “marked”. It is possible that increasing the weight of a “marked” link may further reduce the $cost_{protection}$ value. However, avoiding such links in future iterations in the search process helps avoid loops where same sequence of link weight adjustments is tried over and over again.

In addition to incrementing the weight of the costliest links, we tried several other heuristics such as increasing the weight of the costliest link by more than a unit, increasing the weights of multiple high cost links simultaneously, decrementing the weights of one or more low cost links so as to make them more attractive and adjusting the weights such that a high (low) cost link and a low (high) cost *path* connecting the ends of the high (low) cost link have the same weight. Many other similar heuristics are possible. We observed that none of these heuristics result in a better performance than incrementing the weight of the costliest link. In general, the heuristics involving the weight adjustment of multiple links or significant change in the weight of a link were not very useful. This is because more than a small change in the link weight distribution can significantly alter the traffic distribution on the links in very complex ways. Hence the simple heuristics involving only a small change at a time perform better than others.

### 4.4.2 Link Additions to Reduce Cost of Network Operation for Desired Survivability Levels

Unlike weight adjustments, the search for new links to be added to the topology to reduce the cost of network operation can be performed in a somewhat more exhaustive manner. The new link to be added to the topology is the one that will lead to minimum

$cost_{protection}$ value by its addition, among all possible new links. Since the exhaustive search for the best set of new links to be added to the topology would have involved testing a huge number of possible combinations of new links, we adopted a greedy approach where the new links are added one at a time. We add one new link at a time to the topology until we have added the desired number of new links. The important question that arises then is: how many new links should be added to the topology? While a greater reduction in the cost of network operation may be achieved by adding a larger number of new links to the topology, adding new links may not always be possible for a number of reasons. Further, new link additions soon reach a point of diminishing returns, depending on the topology, where the addition of more new links does not lead to a substantial decrease in the $cost_{protection}$ value for the network. In the following, we show the extent of reduction in the $cost_{protection}$ for the topologies under consideration, by limiting the algorithm to add at most 10% more new links.

### 4.4.3 Benefits of Weight Adjustments and Link Additions

Figures 4.15 and 4.16 show the savings in the $cost_{protection}$ values, after the topology modifications mentioned above, over the $cost_{protection}$ value with the *basic failure-resistant* topology. This is shown for both MPLS end-to-end recovery using *E2E_RCR* scheme and OSPF based failure recovery respectively. In each figure, 3 curves are shown:

1. cost savings achieved solely with weight adjustments,
2. cost savings solely with 10% new link additions to the topology and
3. cost savings with 10% new link additions followed by weight adjustments.

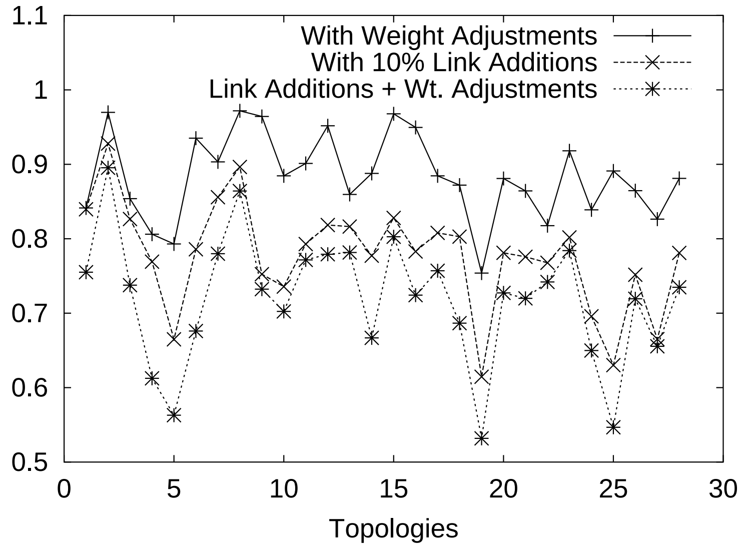


Figure 4.15: Savings in the cost of network operation for survivability with weight adjustments and link additions (MPLS E2E Recovery with *E2E_RCR* routing, zipf traffic loads)

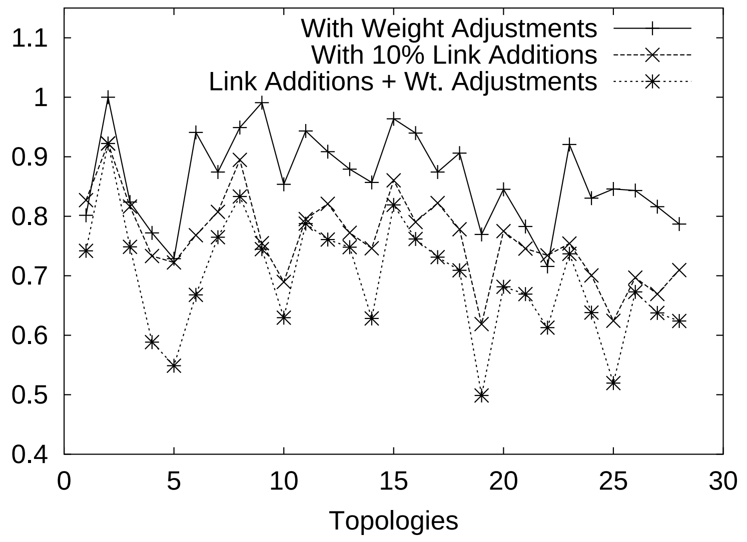


Figure 4.16: Savings in the cost of network operation for survivability with weight adjustments and link additions (OSPF Recovery, zipf traffic loads)

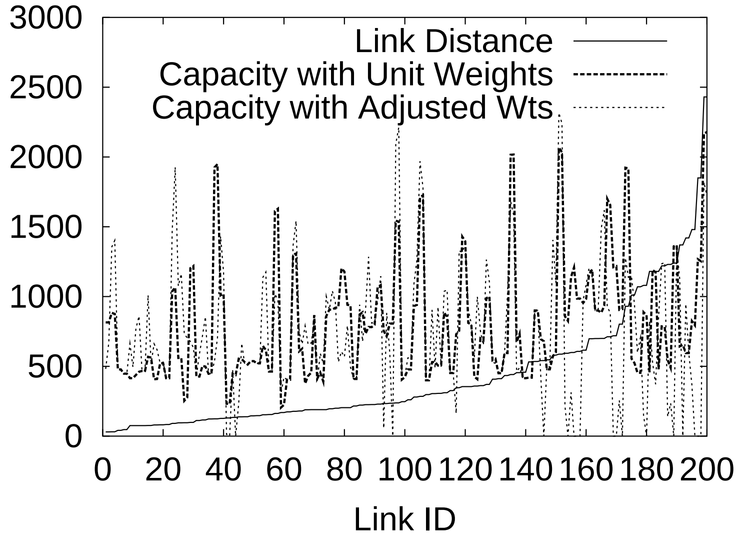


Figure 4.17: Weight adjustments shift traffic from long links to short links (Topology 28, OSPF based recovery, Zipf Traffic)

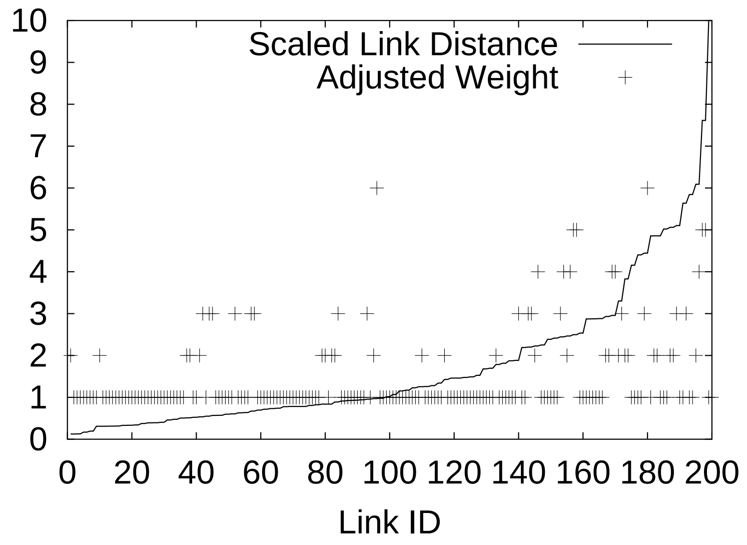
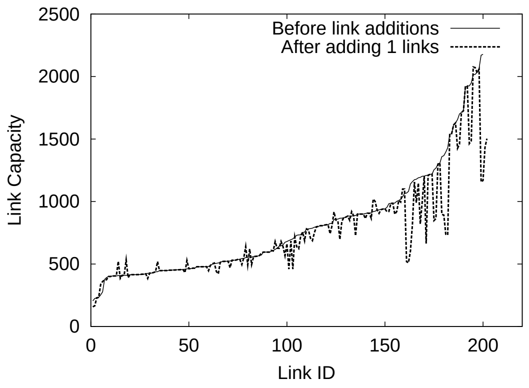
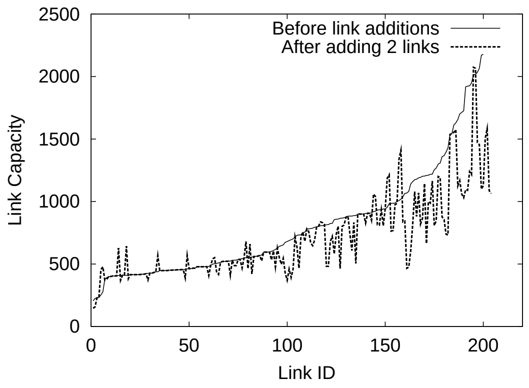


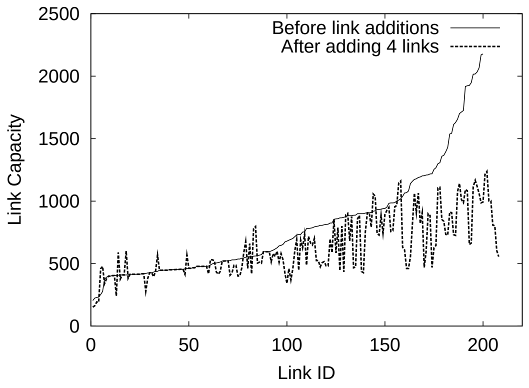
Figure 4.18: Weight adjustments lead to increased weights for long distance links (Topology 28, OSPF based recovery, Zipf Traffic)



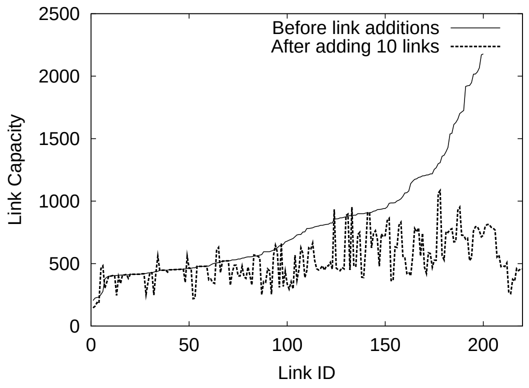
(a) After 1 link additions



(b) After 2 link additions



(c) After 4 link additions



(d) After 10 link additions

Figure 4.19: Reductions in required link capacities for single node failure protection as we add new links to topology 28 (OSPF based recovery, zipf traffic)

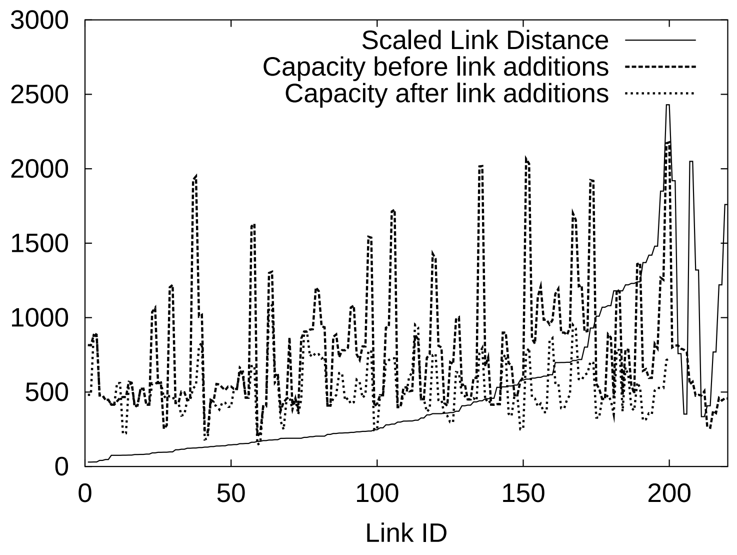


Figure 4.20: Required link capacities (for single node failure protection) before and after adding 10percent new links to topology 28 for OSPF based recovery. The original links are sorted according to link distance.

It is clear from these figures that the weight adjustments and link additions can significantly reduce the $cost_{protection}$ for both OSPF and MPLS end-to-end recovery. It is particularly true when both weight adjustments and link additions are applied together as we described above. When only one approach is used, it appears that new link additions are more effective in reducing the $cost_{protection}$ compared to weight adjustments. As illustrated in Figure 4.17, the weight adjustments are successful in shifting capacity requirements from long links to the shorts links, thereby decreasing the $cost_{protection}$ value. It appears that the net effect of weight adjustments is to cause the weights of long distance links to increase while most of the links continue to have same (unit) weight as before (Figure 4.18). It is interesting to note that the adjusted weights typically lie in a small range of values (usually 1 to 5). This, combined with the fact that for most of the small cost factor links the unit value is probably optimal, suggests that the following heuristic may be effective. To determine a set of link weights close to the optimal, only adjust the weights of the links that have a large cost factor. For even these, one may be able to limit the combinations of weights of these selected links to be within a small range.

We had observed in Figures 4.15 and 4.16 that the link additions are much more effective in reducing the $cost_{protection}$ than mere weight adjustments. Unlike weight adjustments which reduce the $cost_{protection}$ cost of network operation primarily by shifting traffic from high cost links to low cost links, new link additions are able to not only shift the traffic from high cost links to low cost links but also reduce the capacity requirements for the links. In order to understand how newly added links help in reducing the $cost_{protection}$, we examined the nature of the newly added links and observed that the newly added links connect nodes that were otherwise connected by multiple hops of long distance links. The new links are typically added between nodes that act as transit points for significant amounts of traffic. Thus, the new links provide direct paths for the traffic that would have otherwise traveled through multiple hops of long distance links. In this manner, the newly added links can significantly reduce the link capacity requirements and hence the $cost_{protection}$. Figure 4.19 illustrates how the link capacity requirements decrease for topology 28 as new links are added to the topology. In this figure, the links are sorted in the increasing order of their capacity requirements with original *basic failure-resistant* topology without any new link additions. Figure 4.20 illustrates the same information as Figure 4.19 from a different perspective. Here, we sort the (original) links according to their increasing distance and also show the relative distance spanned by these links. It can be seen from Figure 4.20 that new link additions reduce capacity requirements for a large number of links including most of large distance (i.e., high cost factor) links which results in a reduction in $cost_{protection}$.

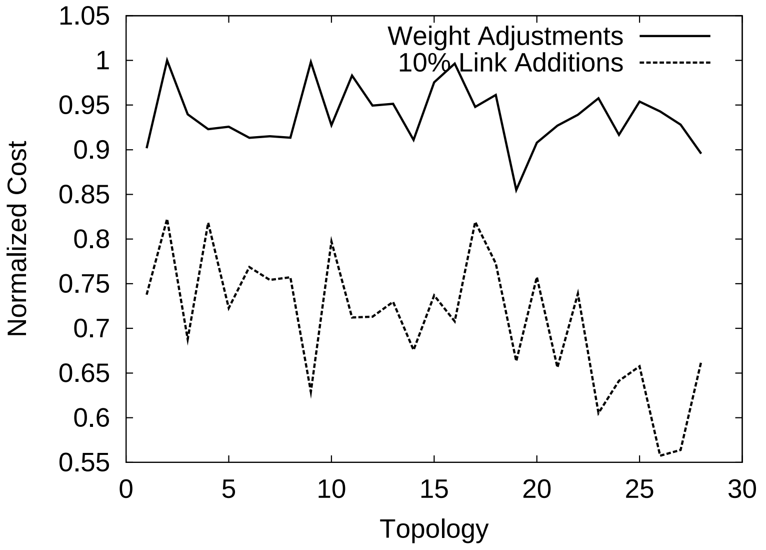


Figure 4.21: Weight adjustments and link additions result in substantial cost savings over basic failure-resistant topology even for a very different cost model (link cost = link capacity for single node failure protection) (OSPF based recovery, zipf traffic)

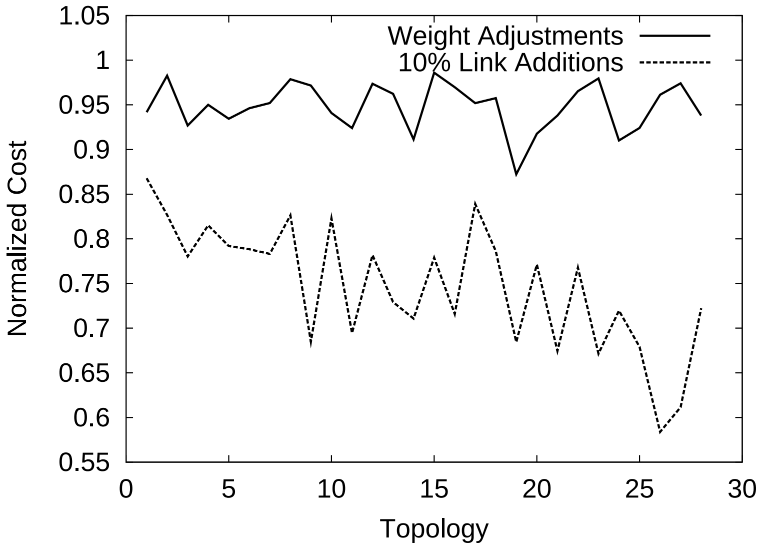


Figure 4.22: Weight adjustments and link additions result in substantial cost savings over basic failure-resistant topology even for a very different cost model (link cost = link capacity for single node failure protection) (MPLS E2E_RCR based recovery, zipf traffic)

To verify that the benefits of weight adjustments and link additions are not limited to the particular cost model we use (where link distance serves as the link cost factor) ,we performed weight adjustments and link additions on all the topologies assuming a different cost model. We let the link cost be simply the required capacity on the link for single node failure protection, i.e., *link cost* $=$ *link capacity*. Figures 4.21 and 4.22 show the resulting savings for OSPF and MPLS *E2E_RCR* scheme respectively. It is clear from these figures that the benefits of weight adjustments and link additions are not restricted to a particular cost model. We examined the nature of adjusted weights and newly added links for the simpler cost model. As can be expected, for the new cost model, the adjusted weights did not show a strong correlation to the link distance. The nature of newly added links is somewhat similar to the links added under the old cost model (defined in Equation 4.1, i.e., the newly added links connect the nodes that are otherwise connected by several hops. As before, the newly added links shift significant amounts of traffic away from the original paths that comprised multiple hops. As the new cost model does not take into account link distances, the sole emphasis of link additions lies in reducing the number of hops. Hence, in a few cases, the new cost model leads to the addition of links that are almost as long as the combined length of the multiple hop path that previously constituted the shortest path connecting the nodes in question. We conclude that for a given network cost model, the weight adjustment and link addition processes described above can identify the appropriate topological changes (i.e., adjusted weights and new links to be added) that can significantly reduce the $cost_{protection}$.

### 4.4.4 Effect of Traffic Matrix

Since topological changes such as weight adjustments and new link additions have a far reaching effect on the cost of network operation, we believe it is important to examine the sensitivity of the cost savings achieved by weight adjustments/link additions to variations in the traffic matrix. Previous work in this area suggests that the traffic load between a given source-destination pair can undergo large-scale fluctuations during the course of a day [44]. In addition, it is reasonable to expect that the overall traffic matrix in a network will change drastically within a period of few months. In such a scenario, the modifications to the topology make sense only if the cost savings persist even after drastic, unpredictable changes in the traffic matrix.

In order to understand the impact of the traffic matrix variations on the utility of the topological modifications, we examined how the $cost_{protection}$ savings (over the $cost_{protection}$ with original *basic failure-resistant* topologies) with zipf traffic matrix are affected if the weight adjustments/link additions are performed using a very different traffic matrix. For this purpose, we performed weight adjustments and link additions on the topologies using two additional traffic matrices:

1. a unit traffic matrix where the traffic load between each source-destination pair is assumed to be a unit value;

2. a uniformly distributed random traffic matrix where the traffic load between source-destination pairs is determined according to a uniform distribution.

This was followed by calculating the $cost_{protection}$ with zipf traffic matrix on the topologies thus modified assuming unit and uniformly distributed random traffic matrices. We then calculated the savings thus obtained in the $cost_{protection}$ value over the $cost_{protection}$

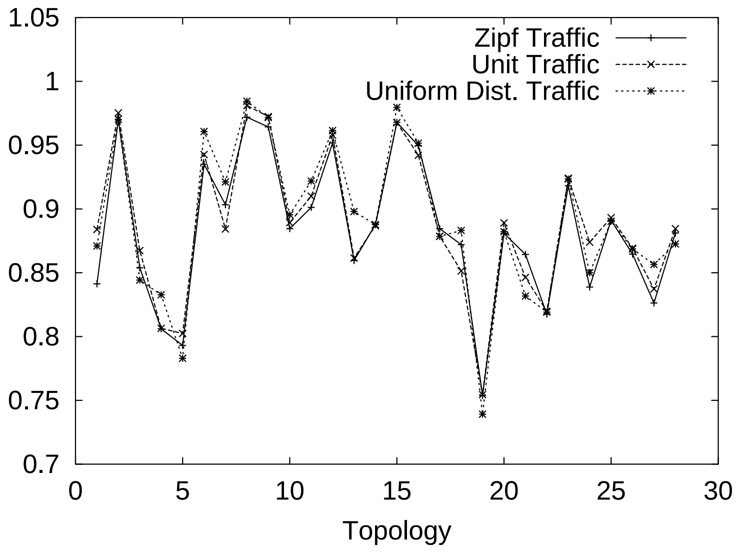


Figure 4.23: The cost savings achieved with weight adjustments over unit weights (MPLS E2E Recovery, Zipf Traffic Matrix). The weight adjustments done assuming unit and uniformly distributed traffic matrices lead to similar cost savings as weight adjustments done assuming zipf traffic matrix.

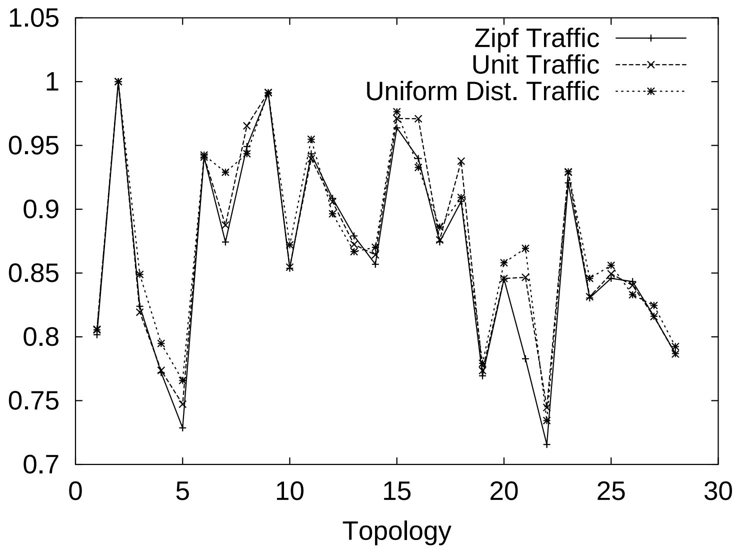


Figure 4.24: The cost savings achieved with weight adjustments over unit weights (OSPF based Recovery, Zipf Traffic Matrix). The weight adjustments done assuming unit and uniformly distributed traffic matrices lead to similar cost savings as weight adjustments done assuming zipf traffic matrix.

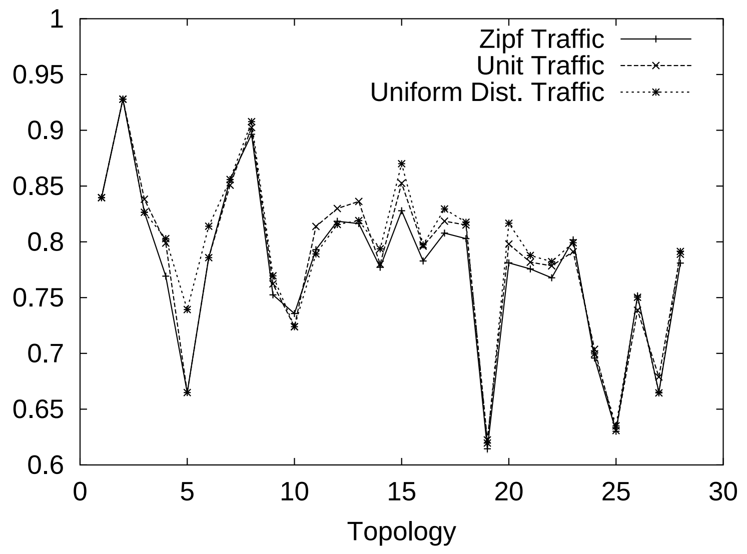


Figure 4.25: The cost savings achieved with 10 percent link additions (MPLS E2E Recovery, Zipf Traffic Matrix). The link additions done assuming unit and uniformly distributed traffic matrices lead to similar cost savings as link additions done assuming zipf traffic matrix.

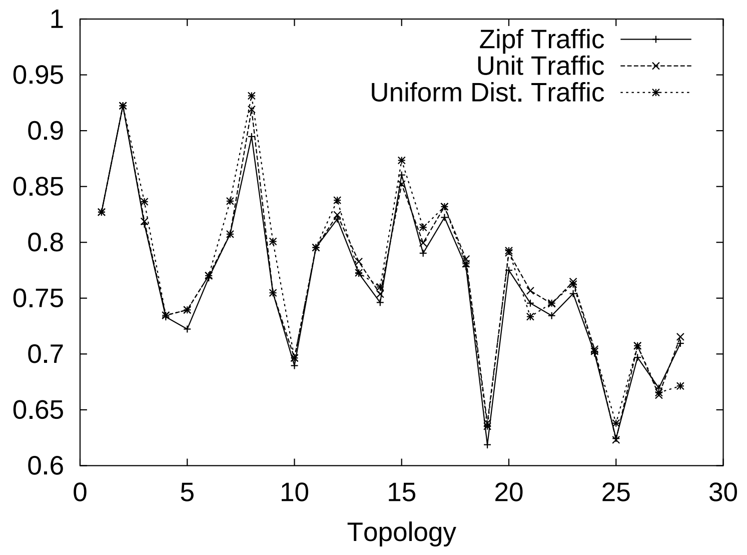


Figure 4.26: The cost savings achieved with 10 percent link additions (OSPF Recovery, Zipf Traffic Matrix). The link additions done assuming unit and uniformly distributed traffic matrices lead to similar cost savings as link additions done assuming zipf traffic matrix.

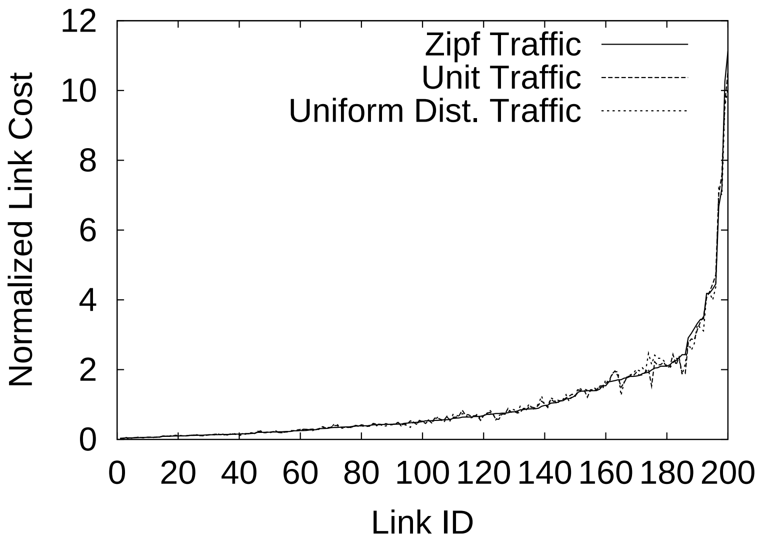


Figure 4.27: The link cost, normalized wrt average link cost, varies a little even after drastic changes in the traffic matrix. Topology 28, MPLS E2E Recovery Results. Link Cost = Link Capacity * Link Distance

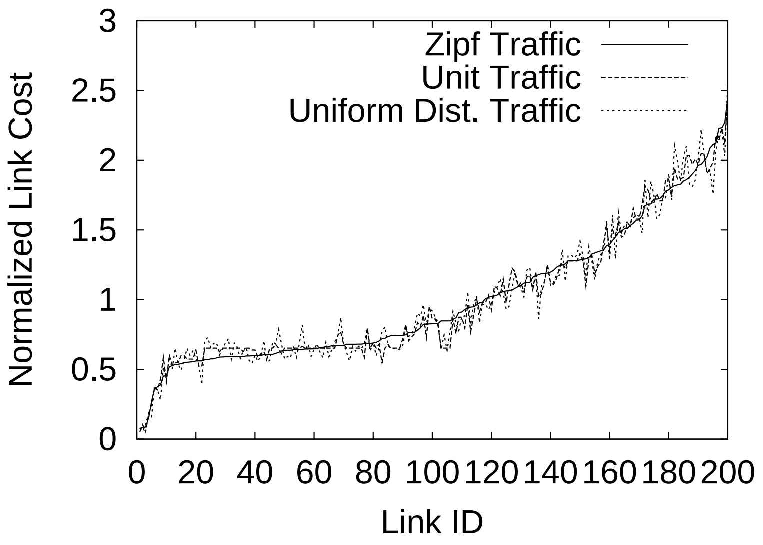


Figure 4.28: The link cost, normalized wrt average link cost, varies a little even after drastic changes in the traffic matrix. Topology 28, MPLS E2E Recovery Results. Link Cost = Link Capacity

value on the *basic failure-resistant* topologies with the zipf traffic matrix. We compared the cost savings thus calculated with cost savings obtained when the topology modifications (weight adjustments and link additions) are performed assuming the zipf matrix itself. The results are shown in Figures 4.23 to 4.26 for both OSPF and MPLS *E2E_RCR* scheme. It is clear that the topological modifications done assuming unit or uniformly distributed random traffic matrices result in very similar $cost_{protection}$ savings for the zipf traffic matrix as the topological modifications done using the zipf traffic matrix itself. The reasons behind this remarkable result become clear when we examine the curves showing the link costs (i.e., *link capacity* $\times$ *link distance*, normalized with respect to average link cost) on a *basic failure-resistant* topology for different traffic matrices (Figure 4.27). Notice that the normalized link costs are remarkably similar for very different traffic matrices. The implication of this observation is that the weight adjustment/link additions processes for different traffic matrices have similar starting points and hence result in weight adjustments/link additions that are beneficial even for other traffic matrices. We have seen earlier that the adjusted link weights have a strong correlation with the link distances and the newly added links directly connect nodes that are otherwise connected via multiple hop paths. Thus, it is intuitive to expect such topological modifications to have a beneficial impact on the $cost_{protection}$ regardless of the traffic matrix. The observed insensitivity of benefits of topological changes to the traffic matrix variations reconfirms our earlier results regarding the significant role played by the network topology (and the relative insignificance of the traffic matrix) in determining the cost of network operations. We verified that the observed insensitivity of benefits of topological modifications to the traffic matrix is not restricted to the particular cost model we have employed. Even if we assume a different cost model (*link cost* $=$ *link capacity*), where link distance does not play any part in calculating the cost of operating a link, we observed a very similar insensitivity of topological modifications to the variations in traffic matrix. This is because, even for the new cost model, the normalized link costs for different traffic matrices on the *basic failure-resistant* topologies look remarkably similar irrespective of the traffic matrix (Figure 4.28).

## 4.5 Load balancing And The Cost of Network Operations For Survivability

Load balancing in communication networks has traditionally been associated with achieving efficient utilization of network resources by traffic route adjustments that move traffic loads from high utilization (or congested) links to low utilization links. For a given topology and traffic matrix, a straight forward metric for measuring the load balance is the standard deviation among link utilization values. The smaller the standard deviation among link utilizations, the better is the load balance in the network. From quality of service point of view, it is important that link utilization values do not become very high so that the link can easily accommodate transient upsurge in the traffic loads. However, when we combine the load balancing problem with the problem of reducing $cost_{protection}$, we encounter a dilemma. The solutions to both problems involve traffic route adjustments which can work in a non-complimentary fashion, i.e. the route adjustments designed to improve the balance in link utilizations may increase the $cost_{protection}$ (or worse - make the hitherto failure-proof network susceptible to link/node failures) and vice versa. In the previous section, we have employed link weight adjustment as a means to reduce the $cost_{protection}$ with very effective results (Figure 4.16). However, such weight adjustments also cause the standard deviation among link utilization values to increase significantly (Figure 4.29). We can modify the iterative weight adjustment process described in the previous section so that the objective is to reduce the imbalance in link utilization values rather than reduce the cost of survivability. Here, the link utilization is measured as the ratio of the traffic load on the link during failure-free operation and the required link capacity for single node failure protection. The modification consists of incrementing the weight of the most heavily utilized link (rather than the costliest link). The weight adjustment will alter the traffic routes, thereby causing the required link capacities for single node failure protection and the load on the links during failure free operation to change. The new link utilization values as well as the new value of standard deviation among link utilizations are calculated. The weight adjustment is accepted if it does not make the network susceptible to single node failures and the new value of standard deviation among link utilizations is less than the old value. The results are shown in Figure 4.30. Clearly, the weight adjustments are effective in improving the balance in link utilizations.[13] However, such weight adjustments may result in increased $cost_{protection}$ for several topologies when compared to $cost_{protection}$ with unit weights. Figure 4.31 illustrates how the $cost_{protection}$ values (relative to the $cost_{protection}$ with unit weights) differ for different topologies when the objective of weight adjustments is changed from reducing the $cost_{protection}$ to reducing the standard deviation among link utilizations. Figure 4.31 tends to imply that the load balancing can be achieved only at the cost of sacrificing the savings in the $cost_{protection}$.

While it appears that the traffic route adjustments for load balancing do not automatically reduce the $cost_{protection}$ and vice versa, it will be interesting to examine how the efficacy of weight adjustments for load balancing/ $cost_{protection}$ reduction is affected if the weight adjustment process is constrained not to deteriorate the other factor (i.e. not deteriorate load balancing while adjusting weights to reduce $cost_{protection}$ and other way around).

[13]We also verified that the load balancing benefits of the weight adjustments persist in face of drastic variations in the traffic matrix.

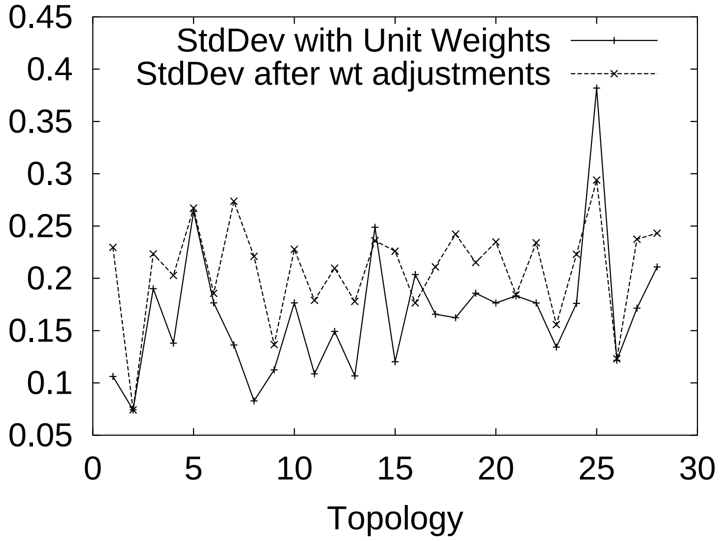


Figure 4.29: Weight adjustments to reduce cost of network operation for survivability can lead to larger standard deviation among link utilization values

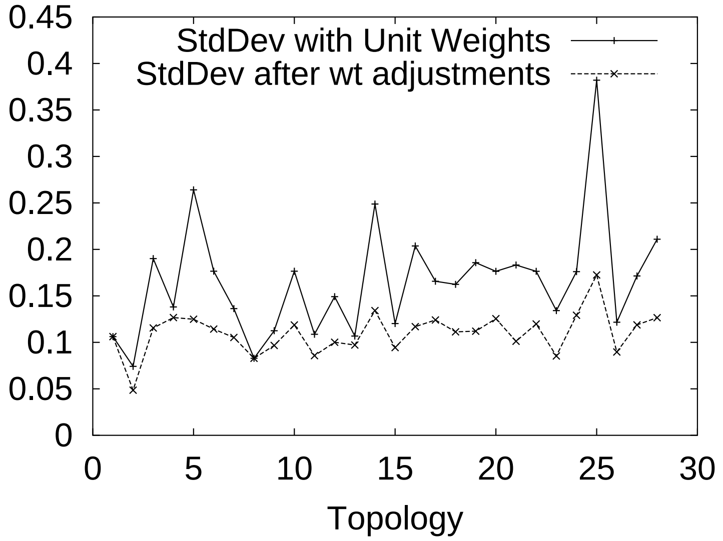


Figure 4.30: Weight adjustments can reduce the standard deviation among link utilizations significantly

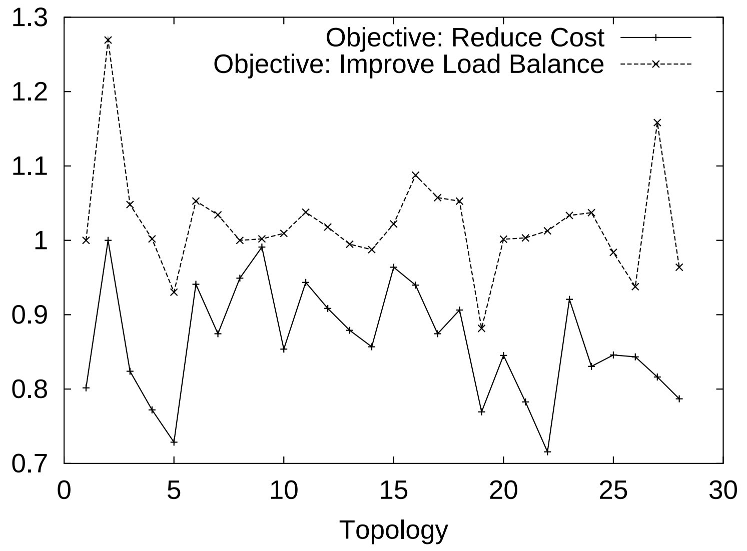


Figure 4.31: Cost of network operation for survivability (relative to the cost with unit weights) after weight adjustments for 1) reducing the cost of network operation for survivability 2) reducing the standard deviation among link utils (OSPF recovery results)

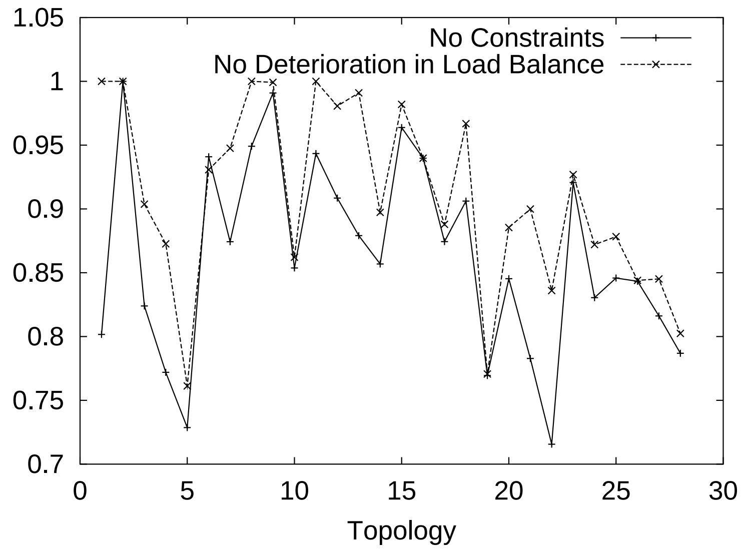


Figure 4.32: Cost of network operation for survivability (relative to cost with unit weights) after weight adjustments with and without the constraint regarding not increasing the standard deviation among link utilizations (OSPF based recovery)

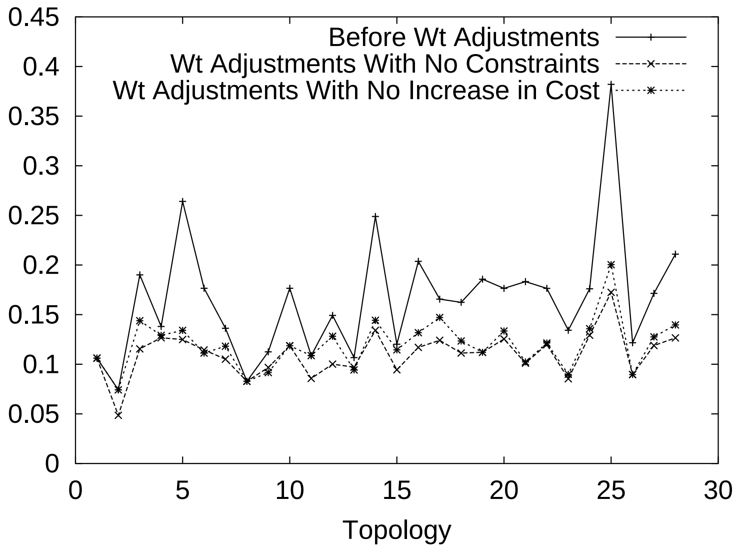


Figure 4.33: The standard deviation among the link utils before and after the weight adjustments to improve the load balance with and without the constraint regarding not increasing the cost of network operation for survivability (OSPF results)

This requires the following additional constraint in the weight adjustment process: reject a weight adjustment if it leads to deterioration in the value of the other factor beyond the original value with unit weights. Figure 4.32 shows that, for most of the topologies, the constraint regarding not deteriorating the load balance has only a minor impact on the efficacy of weight adjustments to reduce $cost_{protection}$. Figure 4.33 shows the new standard deviations in link utilizations before and after the weight adjustments to improve load balance with and without the constraint regarding not increasing the $cost_{protection}$. Clearly, the added constraint does not limit the achieved load balance significantly. Figures 4.32 and 4.33 show that while weight adjustments to reduce $cost_{protection}$ will not naturally lead to better load balance and vice versa, we can avoid deteriorating one factor while pursuing improved values for the other factor. Finally, we examined if a two step weight adjustment process will help us achieve both goals simultaneously. The first step in the two step

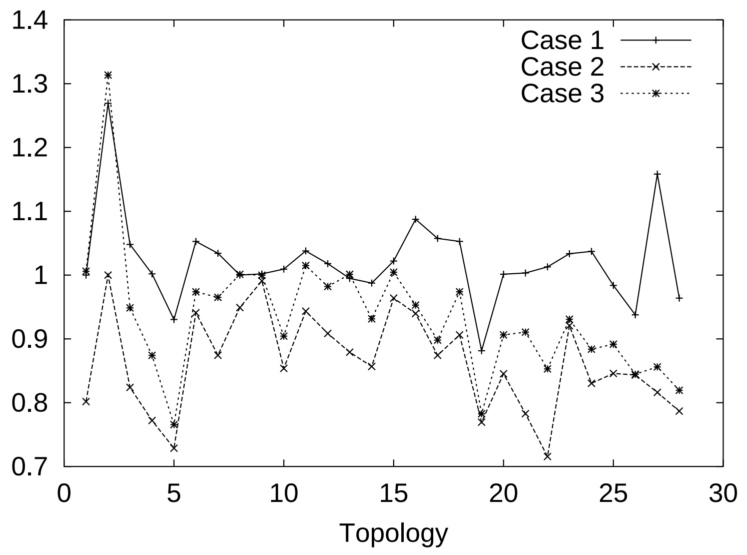


Figure 4.34: Cost of network operation for survivability after weight adjustments to 1) improve load balance (Case 1) 2) reduce cost of network operation for survivability (Case 2) 3) reduce cost without worsening load balance followed by link capacity increase to keep utilizations ¡ 0.7 (Case 3). Adjusting weights to reduce cost of network operation for survivability without deteriorating load balance followed by increasing the link capacities to limit link utilizations below the desired threshold value can be cost effective for many topologies

process consists of adjusting link weights so as to reduce the $cost_{protection}$ with out deteriorating the load balance. The second step consists of weight adjustments so as to improve the load balance without increasing the $cost_{protection}$ on the topology obtained after the first step. However, the experimental results indicate that the second step is largely ineffective with very little improvement in the load balance. On the whole, it appears that most of the times it is not feasible to achieve maximum possible reductions in $cost_{protection}$ and standard deviation among link utilizations simultaneously using weight adjustments.

Since weight adjustments can achieve either better load balance or reduced $cost_{protection}$, necessarily a choice needs to be made regarding the objective for weight adjustments.

Since load balancing can also be achieved by increasing the capacity of highly utilized links, perhaps minimizing the $cost_{protection}$ without deteriorating the load balance is an appropriate objective for the weight adjustments. Once the link weights have been adjusted so as to achieve maximum possible reduction in the $cost_{protection}$ without deteriorating the load balance, the link capacities can be increased so that all the link utilizations are below a threshold value. Increasing the link capacities will necessarily increase the $cost_{protection}$. However, the resulting increase in the $cost_{protection}$ might be less than the increase if load balancing was attempted as the primary goal of weight adjustments. In figure 4.34, we illustrate the $cost_{protection}$ after the weight adjustments are performed to decrease the $cost_{protection}$ without worsening the load balance and then increasing the link capacities so that no link utilization exceeds 0.7. Assuming that 0.7 is considered a reasonable high limit for link utilizations, it can be seen that for many topologies, this way we can achieve desired load balance with very reasonable $cost_{protection}$.

## 4.6 Summary

In this chapter, we examined different aspects of the survivability in IP networks. This work was motivated by the need to ensure that the backbone network is robust to single node failures, ensuring that connectivity between any pair of nodes in the backbone is not impacted and also ensuring that there is enough capacity to carry the existing traffic over alternate paths. We analyzed the survivability of IP networks using both OSPF routing and the current MPLS protocols using shortest path routing. We showed that the survivability, and the cost of survivability, of IP networks can be significantly enhanced using a combination of careful link additions and weight adjustments.

We first examined the connectivity characteristics of 28 different IP backbone topologies belonging to both commercial and educational ISPs and showed that a large percentage of these topologies cannot avoid a network partition in the face of some single node failures. The main reason for the observed lack of survivability is the sparse connectivity within the network. For MPLS based end-to-end recovery, in all but two of the 28 topologies, we could not assign node-disjoint primary and backup paths connecting all the node pairs in the network. Our first step was to enhance the connectivity by adding links to the topology so that it can survive all single node failures. We observed that the average connectivity levels required for single node failure protection in OSPF and MPLS local recovery are comparable to those required for MPLS end-to-end recovery.

We estimated the cost of survivability to protect against all single node failures, for traditional OSPF based recovery as well as the current MPLS based schemes, using local or end-to-end repair. In this estimation, the link cost is represented as the product of link distance and the link capacity. We proposed a novel approach of carefully enhancing the topology, to increase the survivability characteristics of the network. Among the approaches we studied, we found that MPLS based local recovery is more expensive than the other protection mechanisms. The most remarkable result we observed is the insensitivity of the *relative* cost of survivability to the connection traffic loads. We found that the structure of the topology and the cost of operating a link are the main determinant factors for the cost of survivability.

We observed that, for both MPLS based end-to-end recovery and OSPF, the cost of network operation with survivability can be significantly reduced by adjusting link weights and introducing a set of carefully selected new links to the topology. We proposed an iterative, heuristic based approach for modifying the link weights that will result in a reduction in the cost of network operation. Weight adjustments are successful in shifting capacity requirements from long links to the shorts links. We also observed that a small change in link weights can significantly alter the global traffic distribution.

We used a simple approach for choosing a limited number of additional links to reduce the cost of network operation for survivability, by adding one link at a time to the topology. Each link was chosen after an exhaustive search to determine which link reduces the cost the most. We showed that there is a significant reduction in the cost of network operation even when limiting the algorithm to add at most 10% new links. We observed that the link additions are much more effective in reducing the cost of network operation than merely adjusting the weights. The addition of a few carefully selected links can significantly reduce both the overall cost of network operation as well as the required link capacities on the individual links. Once again, we observed that the topological modification process for different traffic matrices results in weight adjustments/link additions that are beneficial even for a very different traffic matrix. We verified that the observed insensitivity of benefits of topological modifications to the traffic matrix is not restricted to the particular cost model we have employed.

Finally, we investigated the relationship between load balancing and the cost of survivability. We observed that a straightforward approach of adjusting weights for load balancing often results in sacrificing the savings in the cost of network operation for survivability. A previously failure-resistant network might become susceptible to failures after the route changes that result from the load balancing operation. We proposed an approach to perform load balancing and reducing the cost of survivability by first performing one with a constraint on the other. We examined the approach of adjusting weights with the goal of minimizing the cost of survivability without deteriorating load balancing. Once the weights were adjusted to achieve the maximum possible reduction in the survivability cost without deteriorating the degree of load balance, the link capacities were increased so that all the link utilizations are below a threshold value. We showed that in this way we can achieve the desired degree of load balancing with very reasonable survivability cost, for many of the 28 topologies we studied.

In this chapter, we have discussed several issues associated with survivability in IP networks. A significant direction for further research is to understand the relationship between the topology and the cost of survivability for different protection mechanisms. The following chapters in the thesis continue the survivability discussion by focusing on another important aspect of survivability - the speed of failure recovery.

# CHAPTER 5

# DYNAMIC TRACKING OF IP LEVEL TOPOLOGY

## 5.1 Introduction

The cost of survivability and the speed of recovery constitute the two important aspects of network survivability. In the previous chapter, we focussed on evaluating and reducing the cost of survivability in terms of extra capacity requirements on the links. In the rest of the thesis, we turn our attention to the speed of recovery from network failures. We have examined the speed of recovery problem from two different perspectives. The first perspective deals with the problem at a macro time scale and is concerned with how quickly can the network administrator detect the failure and take steps to correct it. The second perspective tackles the problem at a micro time scale level and is concerned with how quickly can the in-built network survivability mechanisms detect the failure and restore the user connectivity and service quality while the failure has not yet been corrected. The first perspective is essentially a network management problem and can be viewed as the problem of dynamically tracking the network topology so that the network failures can be quickly and reliably detected by the network administrators. This chapter focuses on the first perspective of the speed of recovery problem, which is finding a practical solution to dynamic topology tracking problem so that the network administrators can quickly detect network failures and take steps to correct the situation. The remaining chapters in the thesis consider the second perspective of the speed of recovery problem, which is improving the in-built network survivability mechanisms for fast recovery from network failures while the failures have not yet been corrected.

IP network management systems today are targeted at element level fault diagnosis and troubleshooting, and not to meet stringent needs of real-time resource and topology tracking. Real-time (or dynamic) tracking for network topology is a necessary building block for a number of traffic engineering solutions including the ones concerning network survivability. For instance, timely detection of failures in the network is necessary to initiate actions that will restore the failed components. In this chapter, we evaluate different approaches to design a fundamental building block for next generation network management systems – a *topology server*. Such a server would provide a real-time, accurate view of the IP network topology. Dynamic tracking of network topology is challenging for several reasons:

- **Growth and Change.** There is continual churn in the IP backbone, as new routers and interfaces to customers and peers are added, upgraded, or removed.
- **Facilities, Size and Scale.** A large IP backbone network today consists of several hundred routers, distributed over a large geographic area, with several thousand internal interfaces between routers, and tens of thousands of external interfaces to customers and peers.
- **Unreliability.** Router interface failures (permanent or intermittent) occur frequently in an IP network, caused by hardware and software problems. In addition, routine maintenance and provisioning events cause interfaces to go up and down.

- **No definitive external databases.** IP networks are designed to self-organize, through the interaction of routing protocols. Centralized databases and management systems, commonly used in circuit-switched networks to hold complex network state, are not required.

Current industry practices involve centralized topology and configuration databases and systems. As a rule, however, owing to the diverse and high rate of change in the network, these systems are inevitably incomplete and inconsistent, and often fall out of sync with the current network state. In addition, the systems tend to focus on associations at the network edge describing interface mappings to customers and peers. A natural approach, bypassing the problems of administrative databases, is instead to treat the network itself as the authoritative database, and monitor its topology directly.

The focus of out work lies on tracking the topology of a single Autonomous System (AS), or administrative domain of authority. Today, the Internet is divided into roughly 10,000 AS's, which represent major network service providers, content providers, and other institutions. Within an AS, routing is controlled by an interior gateway protocol, with OSPF (Open Shortest Path Forwarding) [102, 101] and IS-IS (Intermediate System to Intermediate System) [20] the prevailing choices for IP networks today [65]. We focus here on OSPF, and on constructing a *topology server*, which tracks the network-level topology controlled by OSPF.

The important constraints and requirements for the dynamic topology tracking are:

- **Independent and Incremental Deployment.** No change is needed to existing network routing or management protocols or systems.

- **Safety and Passivity.** There is no impact on packet routing, forwarding, and network reliability.

- **Accuracy and Timeliness.** The view of topology maintained in the topology server is identical to that in the router databases. Following an event that changes the topology, the topology server should converge to the new view near the time that the routers themselves converge to the new view.

- **Reliability and Robustness.** Updates to the topology should be reliably detected by the topology server. The design of the topology server should be very simple, so that it itself is reliable.

The basic design choice for dynamic topology tracking is whether to take a *control plane* or a more traditional *management plane* approach. A control plane approach is one that works by speaking enough of the OSPF protocol to take advantage of the messages (link state advertisements) that OSPF naturally floods throughout the network to ensure the routers arrive at a common view of the network topology. A management plane approach is one that collects network topology using messages called GETs or TRAPs in the Simple Network Management Protocol (SNMP). We implemented both approaches, and verified their correctness in a lab test-bed. It is difficult, however, to gain insight into the reliability and performance of the implementations from test-bed measurements. Instead, we measured aspects of the OSPF behavior of commercial routers, which we then applied to develop and parameterize a simulation model.

In the following sections, we describe advantages and disadvantages of the control plane and management plane approaches and report extensive simulation experiments over real and synthetic topologies carried out to evaluate the design approaches.

## 5.2 Related Work

Feldmann et al. [45] periodically dump router configuration files of routers. Similar tools periodically dump OSPF link state databases. This provides a static view of the topology. One can make it more dynamic by increasing the dumping frequency, but it is hard to go beyond certain limits because of the size of IP networks today. Another approach to topology discovery might be to monitor updates from systems that provision or reconfigure the network. The problem with this approach is that IP networks are essentially distributed and it is hard to design a topology server that sees all changes to the network's configuration. Moreover, the approach is more static than dynamic. With protocols like OSPF and IS-IS that use flooding for distributing topology information, one can use a protocol monitor on a LAN between two production routers, and snoop on the protocol messages describing topology messages. Sifting reliably through all the packets for the protocol messages describing the topology is a difficult task. Moreover, it is hard to obtain the initial view of the topology with this approach; the topology view builds up gradually as packets are captured. Lakshman et al. mention approaches to real-time discovery of topology in their work on the RATES System developed for MPLS traffic engineering [11]. But topology discovery is just one of the modules of their system and they do not go into details. Baccelli and Rajan discuss an OSPF monitoring architecture, tailored to a policy based networking framework [13].

Siamwalla et al. [130] and Govindan [55] discuss topology discovery methods that do not require cooperation from the network service provider, relying on a variety of probes, including pings and traceroutes. Such methods provide indications of interface up/down status and router connectivity. However, these methods do not deal directly with OSPF topology tracking, the topic of this paper. One serious disadvantage of any such method (common to ping-based network management schemes) is the inability to infer causes of missing pings. For example, traffic congestion will sometimes cause loss of OSPF connectivity and sometimes loss of pings, but the two are not directly correlated. In the midst of a topology change, pings directed to interfaces that happen to be up may be lost owing to transient routing loops.

## 5.3 Two Approaches to Dynamic Topology Tracking

We consider two basic approaches to the design of the topology server, a *management plane* approach , which involves speaking to the routers via SNMP, and a *control plane* approach , which involves some level of participation in OSPF.

### 5.3.1 The Management Plane Approach

SNMP provides the most natural and convenient mechanism to monitor OSPF topology from the management plane. The OSPF SNMP MIB [14] defines variables that indicate whether a given OSPF adjacency is up or down. These MIBS are maintained on the routers and values of the variables can be extracted using SNMP GET or GETNEXT methods. We use GET to acquire the initial view of the topology when the topology server starts.

SNMP also defines TRAPs that are triggered upon changes in OSPF status such as interface up/down or neighbor up/down. These TRAPs can be enabled on a router. While enabling a TRAP, one also needs to specify the IP address(es) where the TRAP needs to be sent. Once enabled, TRAPs are sent to these addresses when the associated event occurs. Thus, by registering with a router for TRAPs related to OSPF status changes, the topology server can keep track of the OSPF topology. Of course, the topology server has to register with all the routers in the network to construct the entire topology view. In general, TRAPs are not acknowledged, and so TRAP-based methods are not reliable. (Support for TRAP acknowledgment is beginning to become more widespread [131].)

**Implementing The Management Plane Approach**

The management plane based server uses the OSPF SNMP MIB [14] tables and variables defined therein as well as the TRAPs. The key MIB variables used are those defined in *ospfIfTable*, *ospfNbrTable* and *ospfAreaTable*. The *ospfIfTable* provides OSPF specific information on interfaces of a router while the *ospfNbrTable* describes neighbors of the router. The *ospfAreaTable* contains information about the OSPF areas to which the router is attached. The server polls these MIB variables using GET and GETNEXT methods for topology discovery at the start time and for periodic updates to the topology view after that. Among the TRAPs defined in OSPF MIBs, the ones we rely on are *ospfIfStateChange* and *ospfNbrStateChange* TRAPs. The *ospfIfStateChange* TRAP is generated by a router when the OSPF state of one of its interface progresses or regresses from a terminal state. Similarly, the *ospfNbrStateChange* TRAP is generated by a router when the OSPF state of its neighbor progresses or regresses from a terminal state. Together, these two TRAPs allows the topology server to keep track of when adjacencies between routers get formed or are teared down. This, in turn, allows the server to keep track of the OSPF topology.

The topology server executes three main processes: network discovery, TRAP configuration and handling, and polling. Network discovery process is used for discovering routers in the network. The process extracts local OSPF variable set from the router's OSPF MIB: the administrative state, the interfaces, the attached areas and the OSPF adjacencies among others. It maintains two sets of routers. Set $\Omega$ maintains "yet-to-be-processed" routers, whereas set $\Theta$ maintains those set of routers which are already processed. The discovery process starts with one or more "seed" routers in $\Omega$. While $\Omega$ is not empty, the process removes a router from $\Omega$, processes it, adds it to $\Theta$, and adds all its adjacent routers that have not already been discovered to set $\Omega$.

The TRAP configuration and handling process enables the TRAPs mentioned above on the router as they are discovered. At the time of enabling, the process adds the topology server as a recipient of these TRAPs. The process also handles the TRAPs sent by routers and updates the topology view.

The polling process implements a straightforward polling schedule. The process cycles through the set $\Theta$ at a configurable rate, and checks if these routers have got any new neighbors. If the new routers are discovered, they are first added to $\Omega$. These routers are then processed and moved to the set $\Theta$. This way the topology server gets to know about new routers as and when they are added to the network. The server then enables traps on these newly discovered routers.

### 5.3.2 The Control Plane Approach

The control plane approach for the design of topology server will comprise of two principal functional components [126]. These are an LSA Reflector (LSAR) and an LSA Aggregator (LSAG). The LSAR establishes adjacencies with routers and collects LSAs which are then passed along to one or more LSAGs over reliable TCP connections. Each LSAG implements a topology server, in the sense that it synthesizes a global view of the topology from the received LSAs. By design, an LSAR is close to the network, and so must be very simple in order to achieve a high level of reliability. Moreover, we may require more than one LSARs to cover all the areas if areas are geographically widespread. The LSAG handles all complex logic: storing and managing the updates, filling in the

OSPF topology data-model [127], and providing APIs to applications that require topology views.

Separating the LSAR and LSAG functions provides a degree of fault isolation; each function can be simplified and replicated independently to increase overall reliability. Another benefit is that the LSAG may subscribe to a subset of the LSAs, for example, just the Router and Network LSAs for a given OSPF area. Moreover, the LSAG, having to support applications, may require more complex processing and more frequent upgrades. Separating the LSAG from the LSAR allows us to bring it up and down without disturbing the LSARs.

**Passivity and Safety of LSAR**

One of the main design goals for the LSAR is that it must be *passive*. Though the LSAR speaks OSPF it is not a real router, and it must not be involved in packet forwarding. No other router in the network should send data packets to the LSAR to be forwarded elsewhere. A natural line of defense against having the LSAR participate in normal forwarding is to use router configuration measures: assign effectively infinite OSPF weights to the links to the LSARs, and install on neighbor routers strict access control lists and route filters. A stronger line of defense is to exploit standardized features of the OSPF protocol that permit links to the LSAR to carry LSAs from the routers, but do no permit neighbors to use the link to carry data traffic. We accomplish this by keeping the LSAR-router adjacency in an intermediate state such that the two ends start the database synchronization but never finish it. In order to achieve this, the LSAR originates an LSA $L$ and informs the router that it has this LSA during the synchronization process but never actually sends it out to the router. This makes sure that the database is never synchronized and hence the adjacency is never fully established from the router's perspective. As a result of this, the adjacent router never gets to advertise a link to the LSAR in its Router LSA although it sends all the LSAs it receives to the LSAR as a part of the flooding procedure. A side benefit of this approach is that instability in the LSAR does not impact other routers in the network.

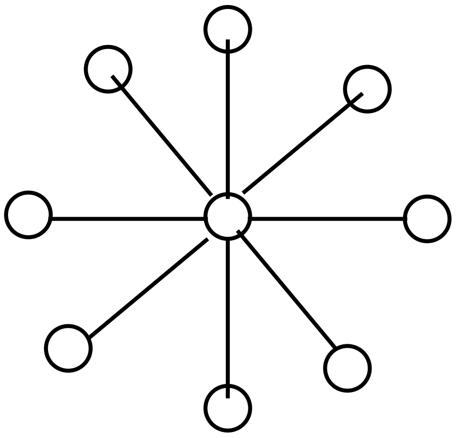
## 5.4 Evaluation of Different Approaches

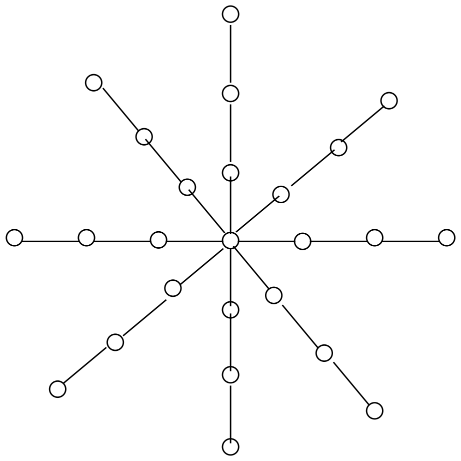
In this section, we present a simulation-based exploration and comparison of the performance of the control and management plane approaches for acquiring topology information. Though we implemented the approaches and checked their correctness in a lab testbed, the testbed did not offer the flexibility needed to evaluate the approaches. Instead, we used the testbed to gather some measurements, which we incorporated in the simulation model. A realistic simulation model allows us to get a quantitative idea of things to be expected in reality.

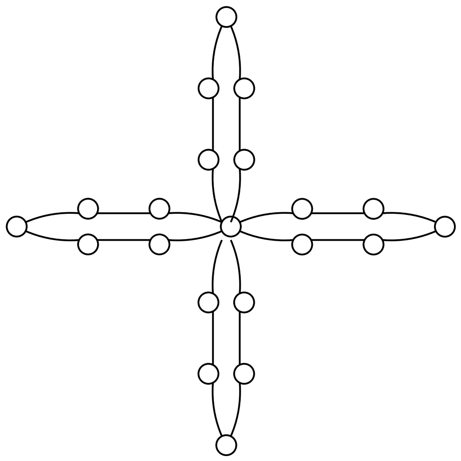
### 5.4.1 Simulation Methodology

We used an enhanced version of NS2 simulator [107] for our experiments. The enhancements were done to implement all the essential features of OSPF and SNMP for the purpose of our experiments. The experiments were focused on understanding what factors affect the speed and reliability of topology state acquisition using the control plane and management plane approaches. The parameter space in this regard can be thought of as consisting of three axes which we describe in detail below. Each of these will play a significant role in determining the performance of a topology server in tracking the dynamic changes to the topology in the network.

1. **Topology:** The size and connectivity of the network considerably influence the propagation of topology state via the two methods. The connectivity in the network


(a) Star with radius 1


(b) Star with radius 3


(c) Flower

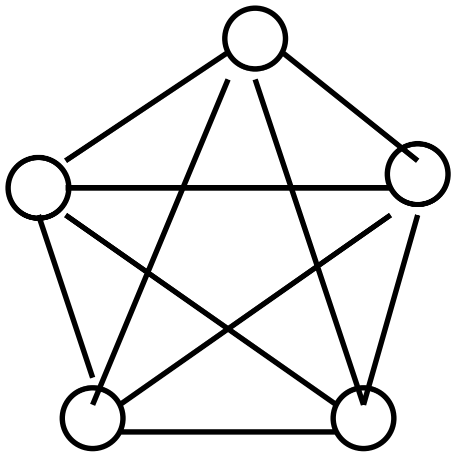

(d) Mesh

Figure 5.1: Simulation Topologies

| **Configurable delays: Standard** | |
|---|---|
| RxmtInterval | The number of seconds between LSA retransmissions for adjacencies belonging to an interface. Usually 5 seconds. |
| HelloInterval | The length of time between the HELLO Packets that the router sends on the interface. Usually 10 seconds. |
| RouterDeadInterval | After ceasing to hear a router's HELLO Packets, the number of seconds before its neighbors declare the router down. Usually 4 times HelloInterval |
| **Configurable delays: Vendor-specific** | |
| Pacing delay | The minimum delay between two linkstate update packets sent down an interface. Observed to be 33 ms. Not always configurable. |
| spfDelay | The delay between receiving the first topology change since last SPF calculation and doing the SPF calculation. Used to avoid frequent SPF calculation. Usually 5 seconds. |
| spfHoldTime | The minimum delay between two successive SPF calculations. Usually 10 seconds. |

Table 5.1: Configurable OSPF and SNMP delays

topology will determine the speed with which LSAs spread throughout the network. The availability of alternate paths and the number of hops from the source of the topology change to a router of interest (or the topology server) will determine if the LSA will reach the router or topology server without requiring a retransmission.

With SNMP TRAPs, the larger the hop count from the point where the change takes place to the topology server, the smaller the probability of the TRAP making its way to the topology server. Further, the network topology may be such that there are a relatively small number of key links and routers that provide interconnections between portions of the network (e.g., long-haul links between cities in the backbone for a service provider). These key links and routers may be part of most of the routes. The usefulness of SNMP TRAPs will depend on these links and routers, and

| **Fixed delays: Standard** | |
|---|---|
| LSRefreshTime | The maximum time between distinct originations of any particular LSA. Set to 30 minutes. |
| MinLSInterval | The minimum time between distinct originations of any particular LSA. Set to 5 seconds. |
| MinLSArrival | For any particular LSA, the minimum time that must elapse between reception of new LSA instances during flooding. LSA instances received at higher frequencies are discarded. Set to 1 second. |
| **Fixed delays: Device-specific** | |
| Route Install delay | The delay between SPF calculation and update of forwarding tables. Observed to be 0.2 seconds. |
| TRAP Generation delay | The delay between occurance of an event and generation of an SNMP TRAP for it. Observed to be around 2.2 seconds. |
| LSA Generation delay | The delay that takes place before generation of an LSA after all the protocol specific requirements for the LSA generation have been met. Observed to be around 0.5 seconds. |
| LSA Processing delay | The time it takes to process an LSA. This includes time to process the linkstate update packet in the operating system before forwarding the LSA to the OSPF process. Observed to be less than 1 ms. |
| SPF Calculation delay | Time it takes to do an SPF calculation. Observed to be ($0.00000247x^2 + 0.000978$) seconds where $x$ is the number of nodes in the topology. |

Table 5.2: Fixed OSPF and SNMP delays

it is important that they do not fail or do not become overly congested. The topology of the network will also determine the duration of routing transients as changes take place, and hence the reachability of SNMP TRAPs to the topology server.

In our simulations, we explored the influence of the network topology by first choosing simple regular topologies (Figure 5.1) that allowed us to understand the impact of hop count and multiple paths on the propagation of LSAs and TRAPs. These simple topologies also helped us in assessing the impact of non-topological factors such as the configuration parameters of OSPF. The number of nodes in these topologies was kept sufficiently high to give us enough sample points. Subsequently, we performed simulations on a much more complex OSPF topology similar to AT&T's nationwide backbone.

2. **OSPF and SNMP delays:** Various delays that will affect the topology state propagation via OSPF LSAs and TRAPs can be classified as configurable delays and fixed delays and are shown in Tables 5.1 and 5.2. In both categories, some delays are standard (specified in the protocol standards document [102]) while others are an artifact of the protocol implementation by vendors. Settings of these delays will affect not only the dynamics of the dissemination of information via LSAs but also the functioning of the OSPF protocol itself. In order to find the main vendor/device specific delays affecting the OSPF and SNMP operation, we performed extensive measurements on Cisco 3600, 7200, 7500, and 12000 series routers. Though these router models vary tremendously in hardware architecture and packet forwarding capability, each was configured with the same operating system: Cisco IOS version 12.0(7), and had a similar central processor (mostly MIPS R5000, running at 200 Mhz). In terms of the delays, we found that the different models behaved similarly. Details

of the measurement methodology and results can be found in [128]. We incorporated these delays in our simulation model and used their observed/default values (randomized uniformly in $\pm 10\%$ range) in our experiments.

3. **Network Characteristics:** The load and characteristics of traffic on the network may impact the ability of the topology server to acquire the current network topology information. LSAs and SNMP TRAPs can be lost as a direct result of link congestion and buffer overflows. The secondary impact of network congestion on OSPF is the potential loss of adjacencies on links due to the loss of successive HELLO packets. In our simulations, we model the impact of network traffic as a loss probability for packets in the network. This loss model seems simplistic in that successive packet losses are assumed to be independent, as are losses on distinct links. However, such a model is reasonable for our purposes, and is not too far from reality since protocol implementations for commercial routers cause successive packets containing LSAs to be paced using time delays that are large enough to prevent strong correlations in packet loss events for OSPF LSAs.

In our experiments, we took the approach of fixing the location of a topology change and observing how the change is intimated to each node in the topology via two approaches. Essentially, we observe the arrival of topology change information for each possible location of the topology server. The process of change intimation can be divided in to generation of the *herald* message (LSA or TRAP) and the propagation of the herald to the topology server. The propagation of the herald follows the similar pattern for different kind of changes except that the the time at which the herald is generated might be different. For example, we have observed an implementation dependent delays of 0.5 and 2.2 seconds in the generation of LSA and TRAP respectively. Here, we focus on understanding the factors affecting the herald propagation once it is generated. The heralds in focus were generated for an *adjacency up* event on an interface on a router. In order to evaluate management plane approach, we made the source of the topology change send the corresponding TRAP to all the nodes (potential location for topology server) in the topology. As noted earlier, the chance that heralds are dropped at any given hop is a parameter of the model, which we vary from run to run.

### 5.4.2 Single Path Characteristics

We start with the simplest topology, the star of radius 1 (Figure 5.1(a)), and make a few observations about the herald propagation. Suppose an *adjacency up* event occurs at the center of the star. In the management plane approach, a TRAP is sent to the topology server. The probability that the TRAP arrives is identical to the packet drop rate, because the topology server is one hop away (unless attached to the center). The time it takes for the trap to arrive is just the propagation, queuing and pacing delays.

Consider the control plane approach. An LSA is flooded to the other nodes (termed *satellites*). Following the reliable flooding protocol, the center will make as many tries as needed to get the LSA to each satellite. The number of LSA retransmissions to the topology server is simply a geometric random variable with parameter equal to the probability of packet loss. If an LSA is lost, the retransmission will have to wait for the retransmission timer to expire. Otherwise, the LSA will reach the destination after the propagation, queuing and pacing delays.

In Figure 5.2 we plot the ratio of satellites that have received this LSA by a certain time for the star topology with radius 1 and 100 satellites. Data is provided for the drop rates $p = 5\%, 15\%, \text{and} 25\%$. Note that for drop rates $5\%$ and $15\%$, $1 - p$ of the routers get the LSA

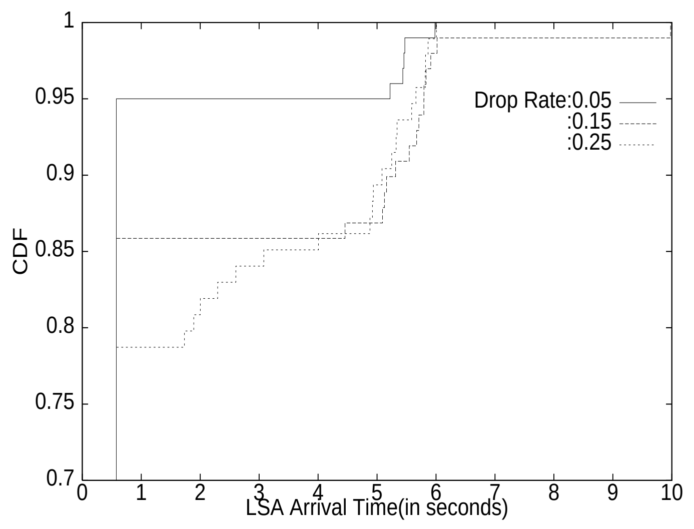


Figure 5.2: Star topology with radius 1 and 100 satellites: Arrival of topology change via LSAs at satellites as drop rate increases.

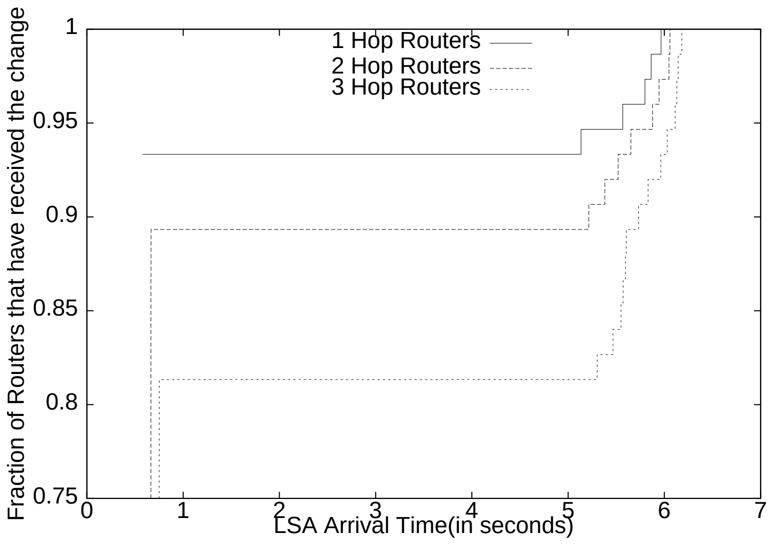
on the first transmission, and the remaining get the LSA on the expiry of the retransmission interval (roughly $5$ seconds). However, for the drop rate of $25\%$, some of the satellites that missed the LSA on the first transmission, get the change intimation earlier than the time the LSA would have been retransmitted. There are many factors responsible for this behavior. Let us consider one. In steady state with such a high loss probability, a number of adjacencies would be established and torn down as a result of HELLO packet losses. Consider an adjacency to a satellite that happens to be down at the time of the generation of the LSA in question. Hence, the LSA won't reach the satellite until it establishes full adjacency with the central router.It is as if the attempted LSA transmissions that all occur immediately after generation in the low loss scenarios are smeared out over a longer time period for $25\%$ loss.

Next consider a star topology with radius $> 1$, say 3, where the satellites are banded at distances 1, 2, 3. We can then examine the characteristics of herald propagation over multihop paths, with still a single path from the source of change to each other node. The herald transmission time is the sum of the single hop transmission times across the path. TRAPs follow the same paths as LSAs; however they are not retransmitted and their losses result in some network nodes (potentially a substantial number, including the topology server(s)) missing the change indication.

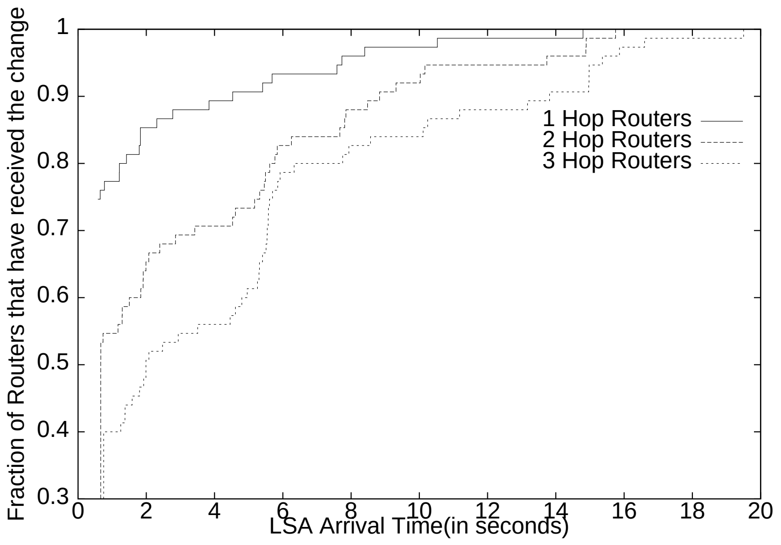
Figures 5.3 and 5.4 show the propagation of a topology change occurring at the center of a radius 3 star with $75$ rays (and hence $75$ satellite each at distance 1,2 and 3)(Figure 5.1(b)). Figures 5.3 shows the arrival of the change via LSAs for satellites 1, 2 and 3 hops away from the center for drop rates of 5% and 25%. It is easy to see that $(1-p)^n$ fraction of the satellites at distance $n$ will get the LSA on the first transmission, The fraction of satellites that will get the LSA or the SNMP TRAP without loss is $1/3 \times ((1-p)+(1-p)^2+(1-p)^3)$. This can be verified from Figure 5.4 which shows the arrival of the topology change via LSAs and TRAPs for all nodes.

### 5.4.3 Multi-path Characteristics

In the case of the star topology, each node has only one path from the center which is taken by both the TRAPs and the LSAs and hence the fraction of nodes receiving the TRAP and those receiving the LSA without any retransmission was the same. However, it is to be expected that if there are multiple paths for the LSA to reach a node, as with the flower and mesh topologies, the packet loss probability has less of an impact on the arrival of a topology change via LSAs. This effect is evident from the Figures 5.5, 5.6 and 5.7 for the topologies with alternate paths. In the flower topology, each node can receive LSA from the center (where the change takes place) on two separate paths. As the packet drop rate increases from 5% to 25% (Figure 5.6), the fraction of nodes receiving the LSA

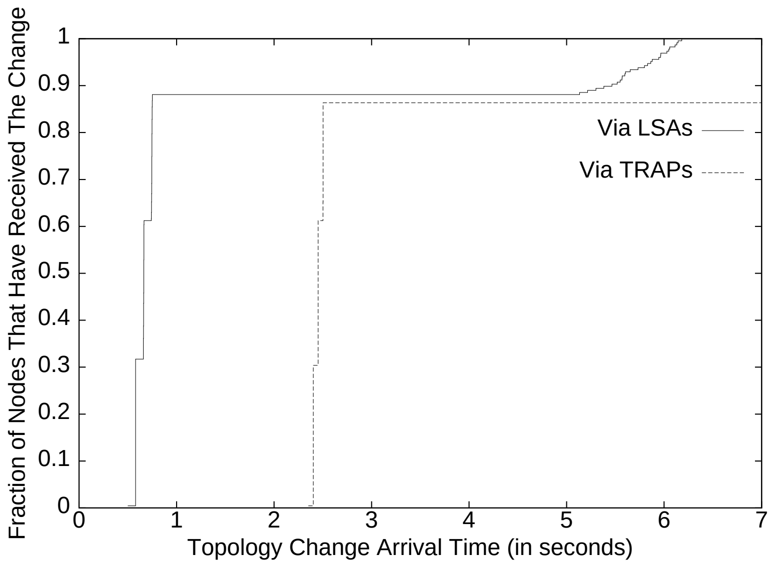


(a) Drop Probability: 5%

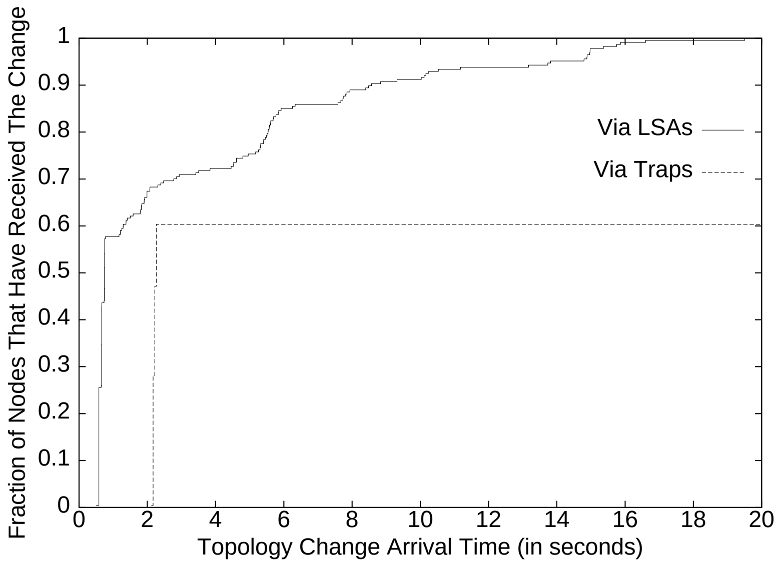


(b) Drop Probability: 25%

Figure 5.3: Star topology with radius 3: Arrival of topology change via LSAs as hop count increases.

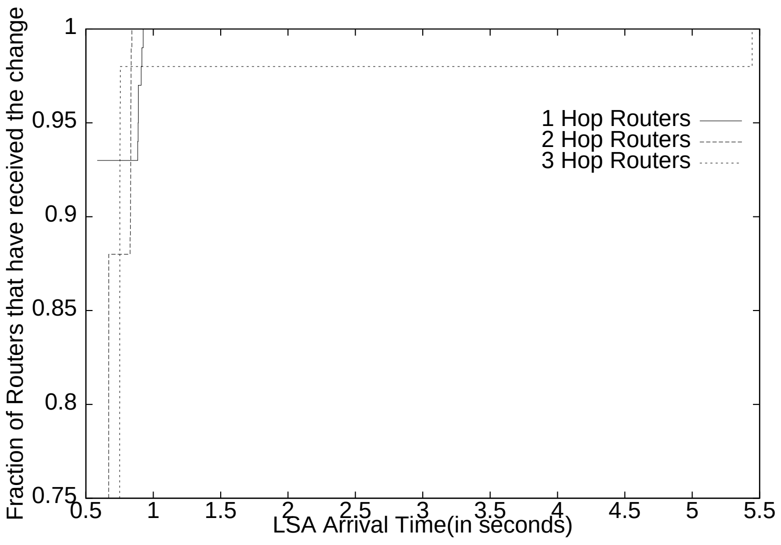


(a) Drop Probability: 5%

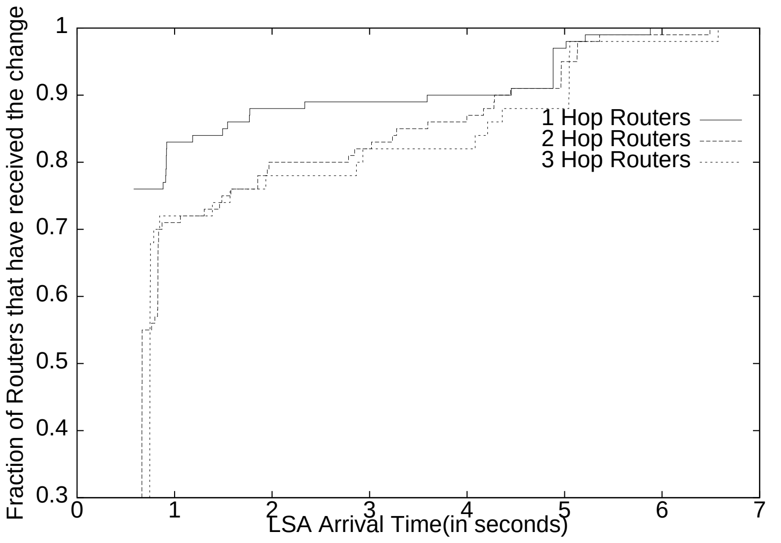


(b) Drop Probability: 25%

Figure 5.4: Star topology with radius 3: Arrival of topology change via LSAs and via TRAPs.

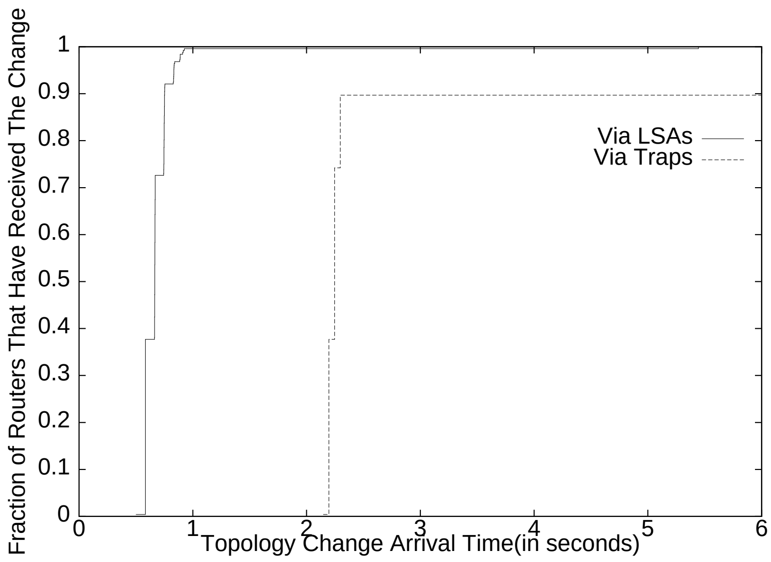


(a) Drop Probability: 5%

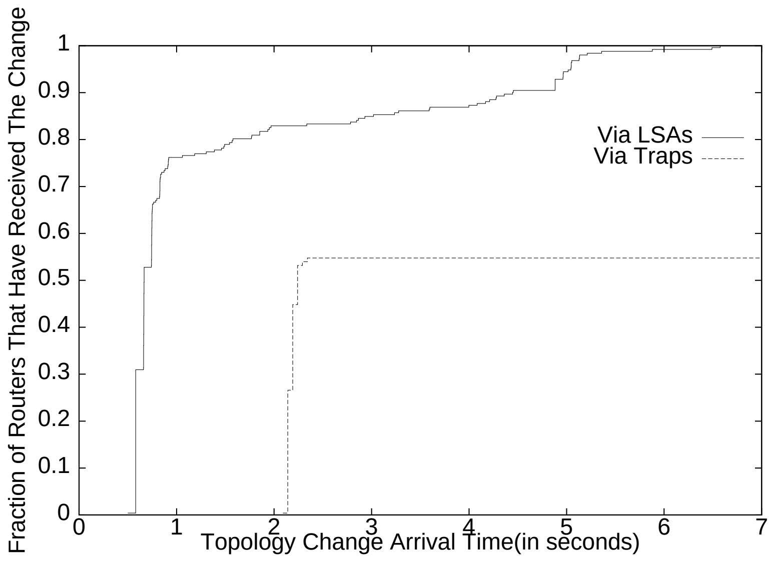


(b) Drop Probability: 25%

Figure 5.5: A Flower with 50 petals and 3 hops in each petal: Arrival of a topology change via LSAs as hop count increases

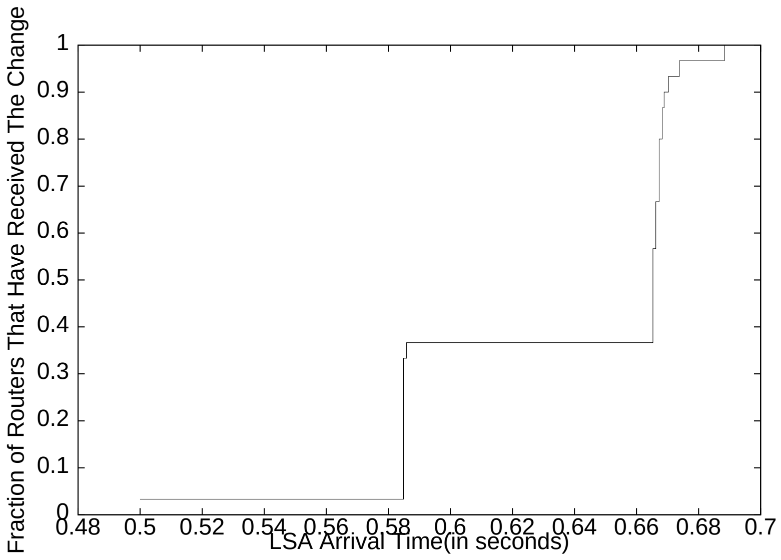


(a) Drop Probability: 5%

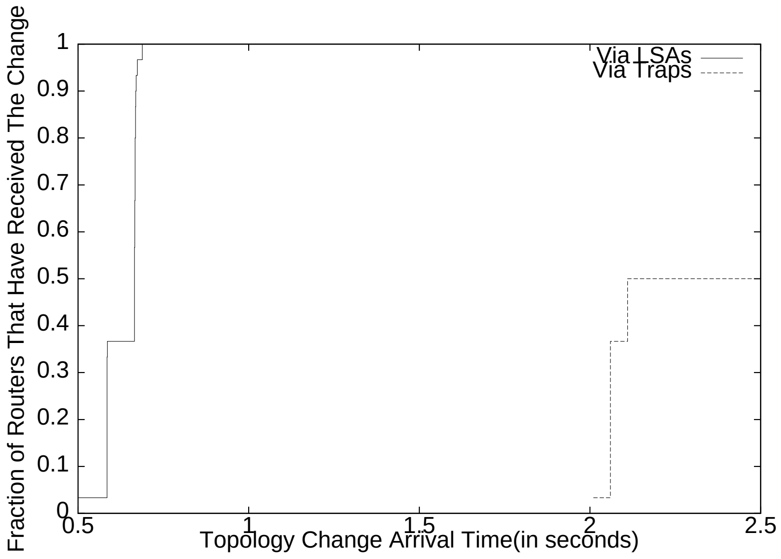


(b) Drop Probability: 25%

Figure 5.6: A Flower with 50 petals and 3 hops in each petal: Arrival of topology change via LSAs and via TRAP.



(a) Arrival of change via LSAs. Drop Probability: 50%



(b) Arrival of change via LSAs and via TRAP. Drop Probability: 50%

Figure 5.7: Results for a mesh of 30 routers

without retransmission become much larger than the corresponding fraction receiving the SNMP TRAP. For the mesh topology (Figure 5.7), since there are a large number of paths available for LSAs to arrive on, even with a huge drop rate of 50%, all nodes get the LSA without retransmission.

The comparison of Figures 5.3 and 5.5 for star and flower topologies respectively provide interesting insights into the effect of multiple paths and hop count of the paths on the LSA arrival. Note that for the flower topology, the fraction of nodes getting the LSA without retransmission are much higher than those for star topology. The benefit is greatest for nodes 3 hops away from the center. For these nodes, there are two independent 3 hop paths for the LSA to take from the center (where the change takes place) in the flower topology as opposed to a single path in the star topology. For nodes at distance 1 or 2 from the center, the longer path from the center involves 4 and 5 hops respectively, and hence becomes less useful as the packet drop rate increases.

The LSA retransmission (roughly 5 second) timer value is the main factor in determining the arrival time of the LSA to the topology server when LSAs can be dropped due to congestion. Figure 5.7(a) provides an interesting case in contrast, where because of rich connectivity among the nodes of the mesh there is very little need for an LSA to be retransmitted, even for very high loss rates (in our experiments). In this case, it can be seen that only 40% of the nodes obtain the LSA directly from the source of change. However this population of 40% in turn floods the LSA to all the other nodes in the network, except the source. Hence, even though the number of duplicate LSAs received is high, all the nodes get the LSA indicating the change within the delay involved in LSAs traversing two hops without loss.

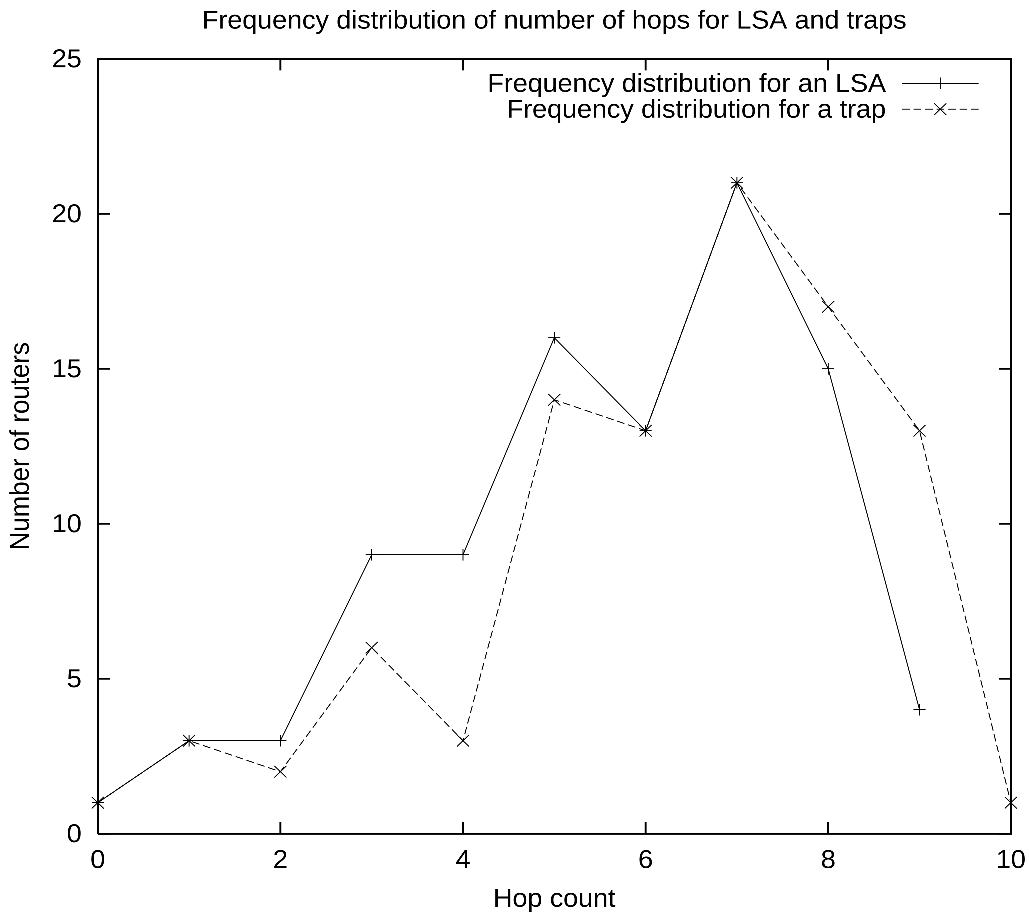


Figure 5.8: Simulated OSPF backbone: Hop counts in paths taken by LSA and TRAP from the source of change to all the routers. Avg path length: 5.67(for LSAs), 6.33 (for TRAPs)

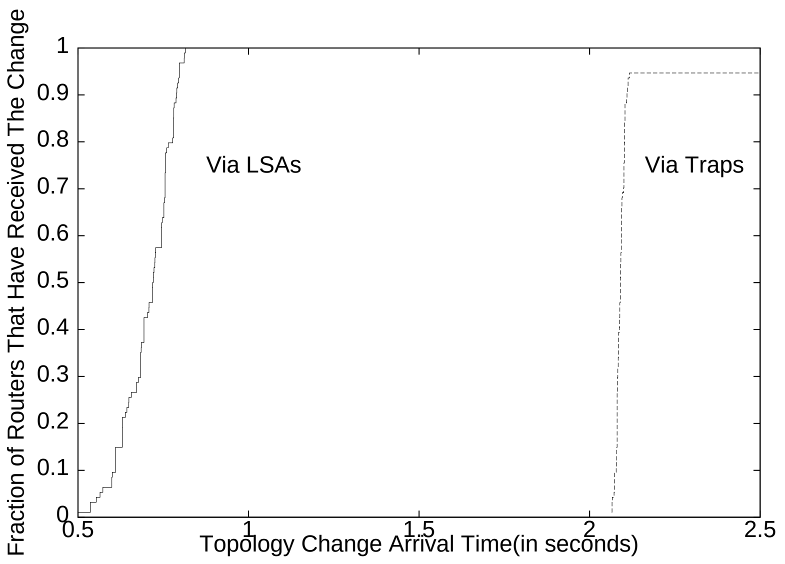
### 5.4.4 Simulations with Real Topologies

After having understood the factors affecting the propagation of heralds in simple regular topologies, we now present results for simulations done using an OSPF backbone topology similar to an AT&T operational IP network. This topology has about 100 nodes and is richly connected. In these experiments, we choose an interface on a node in one corner of the topology to fail. When the interface comes up, the adjacency is re-established with the neighbor. We track how the heralds propagate this event throughout the network. Figure 5.8 shows the hop counts in the shortest paths from this node to all the nodes in topology when all links are functional. The shortest path for the LSA propagation is calculated based on hop counts and the path for SNMP TRAPs is calculated based on minimum cost. Since the interface that goes down is in one corner of the topology and the average hop count for both the LSA and TRAP paths from this interface is around 6, we expect to quantify the worst case delays in propagation of a topology change via both methods.

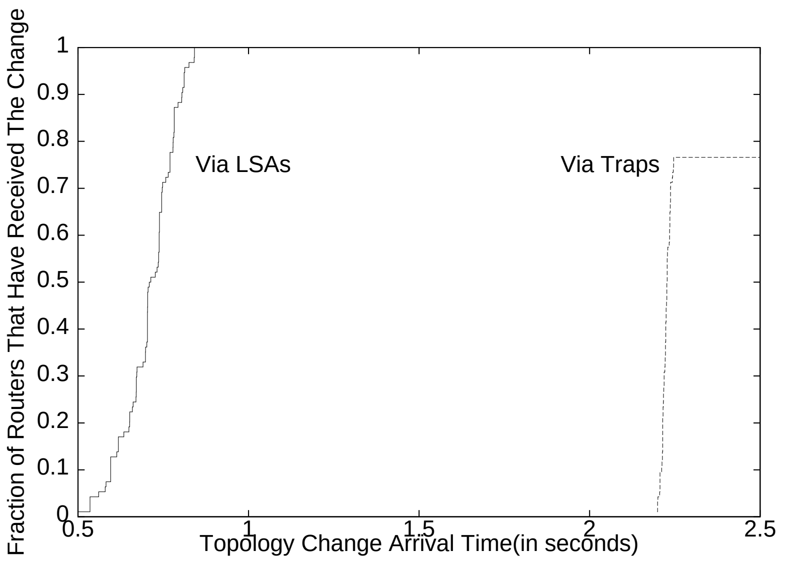
Figures 5.9 and 5.10 show the simulation results for this topology. Since the average hop count is quite high, the fraction of nodes receiving the TRAP drops drastically as the packet loss rate in the network increases. Note that even with drop rates as high as 10%, via LSAs the change propagates throughout the network without requiring a single retransmission. Even in the drastic case of a 25% drop rate, very few nodes required LSA retransmission to get the change.

### 5.4.5 Discussion: Reliability and Robustness

The most significant result we observe from the simulations is the reliability of the LSA approach even in the face of drastic network conditions, such as packet loss in excess of 20%. If there are multiple possible paths from the source of the topology change, generating an LSA, to a destination, the effect of the packet drop probability reduces in proportion to the number of such alternate paths. If the LSA gets dropped along one path, it might still arrive via some other path. In the event of LSAs being dropped along all possible paths, one can still recover because of the retransmission mechanisms built into the flooding of LSAs, causing the LSA to be retransmitted after every *RxmtInterval* until successful.

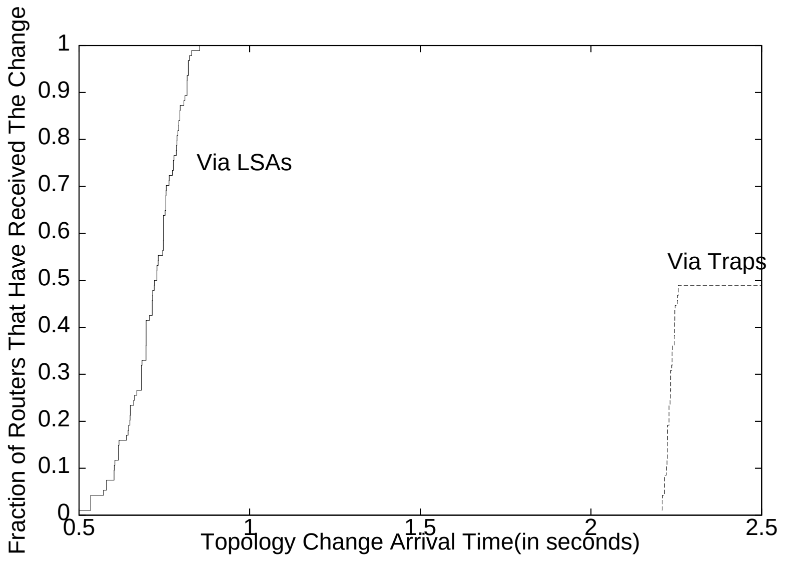


(a) Drop Probability: 1%

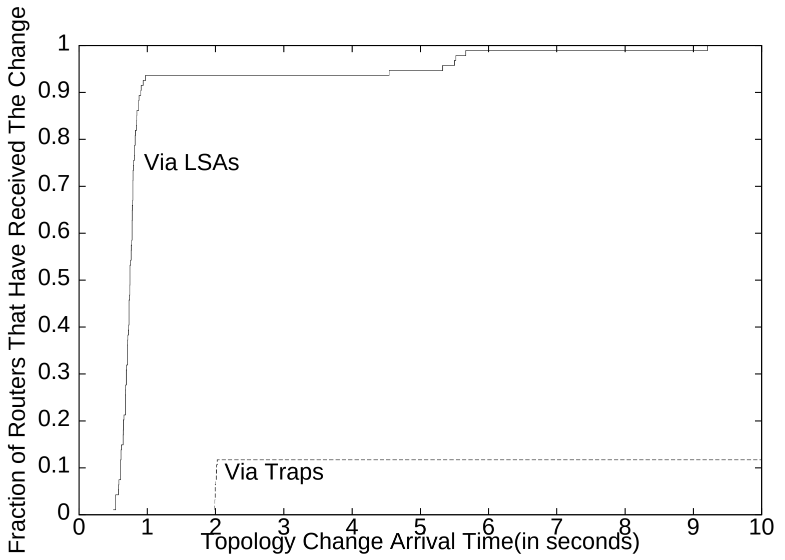


(b) Drop Probability: 5%

Figure 5.9: Simulated OSPF backbone: Arrival of topology change via LSAs and via TRAP. (Drop Probability 1% and 5%)



(a) Drop Probability: 10%



(b) Drop Probability: 25%

Figure 5.10: Simulated OSPF backbone: Arrival of topology change via LSAs and via TRAP. (Drop Probability 10% and 25%)

On the other hand, traps are not a reliable means for communicating topology changes. A trap if lost will not be retransmitted. SNMP Version 2 provides for a reliable version of trap, called “Inform”, which has the property that it will be retransmitted a number of times until acknowledged by the intended recipient. Informs may take care of the problem of lost traps in some cases. Traps may be lost if there is congestion in the network or because the path that the trap traverses is broken. Informs could solve the problem of lost traps if the loss is primarily because of congestion in the network. In the other scenario, Informs are likely to be less successful. A broken path, and especially a routing loop, may be the result of a transient phase after a sequence of topology changes before the changes have been incorporated in routing tables throughout the network. Because of hold down timers on LSA generation and the SPF calculation, this transient phase and hence the routing loop or the break in the path might last for several seconds. All the Informs sent during this duration will be lost. The Inform that finally arrives at the destination might also become ’stale’ by the time the transient phase is complete. The ’stale’ Informs might result in potentially long periods of erroneous view of the network topology at the topology server. For example, consider the scenario where an interface goes down and come back up after some time. The traps that are generated for these events could eventually make their way successfully to the topology server. However, a *neighbor down* trap generated by the neighbor when the interface went down may not reach the topology server until after the *interface up* trap, because the message has to wait for the connectivity to be re-established anyway. Since, the topology server can not re-order the trap received from two different sources, the stale *neighbor down* trap could create an error in the view of the topology until the corresponding *neighbor up* trap arrives which will have to wait for adjacency to be reestablished between the neighbors (potentially a long period before this happens.) On the one hand, it can be argued that the link is “logically” down until the adjacency is reestablished and hence the “stale” *neighbor down* trap is not really stale. However, this argument raises the question on the usefulness of all the non-OSPF traps in this regard since these traps indicate only the “physical” state of the network and not the “logical” state. This argument really leads us to view the non-OSPF traps (or Informs) only as an “early warning” of things to come. In this respect, the traps can be seen as playing a useful, but complementary role to other information dissemination methods such as OSPF flooding, in many applications. An additional problem with traps is the problem of correlating together different traps generated for the same topology event. In the previous paragraph, we gave an example where a single interface failure results in generation of an *interface down* trap and a *neighbor down* trap. Additional traps might be generated for this event by lower level protocols and also by specialized *probe* devices placed in the network for tracking the state of all or part of the network. Further, the events indicated by traps, though different, might be originating due to the same cause. e.g., a *router down* event might be the cause of a number of *interface down* and *neighbor down* events. Thus, it is possible to have many traps generated for a single topology event. Now the problem becomes one of having too many traps being generated. In the presence of “noise” caused by lost and stale traps, it becomes a hard problem to differentiate the fundamental causes of each of the traps, whether they were caused by same or different topology events.

In the previous paragraphs, we proposed that traps be used as carriers of early warning information of things to come. Indeed, the same proposition could be applied to the arrival of LSAs as well. The arrival of an LSA at a router precedes the corresponding SPF calculation, possibly by several seconds. On a practical, logical level, the change does not “arrive” at a router unless that router’s routing table reflects the change. In this respect, the arrival of a network state change at the topology server via an LSA is also “early warning” information in the sense that most probably the change has not yet been reflected in routing tables of all the routers in the network. However, there is a significant difference in the meaning of the “ahead of time” information conveyed by trap and that by LSA. Depending on the type of trap, the event notified may just be a “physical” event which might not have any “logical” consequence. For example, an *interface up* event will be logically inconsequential if it is immediately followed by the *interface down* event. On the other hand, event information carried in the LSA is about a “logical” event that has more significance than a mere “physical” event. However, it should be noted that even a “logical” event will be inconsequential if replaced by another “logical” event before the first gets a chance to affect the routing tables. However, chances for this happening with OSPF are slim because logical events are generated spaced sufficiently apart in time both architecturally and in the implementations.

### 5.4.6 Discussion: Speed of Topology Change Intimation

The simulations brought out very clearly the reliability and robustness of the control plane approach even in face of drastic network conditions. Apart from the reliability and robustness, the speed with which the topology change information is conveyed to the topology server is also important. In this section, we examine how the control plane and management plane approaches fare in this regard. Doing this comparison is tricky because of the different delays involved with the two methods. One way to attempt this is to evaluate each approach with respect to three different factors affecting the total delays. These factors are: 1) the time taken to generate the herald message (LSA or TRAP) at the source of the change, 2) the time taken by the herald to travel across one hop in the network and 3) the path taken by the herald from the source to the destination. In some ways, the network conditions (e.g. the topology and the prevalent packet loss rates) will also affect the speed. However, the issue there is primarily that of robustness.

The generation of a router LSA or a TRAP is affected by the event in question, the flow control mechanisms in place (like the *MinLSInterval* and TRAP throttling [14]) and any implementation dependent delays in LSA/TRAP generation. An *interface down* event will be a sufficient cause to generate a new instance of the LSA while an *interface up* event will have to wait for the adjacency to be established with the neighbor. Flow control mechanisms will put a limit on how frequently the LSAs/TRAPs can be generated. The *MinLSInterval* determines how frequently a new instance of a router LSA can be generated and is a fixed architectural constant having a value of 5 seconds. Similarly, the TRAP throttling mechanism [14] allows only up to a certain number of TRAPs to be generated within a given time window. The recommended values for OSPF TRAPs are a maximum of 7 TRAPs within a window of 10 seconds. Further the throttling is not applied to certain kinds of TRAPs[14]. Finally, there are implementation dependent delays which will have a significant impact. In our experiments with Cisco routers, we observed generation of LSA and TRAP to be delayed by roughly 0.5 seconds and 2.2 seconds respectively after the event. So, with implementation dependent delays as observed and assuming that flow control mechanisms are not a limitation, an *interface down* event will cause a new LSA and a TRAP to be generated roughly after 0.5 seconds and 2.2 seconds respectively. In the similar conditions, for an *interface up* event, the TRAP will still be generated after 2.2 seconds but the LSA will not be generated until 0.5 seconds after the adjacency is established on that interface. Adjacency establishment requires the neighbor to get a HELLO from the newly up interface which will take around 10 seconds. After this, database synchronization has to take place, which will take a time proportional to the amount of information that needs to be exchanged. Thus, adjacency establishment can take well over 10 seconds. Thus, in the considered circumstances, the TRAP approach is faster for *interface up* event and slower for *interface down* event. Often, LSA or TRAP generation times are the major components of the total delay before the change reaches the recipient.

Assuming that packet forwarding in routers takes minimal time, the TRAP will undergo propagation and queuing delays on each hop to the destination. On the other hand, the LSA will undergo not only queuing and propagation delays but also some processing and pacing delays. In our measurements on Cisco routers, we found LSA processing delays to be in sub-millisecond range, however pacing delays can be substantial. Cisco routers have a built-in pacing mechanism that restricts LSAs to be flooded no faster than one link state Update packet (equal to 1500 bytes Ethernet frame size) every 33 ms. Thus, an LSA will undergo an indefinite pacing delay before being forwarded to the next hop.

Finally, the inherent flooding mechanism of the LSA propagation makes sure that the LSA indicating change takes the shortest possible path in terms of delay from the source to the destination if there is no loss. In comparison, for the TRAP, each router on the path will decide the next hop for the TRAP based on what it thinks is the least cost path from itself to the ultimate destination of the TRAP. The hop count or delay may or may not constitute the metric determining the cost of a path. Thus, the minimum cost path might not be the one with minimum delay or minimum number of hops. Even if hop count or delay constitutes the metric for determining cost, because of routing transients, the actual path taken by the TRAP from source to the destination might not be the globally least cost path. Thus, in terms of the path taken from the source to the destination, the LSA approach fares better than the trap approach.

## 5.5 Summary

We have attempted to address the problem of tracking the intra-domain routing topology in real-time. Such topology tracking is essential in ensuring survivability and building any kind of value-added services to the network. In particular, we identified various design choices, and evaluated them by comparing advantages and disadvantages of these choices.

We described two main approaches for building a topology server: control plane and management plane. The control plane approach requires listening to the OSPF messages (LSAs), each of which describes a part of the topology and is flooded reliably. The main issue with this approach was how to make it passive and non-intrusive. On the other hand, the management plane approach relies on SNMP and involves using TRAPs for notification of topology changes and polling. We provided a detailed description of our implementation of a topology server based on each of the approaches. We also compared these two approaches in terms of operational issues, reliability, and the ability to provide early warnings.

We performed simulations to provide some quantitative idea as to how a topology change propagates via the control plane and the management plane approaches. The results clearly demonstrated the superiority of the control plane approach over the management plane approach in terms of reliability and robustness. The results brought out three main reasons for this. First, LSAs are flooded, so there are multiple paths over which the information regarding a topology change traverses to the topology server whereas TRAPs have only one path to get to the topology server. Second, LSAs are flooded hop-by-hop and hence do not depend on routing state whereas TRAPs do. Third, LSAs are flooded reliably whereas TRAPs are not reliable.

At first blush, it might seem that with the tremendous investment in SNMP based network management systems today, the obvious and natural approach to building an OSPF topology server is the management plane approach. However, as pointed out in the paper, significant operational and reliability issues loom large with this approach. The management plane approach can at best compliment the control plane approach because of its ability to provide early warning. Some applications might leverage this early warnings to do some useful precomputation.

The main task for the future is extending the topology server to track status “below” and “above” intra-domain routing like tracking physical level assets and inter-domain reachability information.

# CHAPTER 6

# FAST RECOVERY FROM NETWORK FAILURES IN OPTICAL WDM NETWORKS

## 6.1 Introduction

Fast recovery from network failures is a critical user requirements. In the previous chapter, we examined one aspect of the problem - dynamic topology tracking so as to enable fast detection (and hence fast correction) of network failures by the administrators. In the current and the next chapter, we examine how to achieve rapid re-routing of the traffic affected by the failure while the failure has not yet been corrected. Since the interruption of the Internet service can have serious consequences, it is important that the network survivability mechanisms detect the failures and re-route the affected traffic before the users have a chance to notice the failure. Modern communication networks are multi-layer in nature with IP and optical WDM layers rapidly emerging as the consensus choice for the top and bottom of the stack. Each layer in the network hierarchy provides several choices of survivability mechanisms designed to protect the network against failures occurring at that layer. For example, the IP layer survivability mechanisms- whether in-built in the routing protocols like OSPF or explicitly deployed in MPLS based architectures - provide protection against IP layer failures such as the failure of a router or a router interface. Similarly, the optical WDM layer provides several choices for protection against optical layer failures such as a fiber break. The different survivability mechanisms at each layer basically provide different levels of trade-offs between the cost of recovery and the speed of recovery. Often, the survivability mechanisms providing very fast recovery from network failures are also very expensive and the survivability mechanisms that provide a cost-effective solutions are not fast enough. Hence, at present, one of the most important problems related to network survivability is improving the “speed of recovery” characteristics of the cost-effective survivability mechanisms. OSPF and shared mesh restoration are two such cost-effective mechanisms operating at the IP and optical WDM layers respectively. The next chapter examines the OSPF protocol with the goal of improving the failure recovery times at the IP layer while the current chapter examines the operation of shared mesh restoration with the objective of achieving fast failure recovery even in adverse network conditions.

The ever increasing demand for bandwidth in the Internet means that optical networks have become an indispensable part of the modern communication networks. Hence, the survivability at the optical layer becomes an important component of the overall network survivability. There exist several mechanisms for survivability at the optical layer. The failure recovery can be *local* or *end-to-end*. In local recovery, the optical cross-connects (OXCs) around the failed component (link/OXC) provide failure recovery by routing the failed connections around the failed component without involving the source/destination OXCs of the failed connections in the process. Local recovery holds the promise of very fast recovery from failures. However, a number of studies [8, 76, 103, 67, 144, 118] have confirmed that local recovery required significantly more capacity than end-to-end recovery. Among the end-to-end based recovery schemes, connection protection and restoration schemes have been identified as the cost effective and yet fast mechanisms for recovering connectivity in the face of service affecting network failures [41]. In the majority of these schemes, a connection is defined by two paths - one for service (*primary*) and one for restoration (*backup*). Within a distributed control plane framework, such as the Generalized-Multi Protocol Label Switching (GMPLS) framework being developed by the IETF [43], both of these paths are established via signaling (i.e., messages are sent from node to node to indicate desired actions). Protection schemes establish the backup path before service path failure. Such mechanisms, while minimizing the service restoration time, require dedicated resources on both the primary and backup paths. For most applications, this bandwidth inefficiency is simply too costly. Significant improvements in resource utilization can be achieved by instead sharing restoration capacity among connections that do not fail simultaneously. In order to share restoration resources, the restoration path is established only after the service path fails. We refer to such schemes as shared-mesh restoration [42]. In such schemes, when the service path for a connection is established, resources may be reserved along the restoration path without being allocated. When a failure occurs on the primary path, signaling along the restoration path allocates the necessary resources to the failed connection. Resource allocation typically involves selection and cross-connection of the restoration channels. It is crucial that the overall restoration process be rapid so as to minimize the service interruption time.

## 6.2 Factors Affecting Fast Failure Recovery in Shared Mesh Restoration

The connection restoration times in shared-mesh restoration schemes depend on multiple factors, such as the finite state machine of the signaling schemes, the optical cross-connect (OXC) architecture, the network topology and the restoration load. In the following, we discuss these factors in more details.

### 6.2.1 Restoration Signaling Schemes

There exist many variants for shared mesh restoration signaling schemes. In the following, we describe the general mechanism used by most of the shared mesh restoration schemes followed by different categories of such schemes.

The failure recovery mechanism of shared mesh restoration kicks in as soon as a failure is detected by an OXC in the vicinity of the failure. The OXC waits for a small duration of time, called *Alarm Age Delay*, and then notifies the end nodes (source and destination OXCs) of the optical connections affected by the failure. Usually, the failure detecting OXC sends the failure notification (henceforth called *Alarm*) only to one end node of the failed connection - the node which it can still reach along the primary path of the connection. Usually, the Alarm message is sent just to the source OXC. The following description assumes that this is the case. Once the source OXC receives the Alarm message, it initiates the restoration signaling process. The process starts with the source OXC sending a *Request* message towards the destination OXC along the backup path. Each OXC along the backup path processes the Request message, allocates channels for the connection and forwards the message to the next OXC along the backup path. On receiving the Request message, the destination OXC sends a *Reply* message back towards the source OXC again along the backup path. As each OXC along the backup path receives the Reply message, the allocated channels are cross-connected. When the Reply message reaches the source OXC, the backup path has been established.

Having described the basic signaling process, now we describe different categories of shared-mesh restoration signaling. There exist several variations of shared-mesh restoration signaling based on the following features:

1. number of ends initiating the signaling process
2. *Contention Avoidance* versus *Contention Resolution* signaling
3. channel cross-connection time at the OXCs
4. signaling message forwarding
5. *Misconnection* Avoidance mechanism

The failure recovery process can be initiated either by both source and destination OXCs or just the source OXC. The general signaling process for the later case has been described above. In the former case, both source and destination OXCs initiate the recovery process simultaneously by sending Request messages along the backup path towards each other. These messages trigger channel allocation in the intermediate OXCs before they meet each other at an OXC. This OXC generates the Reply messages which are sent back towards the source and destinations OXCs. As the intermediate OXCs receive the Reply message, they cross-connect the allocated channels. The backup path is established as soon as the Reply messages reach the source and destination OXCs.

The restoration signaling schemes can also be categorized on the basis of how they handle *contention* for resources. Resource contention can occur when two ends of an optical link assign the same resource (such as pair of ports) to two different bi-directional connections at effectively the same time. The *Contention Avoidance* schemes avoid contention altogether by requiring that only one node (usually the one with higher ID) is responsible for resource allocation on each link. In contention avoidance schemes, a slave OXC forwards the Request message to the next (Master) OXC without making the channel selections. The Master OXC makes the channel selections and lets the Slave OXC know via an *Acknowledgement* (Ack) message. On the other hand, *Contention Resolution* schemes do not attempt to avoid contention. Such schemes operate by resolving contention once it occurs, usually by having the node with the higher node ID win the contention. The connection losing the contention will have to undergo a time-consuming *backoff* process before a new attempt to restore it can be made.

In the previous discussion of the restoration signaling, we have assumed that the receipt of Request message triggers the allocation of channels to the connection whereas the receipt of Reply message causes the actual channel cross-connection to take place. However, the channel cross-connections need not necessarily wait till the receipt of Reply message. Faster failure recovery times can be achieved if the channel cross-connection is performed immediately after channel allocation on receiving the Request message. However, such a scheme will require the channel *de*-allocation if the backup path setup can not be completed due to contention. Once an OXC receives the message (Request/Reply) that will trigger the channel cross-connection, the OXC can either forward the message immediately to the next OXC (*forward before cross-connect*) or wait for the local channel cross-connection to complete before forwarding the message (*forward after cross-connect*). Another possibility is that the channel cross-connection is initiated by the Request message which is forwarded immediately however the Reply message received by the OXC is not forwarded until the local channel cross-connection is complete. Note that for contention avoidance signaling, only forward before cross-connect option is applicable, as the slave OXC may need to forward the message to the master for resource selection.

Since customer traffic is being lost during the execution of the restoration procedure, it makes sense to immediately cross-connect the customer traffic to the restoration connection while that connection is still being established. As a result, traffic gets through as soon as physical connectivity is achieved along the restoration path. The risk that must be avoided is customer data being temporarily sent to the wrong destination owing to contention resolution. We refer to this phenomenon as *misconnection*. Ruling out misconnection puts the following additional constraints on the protocols. Misconnection will not occur with contention avoidance signaling as long as there is no preemptable traffic using the restoration channels. However, contention resolution methods must be modified to avoid misconnection if they are to be used for restoration. Contention resolution signaling can avoid misconnection either if the customer traffic is not cross-connected to the restoration path by the source OXC until after every other OXC on the restoration path has finalized its local cross-connection channel selection, or if each OXC selects the channel on the receiving the Request message, but waits and performs the local cross-connection only upon receipt of the Reply message (i.e, after the contention has been resolved). The first method provides faster restoration of customer connectivity. In our experiments, we modify the contention resolution schemes according to the first method to ensure avoidance of misconnection. Note that while this step ensures that contention resolution signaling avoids misconnection, it also considerably slows down the restoration process.

### 6.2.2 Optical Cross-connect Architectures

The OXC node architecture determines the time to complete local cross-connections and hence impacts the overall restoration times for the failed connections. An optical network XC generally consists of an intelligent controller and a cross-connect fabric. The intelligent controller processes each signaling message, selects channels and initiates cross-connection requests that are performed by the cross-connect fabric. These two components may be integrated into a single unit or may exist as separate units communicating via an external interface (e.g. Fast Ethernet). Further, the two components may share a common processor or may have independent processing capabilities. Once the controller has selected channels for a new cross-connection, the time involved in completing the request consists of the waiting time before the request gets service (e.g., if it has to be queued waiting for other cross-connections to complete) plus the time to perform the physical cross-connection. The time required to perform the physical cross-connection depends on the technology used in the fabric. For example, popular MEMS technology involves physical mirror movement and may take 5-10 ms. However, the more worrisome delay is the waiting time before a cross-connection is serviced in the event of a failure when a surge of cross-connect requests arrive within a small time interval. Some XC architectures are able to reduce waiting times by executing multiple cross-connect requests together while others may eliminate the waiting time completely i.e. provide *parallel* execution of any number of cross-connect requests. These considerations led us to classify the OXC architectures into three categories:

1. *sequential* cross-connects where a XC operation cannot begin until after the previous one has completed;

|  |  |
|---|---|
| Processing times for Request and Alarm | 0.38 ms |
| Processing times for Reply and Ack | 0.05 ms |
| Forwarding delay for a message in transit | 0.1 ms |
| Alarm Age delay | 10 ms |

Table 6.1: Signaling Message Processing Times at The OXCs as Used in The Simulations

2. *parallel* cross-connects where each XC operation is performed immediately without any waiting;

3. *batch* cross-connects where a number of cross-connection requests are batched together to be processed simultaneously, but where each batch of commands must wait until the previous batch have been completed.

## 6.3 Evaluating OXC Architectures and Signaling Variations

Using the NS2 simulation tool [107], we modeled the different OXC architectures and signaling schemes to analyze their speed of recovery performance. The node message processing times were obtained from measurements attained using a non-commercial, prototype OXC testbed [89] and are shown in Table 6.1. We simulated a number of topologies, including the 21-node, 26-link ARPA2 network [86] with random traffic. The results reported next are for the ARPA2 network. Similar conclusions were drawn from other topologies. The simulations covered a whole spectrum of different parameters as detailed below:

1. *Restoration Signaling Variations*: We experimented with *contention avoidance* signaling where the cross-connections are done on the receipt of Request message and two variants (*forward before cross-connect* and *forward after cross-connect*) of

*contention resolution* signaling. The contention resolution signaling variants perform local cross-connection on getting the Request message but delay channel cross-connection at the source OXC until after every other OXC on the backup path is done (in order to avoid misconnection).

2. *Sequential*,*batch* and *parallel* OXC architectures with two different physical cross-connect times: 2-3 ms and 8-10 ms. The exact cross-connect time is determined according to a uniform distribution within these values.

3. To gauge the impact of increasing load on restoration times, we conducted experiments with 1000, 2000, 4000 and 8000 connections established between randomly selected source and destination OXCs. Each experiment involved sequentially failing every link in the topology and observing the restoration times for the affected connections. The time by which 90representative of the restoration time.

### 6.3.1 The Scaling Behavior of Cross-Connect Architectures

Figure 6.1 quantifies the deterioration in the restoration performance with sequential cross-connects as the connection load and the per-node cross-connect times increase. The results shown are for contention avoidance signaling. Similar results were obtained for other signaling schemes. Table 6.2 shows the 90 percentile recovery times for different OXC architectures for increasing connection load and physical cross-connect times. Clearly, sequential cross-connects scale poorly with the increase in cross-connect times and connection load. The fact that each cross-connect command has to wait for previously issued commands to finish results in massive increase in overall recovery times. In comparison to sequential cross-connects, the parallel cross-connects scale considerably better with increases in connection load. Figure 6.2 shows that, with parallel cross-connects, the

| **OXC Arch** | **1000 Conn.** | | **2000 Conn.** | | **4000 Conn.** | | **8000 Conn.** | |
|---|---|---|---|---|---|---|---|---|
| **XC Times** | **2-3** | **8-10** | **2-3** | **8-10** | **2-3** | **8-10** | **2-3** | **8-10** |
| **Sequential** | 469 | 1645 | 921 | 3276 | 1807 | 6437 | 3624 | 12952 |
| **Parallel** | 112 | 118 | 205 | 210 | 391 | 397 | 772 | 778 |
| **Batch** | 113 | 123 | 207 | 214 | 392 | 401 | 772 | 783 |

Table 6.2: Restoration times (90% of the connections restored) in milliseconds for different OXC architectures (ARPA2 topology, contention avoidance signaling)

increase in the cross-connect time causes only a minor increase in restoration times. However, as shown in Figure 6.2 and Table 6.2, parallel cross-connects also suffer from failure recovery times increasing linearly with the connection load in the network. This linear increase arises because the number of signaling messages and hence the queuing delays increase linearly with the number of connections. The linear increase in the failure recovery time with the increase in the connection load becomes clear from Figure 6.4, which shows the failure recovery times with Parallel cross-connects as the number of connections in the network increase. In Batch cross-connects, a cross-connect command might have to wait for the current batch of cross-connect commands to get executed before it gets service. This waiting time will be atmost the maximum physical cross-connect time. Hence, the performance of Batch cross-connects should be almost as good as that of Parallel cross-connects as long as there is no restriction on the number of cross-connect commands that can be simultaneously executed in a batch. Figure 6.3 shows the failure recovery time for Batch and Parallel Cross-connects for different physical cross-connect times when there are 8000 connections in the network and assuming that there is no restriction on the number of commands executed in a batch. It can be seen from this figure that Batch cross-connects with no restriction on the number of cross-connect commands in a batch are almost as good as Parallel cross-connects.

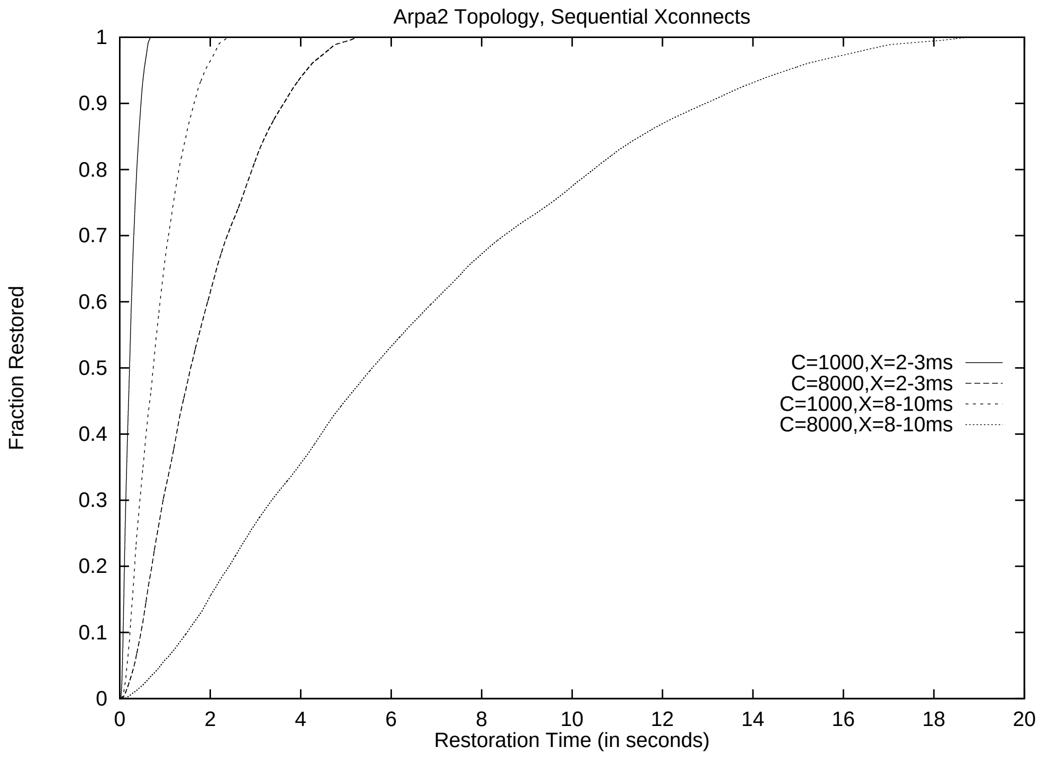


Figure 6.1: Failure recovery times with sequential cross-connects as the connection load increases from 1000 to 8000 and physical cross-connect time increases from 2-3 ms to 8-10 ms. (Contention avoidance signaling)

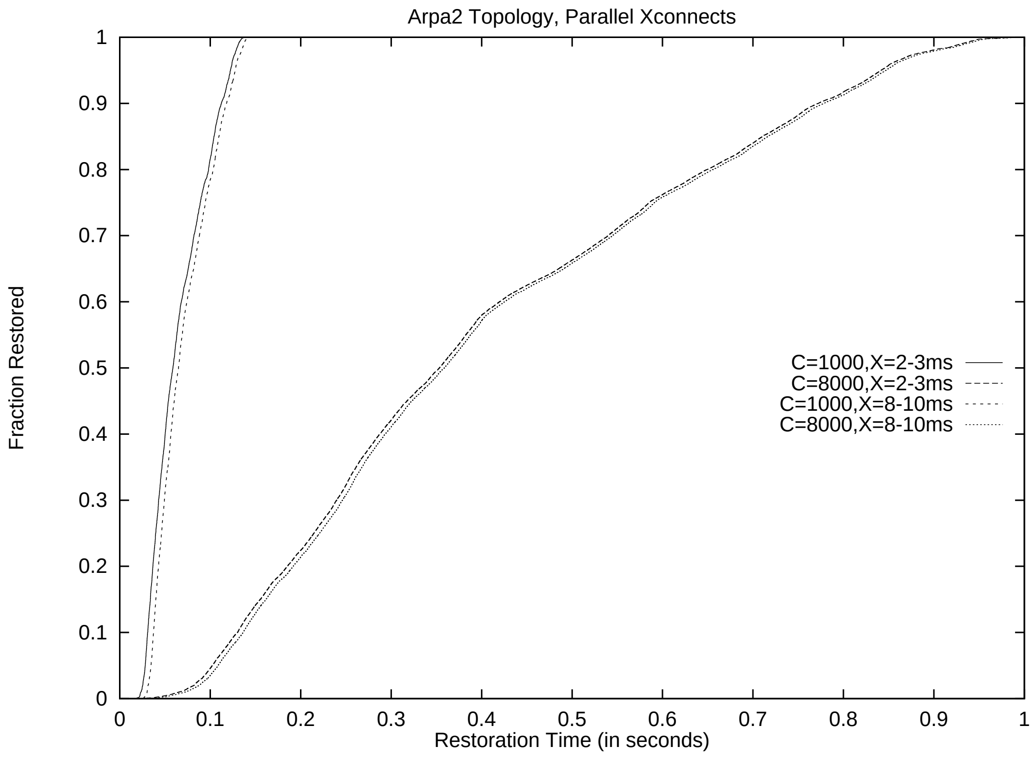


Figure 6.2: Failure recovery times with parallel cross-connects as the connection load increases from 1000 to 8000 and physical cross-connect time increases from 2-3 ms to 8-10 ms. (Contention avoidance signaling)

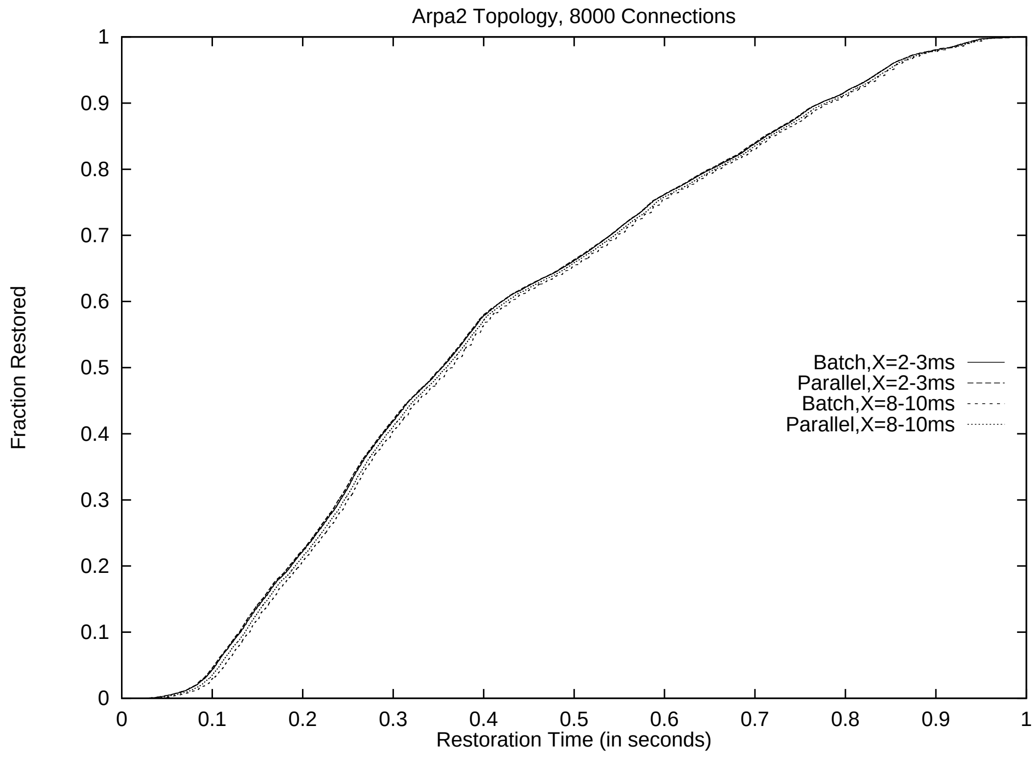


Figure 6.3: Batch cross-connects perform almost as well as parallel cross-connects in terms of failure recovery times. (Contention avoidance signaling)

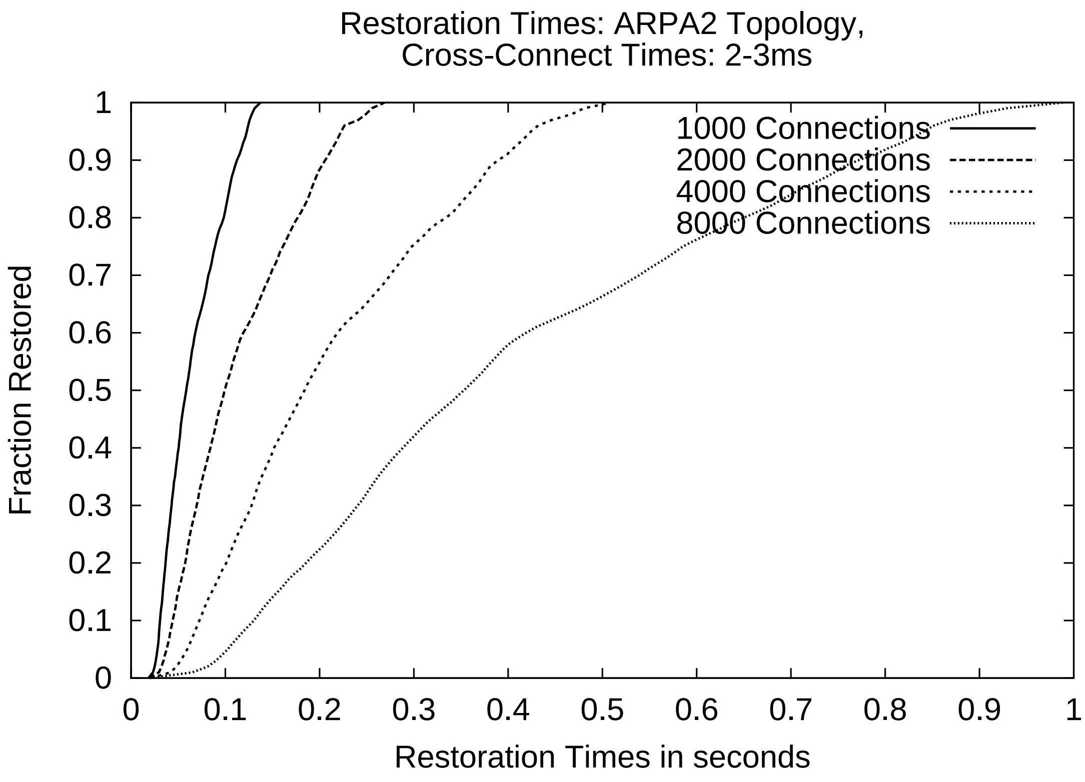


Figure 6.4: Failure recovery times with parallel cross-connects as the connection load in the network increases. (Contention avoidance signaling, physical cross-connect times: 2-3 ms)

### 6.3.2 Contention Avoidance Versus Contention Resolution

Now we compare the restoration speed performance of contention avoidance and contention resolution signaling schemes. We compare the versions where both schemes perform channel cross-connections immediately after channel selection. This means that for contention resolution scheme, the channel cross-connection can proceed on receiving the Request message, however, the contention avoidance scheme will have to wait for the master OXC to allocate channels before the channel cross-connections can take place. Naturally, contention resolution signaling should have somewhat better failure recovery times than contention avoidance signaling. However, while contention avoidance is naturally misconnection-proof, the contention resolution signaling will have to delay the channel cross-connections at the source OXC until after every other OXC on the backup path is done. Figure 6.5 shows the failure recovery times with contention avoidance signaling and contention resolution signaling with and without the delay in the channel cross-connections at the source OXCs. Clearly, the misconnection avoidance delay in contention resolution schemes makes them somewhat slower than contention avoidance schemes. It should also be noted that in case there are not enough backup channels in the network and hence resource contention is a real possibility, the performance of the contention resolution schemes will be even worse.

## 6.4 Faster Recovery in Shared Mesh Restoration with Aggregate Signaling

So far, our investigation on the impact of different OXC architectures and restoration signaling variants on failure recovery times has revealed that:

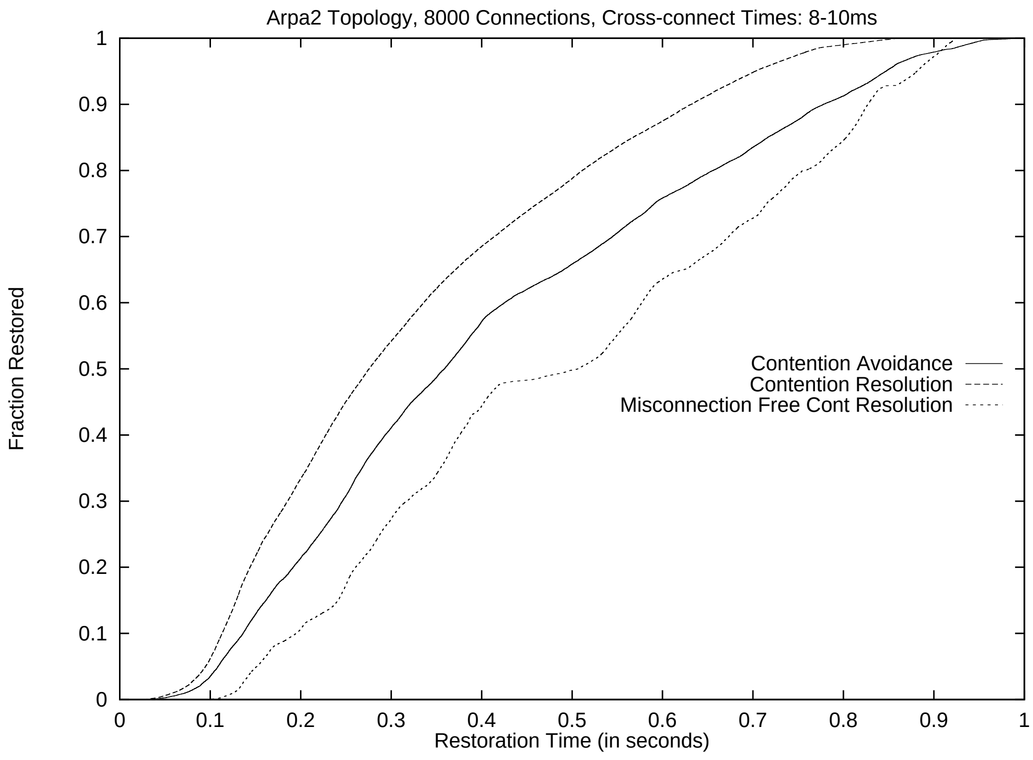


Figure 6.5: Failure recovery times with contention avoidance and contention resolution (with and without misconnection avoiding cross-connection delay at the source OXCs) schemes. (Parallel cross-connects, 8000 connections, physical cross-connect times: 8-10 ms)

1. Sequential processing of channel cross-connection commands in OXCs leads to very high restoration delays. It is important that OXCs can perform multiple channel-cross-connections in parallel.

2. The contention avoidance signaling schemes naturally avoid misconnections and thus result in somewhat faster restoration than contention resolution signaling schemes that require additional steps to avoid misconnection.

3. The restoration times increase roughly linearly with the number of connections in the network even for the Parallel OXC architecture and signaling scheme (Figure 6.4).

Among the observations made above, perhaps the most significant is the linear increase in the failure recovery times with increase in the connection load in the network. This is because such a behavior puts a question mark on the utility of shared mesh restoration in providing desired fast failure recovery (i.e. about 100 ms or less) in face of increasing number of connections in the network. A deeper look at the reasons behind such a behavior reveals that: As the number of connections in the network increase, more connections are affected by a given network failure and hence more signaling messages are generated in response to a network failure to restore the affected connections . Figure 6.6 shows the number of different signaling messages generated (for Parallel cross-connects, contention avoidance signaling) as the connection load increases. Clearly, the number of signaling messages are increasing linearly with connection load. As a result, the queuing delays suffered by the signaling messages while waiting to be processed by the OXCs also increase proportionately with the connection load. The observed increase in restoration times is a direct result of the increased queuing delays. The ARPA2 topology has 21 OXCs and 26 links. The uniformly random assignment of 8000 connections between 420 unique

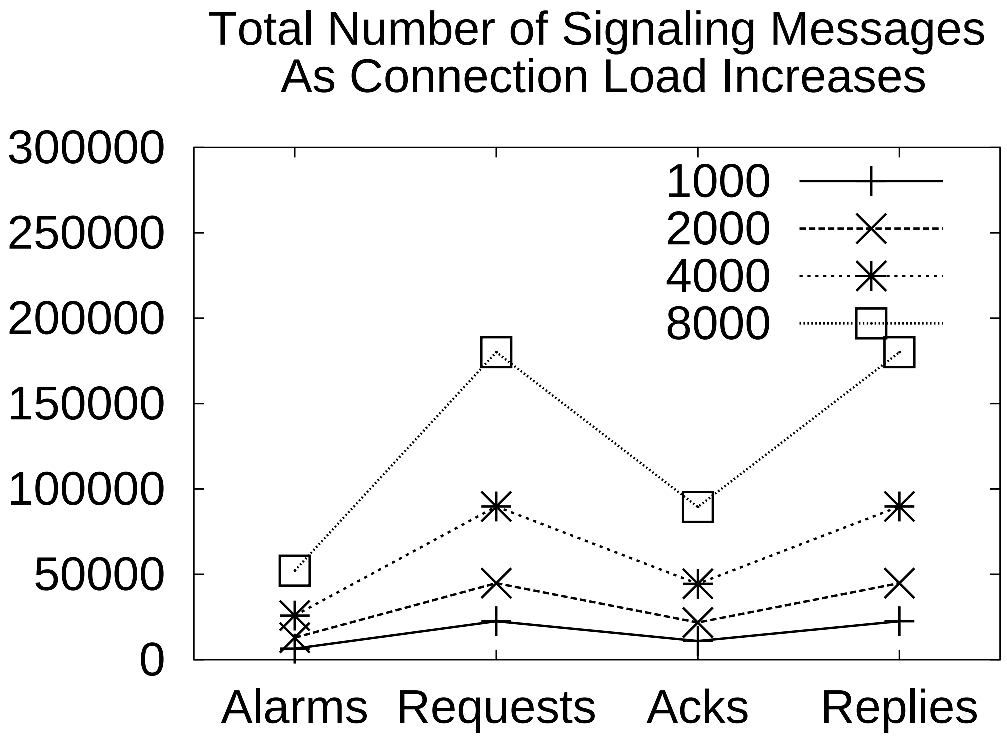


Figure 6.6: The increase in the number of signaling messages generated during connection restorations as the connection load in the network increases (Contention avoidance signaling, parallel cross-connects, 8000 connections, physical cross-connect times: 2-3 ms)

source- destination pairs, used in our experiments, results in average 19.05 connections between any source-destination pair and we observed that the channel cross-connection capacity requirements on the OXCs are below 2100. A typical state of the art OXC will have 256x256 ports with each port supporting up to 48 or 192 STS-1 connections and hence can easily support much more than 2100 simultaneous cross- connections. Hence, a load of 20 connections per source- destination pair seems a reasonable possibility. The unsatisfactory recovery times obtained for this level of connection load with shared mesh restoration indicates a lack of scalability in terms of connection loads.

The deterioration in recovery times can be avoided if the number of signaling messages generated while restoring the failed connections is reduced. It is possible to achieve dramatic reduction in the number of signaling messages by employing *Aggregated Signaling* schemes rather than *Per-connection Signaling*. The current set of signaling schemes proposed for use with shared mesh restoration can be characterized as per-connection signaling as one signaling message helps restore only one failed connection. In contrast, in aggregated signaling, one message will help restore many connections. Several different aggregated signaling schemes are possible based on different ways to limit the number of signaling messages. In the following, we propose a number of aggregated signaling schemes and examine their effectiveness in achieving fast failure recovery times with shared mesh restoration.

## 6.5 Different Aggregate Signaling Schemes

In this section, we propose several different aggregated signaling schemes. These schemes achieve reduction in the number of signaling messages by combining signaling information where possible.

1. **Aggregated Alarms:** The number of alarms generated in response to a failure event can be reduced by grouping together the affected connections with same source OXC and sending just one *aggregated alarm* per source OXC. This scheme provides the basis for other aggregated signaling schemes by allowing a source OXC to learn about all its failed connections simultaneously. The aggregared alarms can also be used as such with per- connection signaling.

2. **Aggregation Over Common Paths (AOCP):** Substantial reduction in the number of signaling messages can be achieved if all the connections sharing the same backup path are restored together with same set of signaling messages. In this scheme, a source OXC, upon receiving the failure notification via aggregated alarm, initiates simultaneous restoration of all the failed connections sharing the same backup path by sending one Request message along the common backup path. The Request message is processed and forwarded along the backup path triggering channel cross-connections for all the connections identified therein. The Ack and Reply messages generated in response to the aggregated Request message are again aggregated in nature serving all the connections together.

3. **Aggregation Over Next Hop (AONH):** In this scheme, the source OXC sorts the failed connections in groups with same next OXC on their backup paths. This is followed by sending one common Request message to each next OXC for all the connections having that OXC next on their backup paths. On receiving such a Request message, the next OXC repeats sorting signaled connections in groups with same next OXC and sending one common Request message forward for all the connections sharing the same next OXCs. Thus, in this scheme, the Request messages

split as the next OXCs of the signaled connections diverge. If the OXC receiving a Request is the destination for a subset of connections identified therein, it generates a Reply message for these connections and sends it back towards the source OXC along their common restoration path. Note that the *Aggregation Over Common Paths* (AOCP) is actually a special case of *Aggregation Over Next Hop* (AONH) signaling.

4. **Aggregation Over Next Hop With Delay :** It is possible to further reduce the number of signaling messages in Aggregation Over Next Hop scheme, if we allow a signaling message to wait for a short while at an OXC after it is processed. Such a wait will make it feasible for the OXC to re-aggregated multiple signaling messages based on common next hop. We refer to this approach as *Aggregation Over Next Hop With Delay*. Note that *Aggregation Over Next Hop* is a special case of *Aggregation Over Next Hop With Delay*, when the delay is set to 0.

## 6.6 Benefits of Aggregated Signaling

In the following, we present the simulation results evaluating the performance of different Aggregated Signaling schemes proposed in the previous section. The simulation results presented here are based on ARPA2 topology and Parallel OXCs. In the previous sections, we saw that the contention avoidance signaling performs better than other signaling variants. Hence, the aggregated signaling schemes evaluated in this section are based on contention avoidance signaling.

### 6.6.1 Aggregated Signaling Vs Per-connection Signaling

Figure 6.7 shows the connection restoration times with per-connection signaling, per-connection signaling with aggregated alarms and AOCP signaling for 8000 connections on

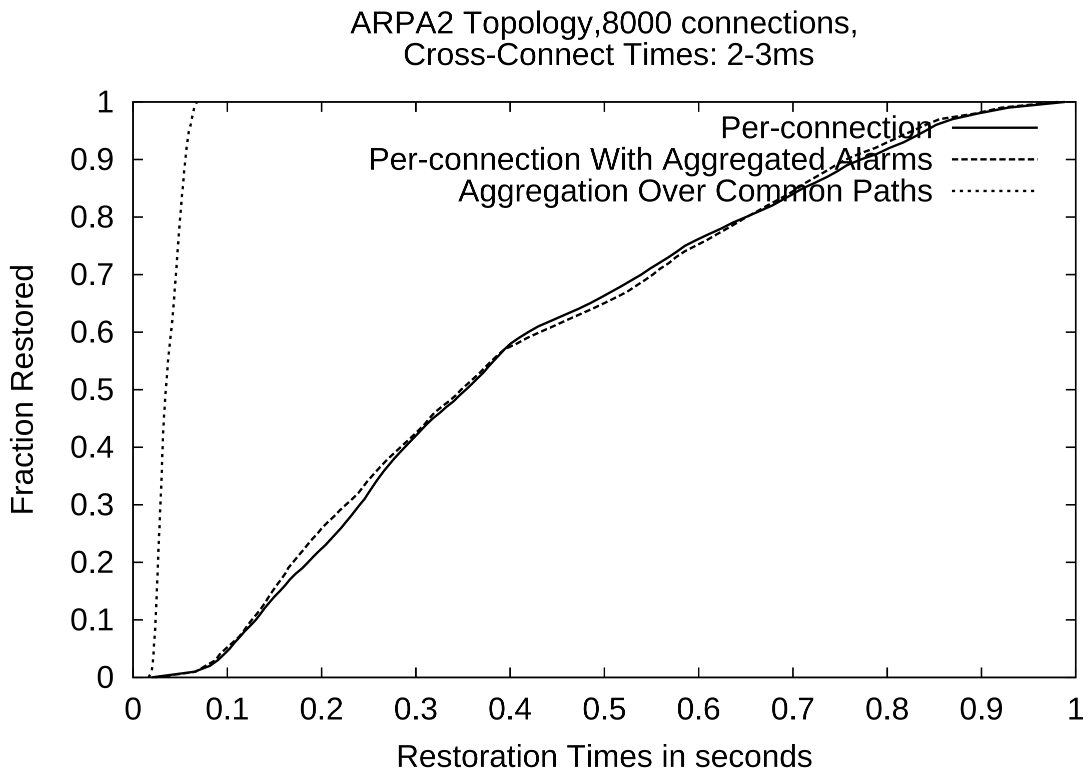


Figure 6.7: Connection restoration times with per-connection signaling, per-connection signaling with aggregated alarms and aggregation over common paths signaling. (Parallel cross-connects, 8000 connections, physical cross-connect times: 2-3 ms)

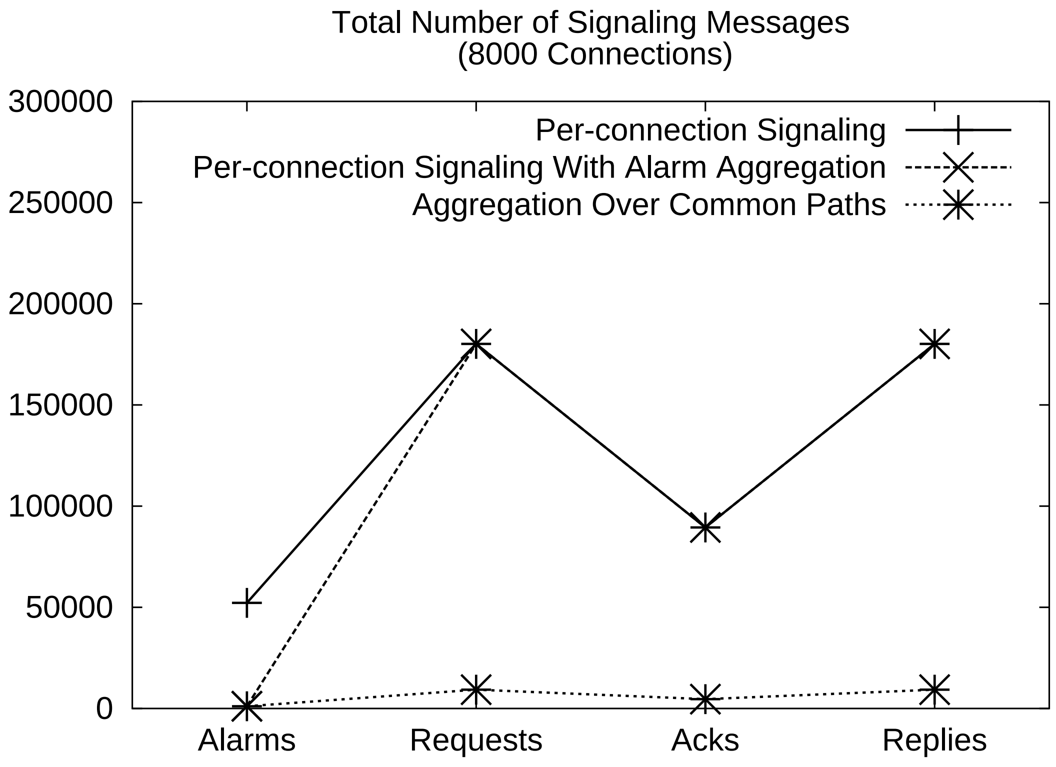


Figure 6.8: Signaling messages generated during connection restorations for per-connection signaling, per-connection signaling with aggregated alarms and aggregation over common paths signaling. (Parallel cross-connects, 8000 connections, physical cross-connect times: 2-3 ms)

ARPA2 topology. Figure 6.8 shows the number of signaling messages generated during connection restorations with these schemes. It can be seen from Figure 6.7 that even the simplest signaling scheme, AOCP, is able to dramatically reduce the failure recovery times. However, just using aggregated alarms with per-connection signaling does not impact failure recovery times at all. The reasons behind such a behavior are easy to understand. Most of the queuing delays occur due to the storm of Request/Reply/Ack messages generated during failure recovery with Alarm messages making a very small contribution. Hence, reducing the number of alarm messages by using aggregated alarms (as shown in Figure 6.8 does not help significantly. However, AOCP signaling is able to dramatically reduce the number of Request/Reply/Ack messages (Figure 6.8) by using common signaling messages for all the connections sharing a path, which results in drastic reductions in the connection restoration times.

### 6.6.2 Comparing Different Aggregated Signaling Schemes

Now we turn our attention to comparing the restoration performance of different aggregated signaling schemes. Figure 6.9 compares the performance of AOCP and AONH signaling schemes. It is clear from the figure that AONH has significantly better restoration performance than AOCP scheme. It should be noted that the performance of AOCP scheme depends on the number of connections sharing the same backup path end-to-end . As we will see later, the backup path sharing among connections is deeply influenced by the routing scheme used to calculate backup paths. Hence, the performance of AOCP scheme is greatly influenced by the routing scheme in use. On the other hand, it is intuitive to expect that the AONH scheme will be less dependent on particular routing scheme used. Next, we compare the performance of pure AONH scheme with that of AONH With Delay scheme

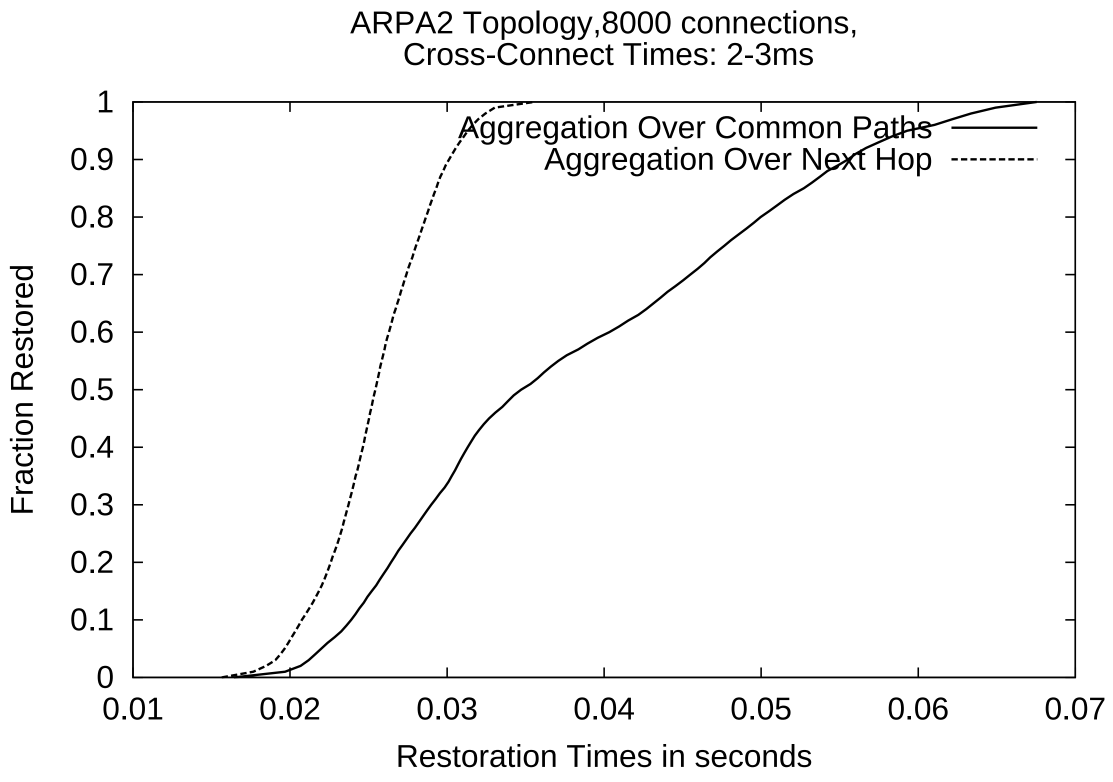


Figure 6.9: Comparing the failure recovery performance of aggregation over common paths and aggregation over next hop signaling schemes. (ARPA2 topology, parallel cross-connects, 8000 connections, physical cross-connect times: 2-3 ms)

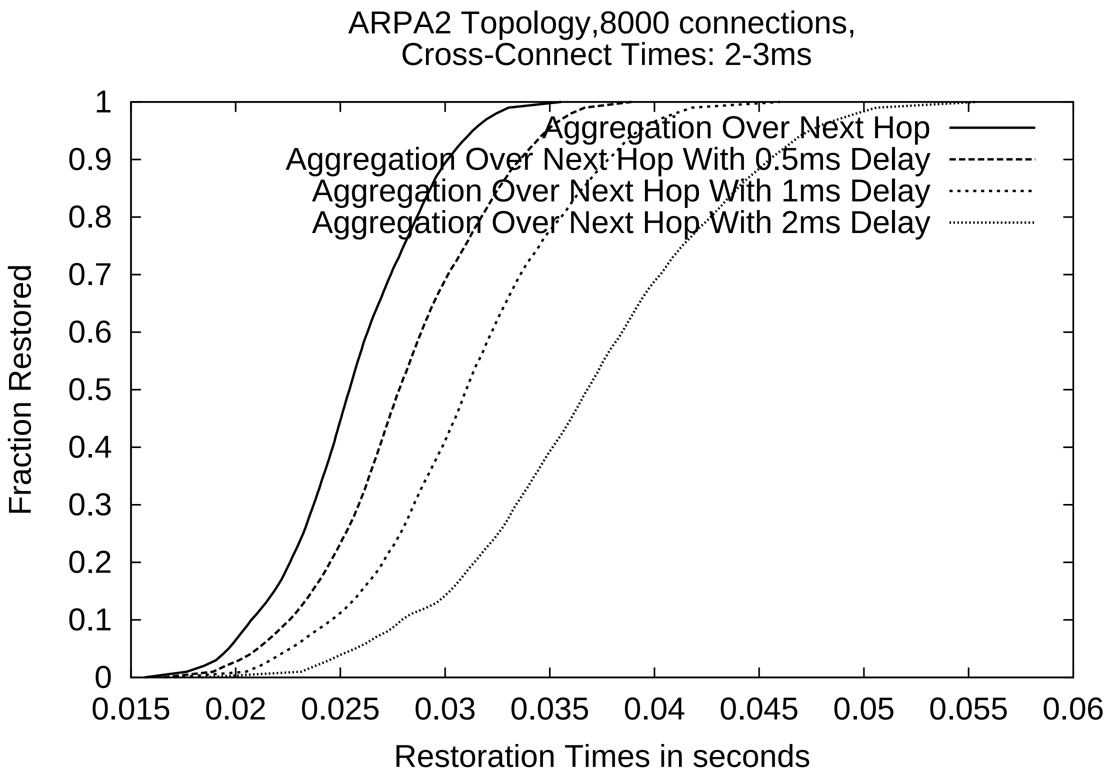


Figure 6.10: Comparing the failure recovery performance of aggregation over next hop and aggregation over next hop with delay signaling schemes. (ARPA2 topology, parallel cross-connects, 8000 connections, physical cross-connect times: 2-3 ms)

(Figure 6.10). It can be seen from Figure 6.10 that delaying the forwarding of signaling message to facilitate better aggregation does not usually help. It appears that most of the queuing delays in the processing of signaling messages are removed by the aggregation achieved with AONH signaling. Thus, the reduction in the number of signaling messages achieved with delay in forwarding the messages is largely inconsequential and the delay itself simply causes increase in restoration times. Even if delaying the message forwarding could help in certain scenarios, it is difficult to ascertain what should be the optimal delay value. Hence, given the uncertain benefits, it is prudent not to delay signaling message forwarding in AONH scheme. On the basis of these results, it can be concluded that AONH signaling is the most appropriate practical choice for signaling message aggregation.

## 6.7 The Performance of Aggregated Signaling in Less Favorable Circumstances

The performance of the aggregated signaling schemes reported in the previous section was based on the assumptions that:

1. the processing times for aggregated messages is the same as for non-aggregated messages;

2. there is no limit on the number of connections that can be signaled within a single message;

3. the backup paths are calculated using shortest path routing after excluding the links on the primary path.

The first assumption is based on the observation made in other control plane protocols (e.g. OSPF) that the per-packet processing overhead dominates the total message processing

[128]. The validity of the second assumption depends on how much connection-related information is carried in the signaling message. If it is only the connection ID, which would be adequate for shared mesh restoration [43], we can assume that fairly large numbers of connections can be signaled together in a message. Finally, the third assumption will not be valid if the restoration paths are selected so as to increase the resource sharing among connections [87]. In this case, connections with a common source and destination may not share the same restoration path which will presumably have a significant effect on the performance of AOCP scheme. The performance of other signaling aggregation schemes will also be affected since the restoration path will no longer be the shortest available path.

In the following, we investigate the performance of the aggregated signaling schemes when one or more of the previously mentioned assumptions do not hold.

**Impact of Increase in Message Processing Times as the Number of Connections Signaled Together Increases:**

The time it takes an OXC to process a signaling message can be divided into a fixed and a variable component. The fixed component corresponds to the time spent in executing the *packet received* interrupt and then copying the message from network interface to the main memory. The variable portion consists of the time spent in processing the contents of the message. Experience with OSPF LSA processing, where multiple LSAs are bunched together in a single LS Update packet, suggests that the total processing time increases linearly with the size of the packet irrespective of the number of LSAs contained therein [128]. Assuming that processing of OSPF LSAs is similar in nature to processing of signaling messages, increased contents of a signaling message as a result of aggregated signaling should not impact its processing time on an OXC. Nevertheless, we examine

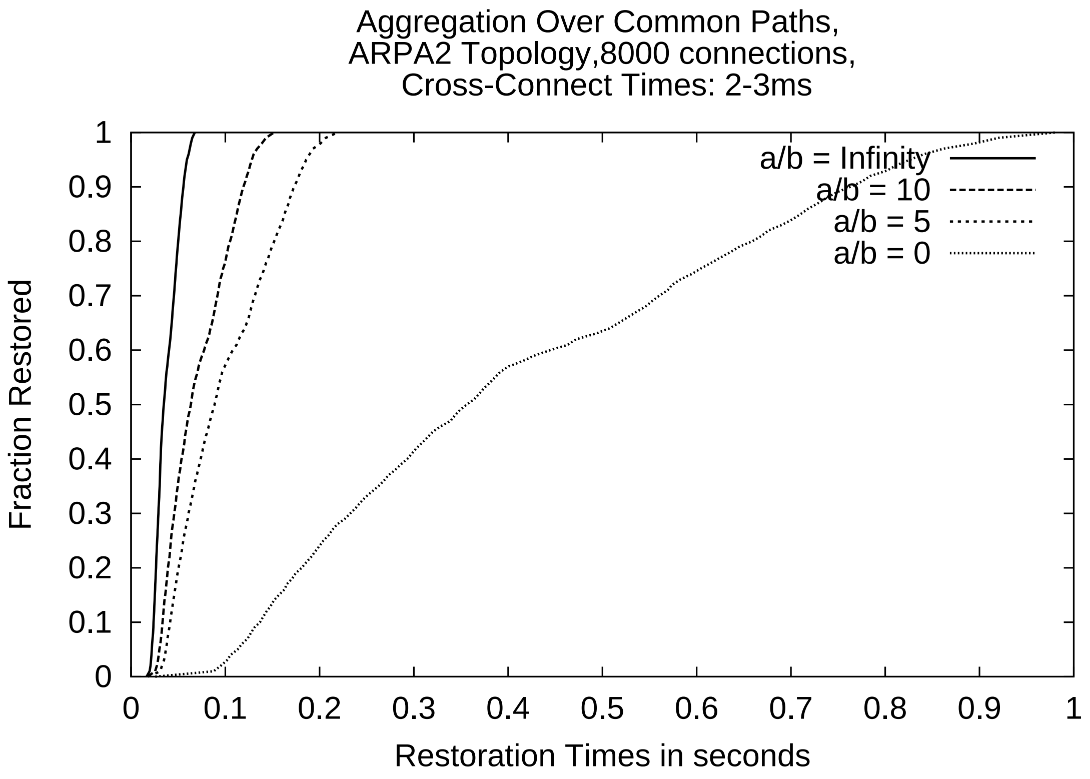


Figure 6.11: Deterioration in failure recovery time with aggregation over common paths signaling as the message processing time increases with connections signaled. a and b represent the fixed and variable parts of the message processing time. (ARPA2 topology, parallel cross-connects, 8000 connections, physical cross-connect times: 2-3 ms)

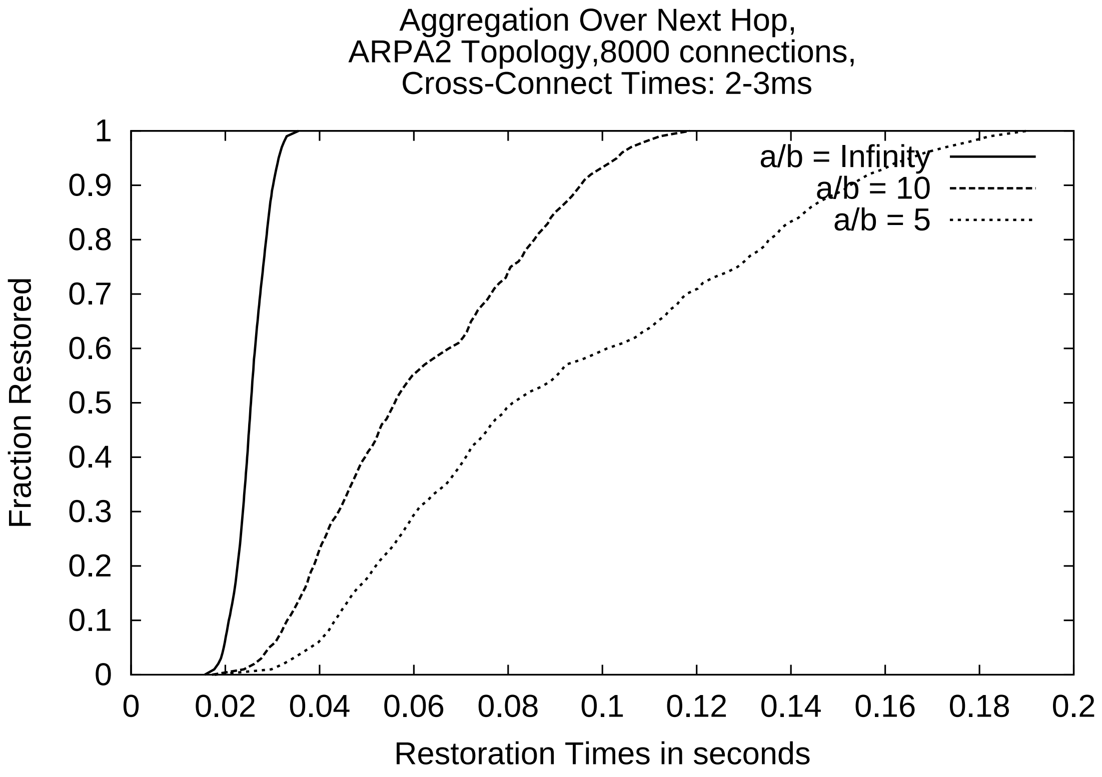


Figure 6.12: Deterioration in failure recovery time with aggregation over next hop signaling as the message processing time increases with connections signaled. a and b represent the fixed and variable parts of the message processing time. (ARPA2 topology, parallel cross-connects, 8000 connections, physical cross-connect times: 2-3 ms)

how the performance of aggregated signaling schemes will suffer if the message processing times increase as a result of increased load. It is reasonable to represent the load of a message in terms of the number of connections it signals. Let $a$ be the fixed part of the processing time and $b$ be the per-connection processing time. Then the total message processing time $t$ for a message load on $n$ connections can be modeled as: $t = a + n \times b$. Figures 6.11 and 6.12 show the deterioration in the performance of aggregated signaling schemes as the ratio $a/b$ decreases. It is clear from these figures that as the variable part of the processing time becomes more prominent, the efficacy of aggregated signaling in reducing the restoration delays decreases. When the message processing time doubles with the number of connections signaled, the aggregated signaling becomes completely useless giving same restoration times as the per-connection signaling.

**Impact of Limitation on The Number of Connections That can be Signaled Together:**

The number of connections that can be signaled together in a message depends on the per-connection information required for restoration purposes. If a connection can be identified by a simple ID and most of the connection information (such as channel requirements) is already available with the OXCs, a signaling message should be able to carry a large number of connections. Otherwise, the number of connections that can be signaled together in a message will necessarily be limited. The performance of aggregated signaling schemes will necessarily deteriorate if there is a limit on the number of connections that can be signaled together. In Figure 6.13, we show the deterioration is restoration performance of AOCP and AONH schemes as the number of connections that can be signaled together is reduced to 5. Clearly, such a move has huge performance implication for the two schemes.

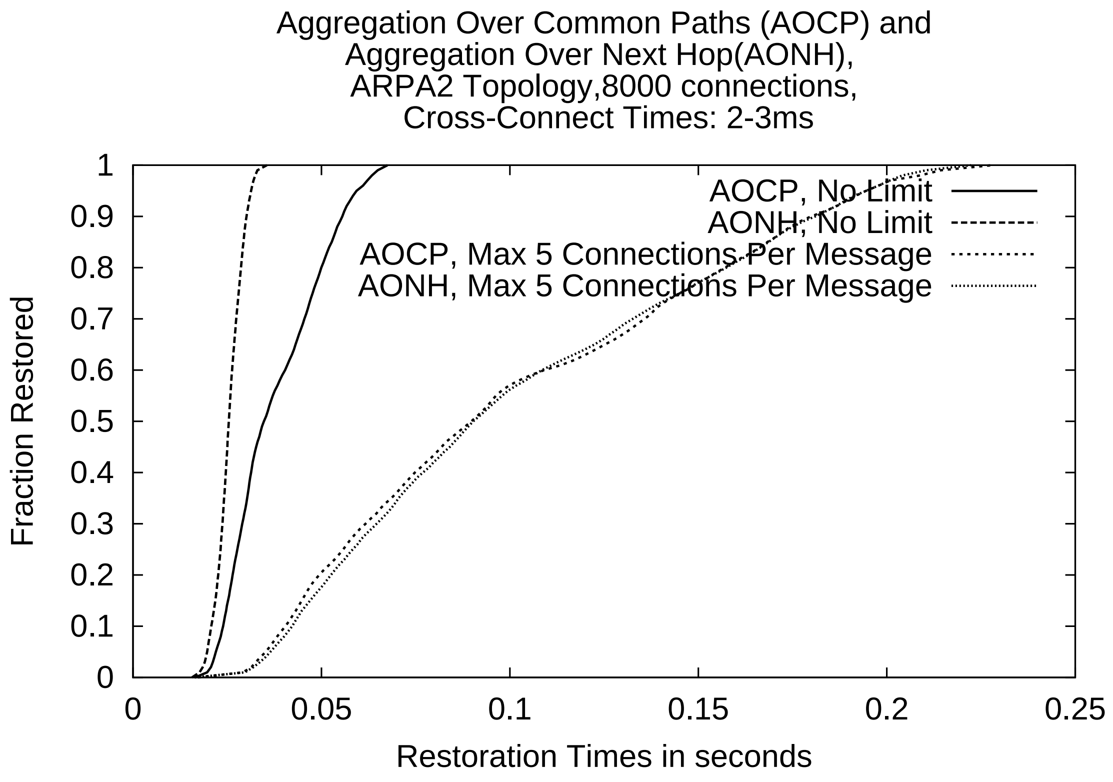


Figure 6.13: Deterioration in failure recovery time as the number of connections signaled in a message is limited to 5. (ARPA2 topology, parallel cross-connects, 8000 connections, physical cross-connect times: 2-3 ms)

Also, it should be noted that the performance difference between the two schemes become minor as the number of connections that can be signaled together is reduced.

**Impact of Routing Scheme Used on the Performance of Aggregated Signaling Schemes:** The performance of aggregated signaling schemes will be affected if the backup paths are not the shortest paths after excluding the links on the primary paths. The use of shortest path routing makes sure that the restoration signaling messages have to travel the smallest number of hops. Further, it ensures that all the connections between the same source and destination OXCs will follow the same backup path (unless there exist multiple shortest backup paths)i, whereas for a different routing scheme such connections might take different paths. The number of connections sharing the same backup path will have a significant influence on the performance of the AOCP signaling. In order to understand these efects, we compared the performance of the aggregated signaling schemes with different routing schemes. The two routing schemes used in this comparison were shortest path routing and the scheme proposed in [87]. The routing scheme presented in [87] calculates the routes using shortest path routing after modifying the static link weights in such a manner that the sharing of backup resources is increased [14]. This routing scheme encourages the sharing of backup channels between connections that will not fail simultaneously. Hence forth, we refer to the shortest path routing scheme as *SPR* and the scheme proposed in [87] as *RCR* (*Routing to Reduce Capacity Requirements*). With the RCR routing, the connections that would have otherwise shared the same backup path now assign different weights to the links and hence choose different backup paths. In Figure 6.14, we show the number of connections sharing a common backup path for SPR and RCR routing schemes. It is clear from this figure that RCR scheme can significantly reduce the number of connections

[14]The details about the routing scheme presented in [87] can be seen in Section 4.3.2.

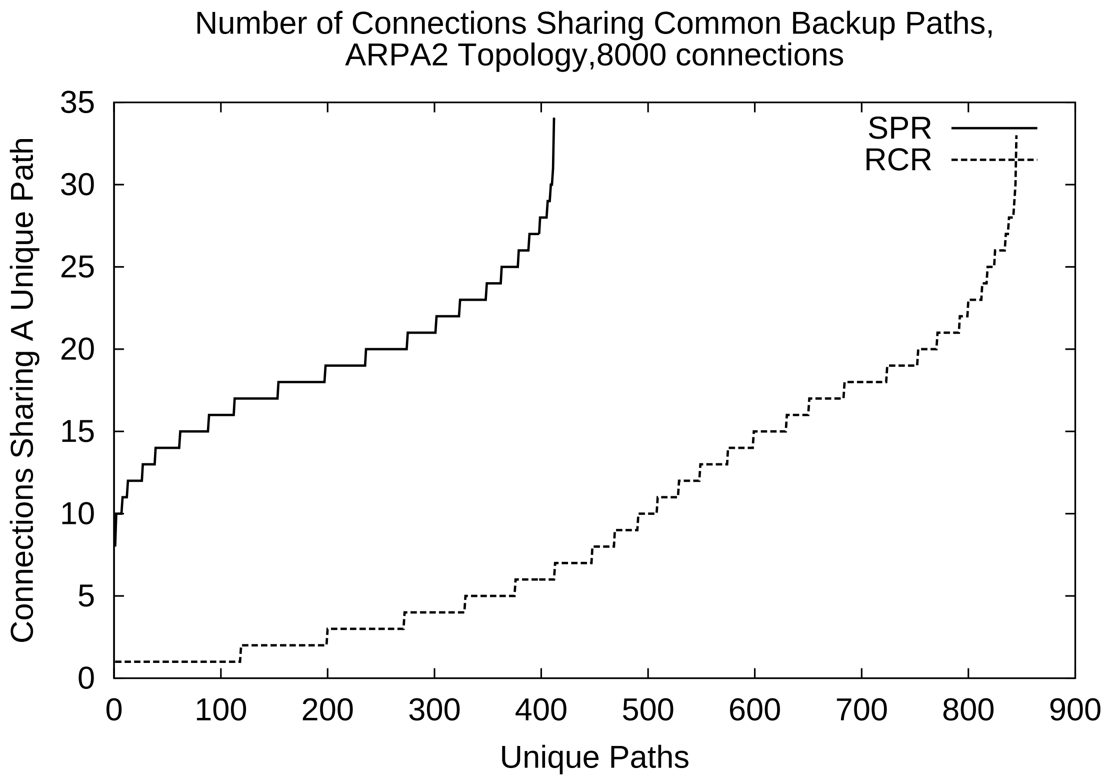


Figure 6.14: The number of connections sharing a common backup path reduce drastically as routing scheme is changed from SPR to RCR. (ARPA2 topology, 8000 connections)

sharing a common backup path. Such a behavior means that the performance of AOCP signaling will be significantly impacted by using RCR scheme rather than SPR. Figures 6.15 and 6.16 show the deterioration in the performance of AOCP and AONH signaling schemes respectively as the backup path calculation scheme is changed from SPR to RCR. As expected, the reduction in the number of connections sharing a backup path causes significant deterioration in the performance of AOCP signaling when used with RCR routing. On the other hand, since AONH signaling does not depend on many connections sharing the same backup path, the performance of this scheme is not significantly impacted by the use of RCR routing.

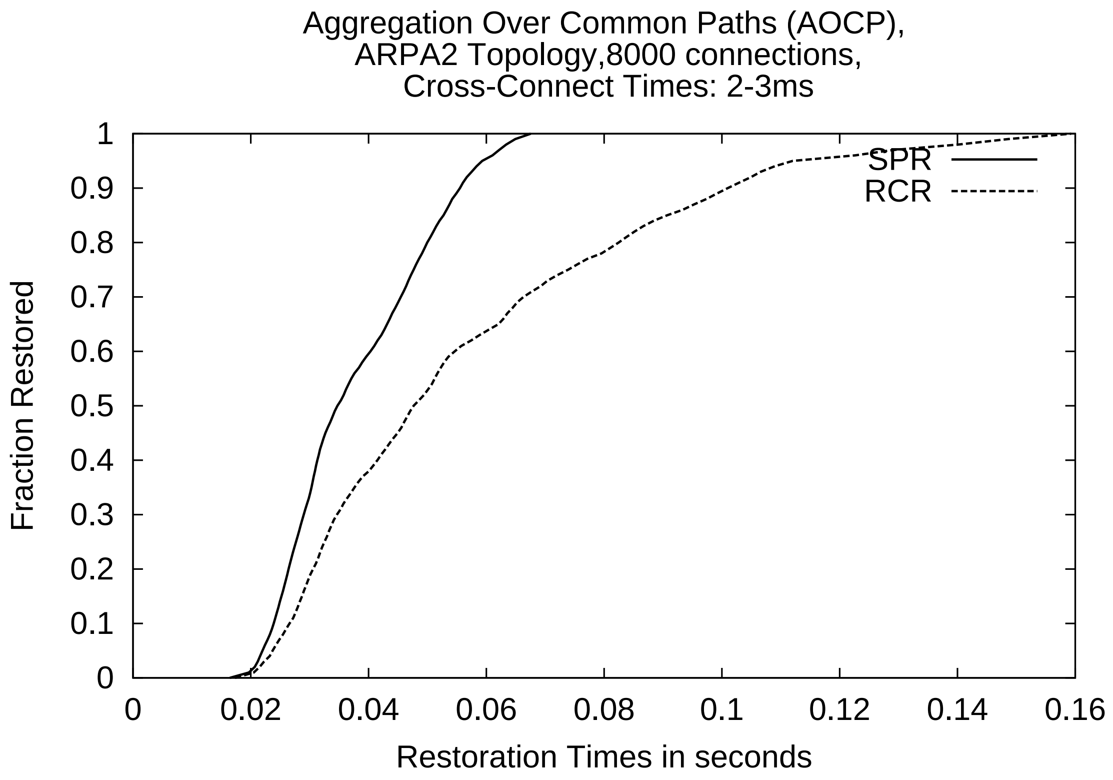


Figure 6.15: The restoration performance of aggregation over common path signaling as routing scheme is changed from SPR to RCR. (ARPA2 topology, 8000 connections)

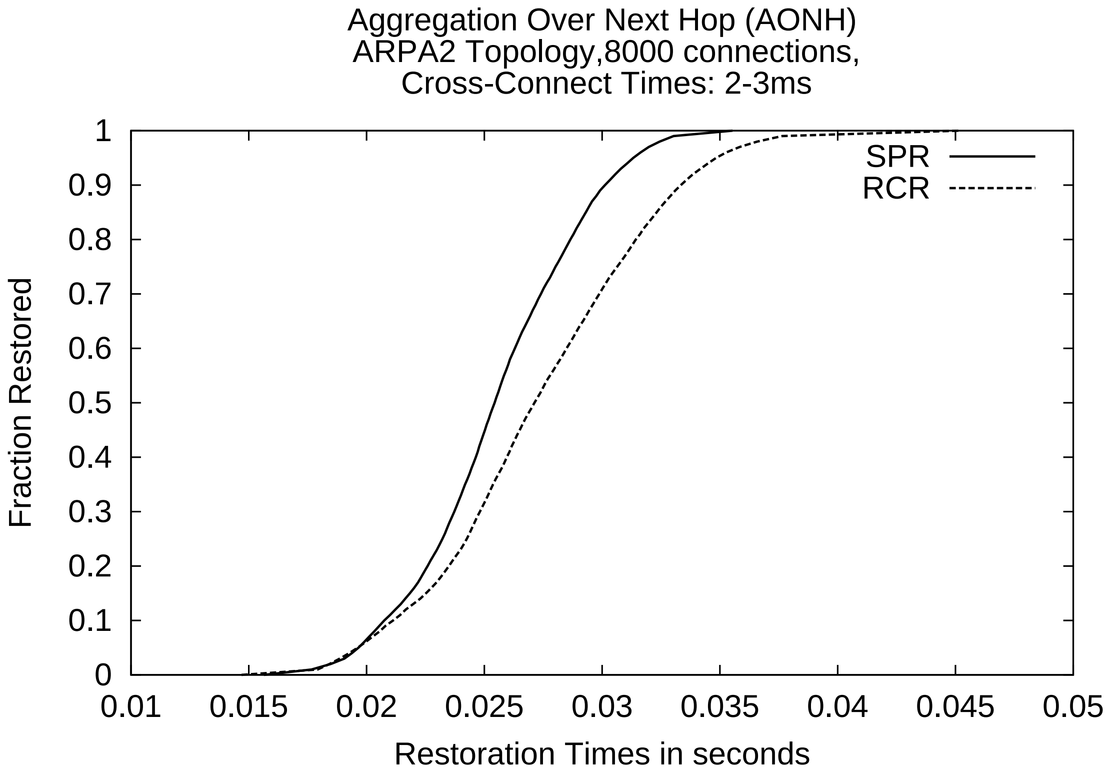


Figure 6.16: The restoration performance of aggregation over next hop signaling as routing scheme is changed from SPR to RCR. (ARPA2 topology, 8000 connections)

**Combined Impact of All Unfavorable Conditions:** Finally, we examine the combined effect of all the factors mentioned above on the performance of aggregated signaling schemes. Figure 6.17 shows the deterioration in achieved restoration times with aggregated signaling schemes as the "best case" scenario (characterized by SPR routing, fixed message processing times and no limitation on aggregation) changes into a "not so favorable" scenario (characterized by RCR routing, message processing times increasing with connections signaled with $a/b = 10$ and a limit of 10 on the number of connections signaled together). Note that the performance of the aggregated signaling schemes in this "not so favorable" scenario is still much better than what we get with per-connection signaling. This result indicates that the aggregated signaling schemes will have great practical utility even in unfavorable conditions. Also note that, there is not much difference in the performance of AOCP and AONH signaling schemes in the "not so favorable" scenario. Thus, the choice of which aggregated signaling scheme to use will inherently depend on the implementation complexity of different schemes.

## 6.8 Summary

Shared mesh restoration offers significant capacity savings over the protection based schemes. However, there are some doubts regarding the ability of shared mesh restoration in providing fast recovery from network failures. In this chapter, we analyzed different factors that impact the failure recovery times obtained with shared mesh restoration scheme. Our analysis pointed out that Sequential processing of cross-connect commands in the OXC is unacceptable for fast failure recovery with shared mesh restoration. We compared the performance of several different variations of shared mesh restoration signaling and showed that Contention Avoidance signaling results in faster failure recovery than other schemes.

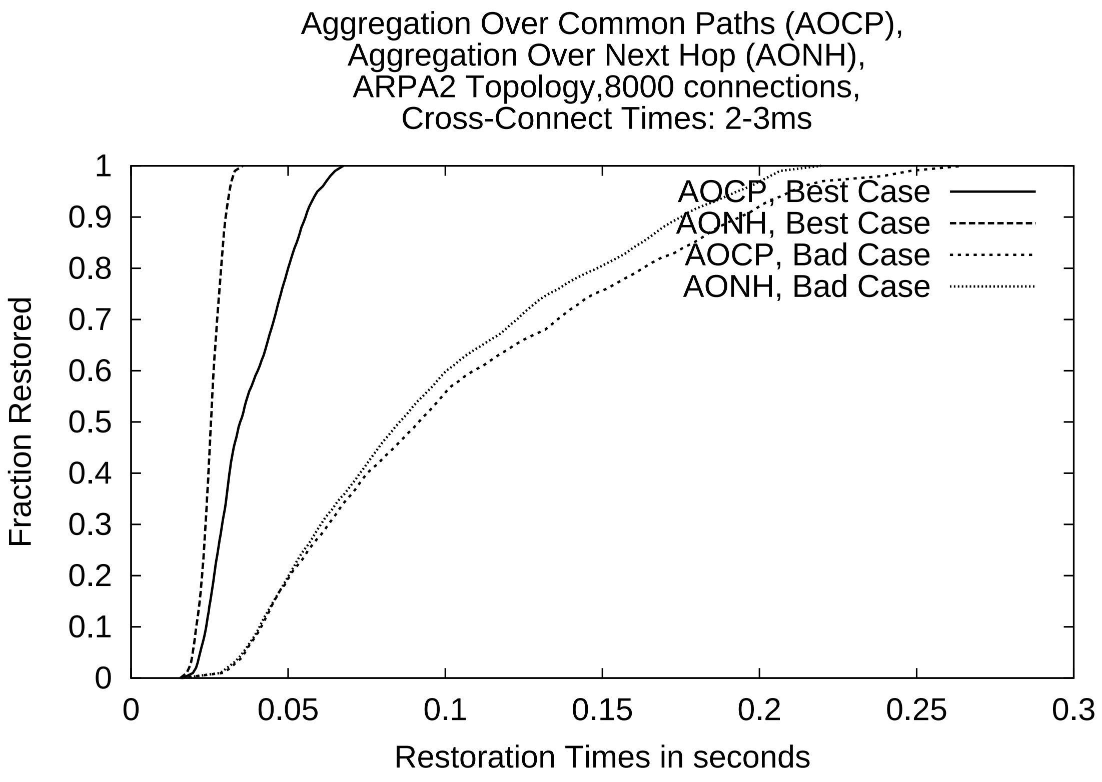


Figure 6.17: The restoration performance of different aggregated signaling schemes with and without favorable assumptions. (ARPA2 topology, 8000 connections, physical cross-connect times: 2-3 ms)

We showed that the use of conventional per-connection signaling in shared mesh restoration can lead to unacceptably large recovery times as the connection load on the network increases. We proposed several aggregated signaling schemes that will help in achieving fast recovery times with shared mesh restoration when the connection load on the network is high. We examined the performance of these schemes under a wide range of scenarios. The results show that under each scenario the aggregated signaling provides significant gains over per-connection signaling. The next steps to the work done so far consist of examining the implementation complexity of the proposed aggregated signaling schemes and performing a cost/benefit analysis of different options.

# CHAPTER 7

# FAST RECOVERY FROM NETWORK FAILURES IN OSPF NETWORKS

## 7.1 Introduction

The previous chapter examined the operation of shared mesh restoration, a cost effective mechanism to achieve protection against failures at the optical WDM layer, with the goal of improving its “speed of recovery” characteristics. In the current chapter, we shift our attention to the protocols at the IP layer. Like the optical layer, the IP layer offers several choices for protection against failures occurring at the IP layer. Traditional OSPF [102] routing and modern MPLS based end-to-end/local recovery mechanisms constitute the popular choices for survivability mechanisms at the IP layer. In Chapter 4, we compared these mechanisms in terms of the cost of survivability and observed that the traditional OSPF based recovery and modern MPLS based end-to-end recovery provide protection from IP layer failures at a much smaller cost than MPLS based local recovery mechanisms. As in the optical layer, the local recovery mechanisms at the IP layer suffer from a very high cost of operation even though they provide very fast recovery from network failures. Further, we observed that no clear favorite emerges when we look at the cost effectiveness of traditional OSPF based recovery and modern MPLS based end-to-end recovery. However, OSPF based recovery enjoys a big advantage over modern MPLS based mechanisms since OSPF already has a huge deployed base in the current Internet infrastructure. In fact, OSPF (and a very similar protocol IS-IS [111]) is the most popular choice for *Interior Gateway Protocol*, i.e. the protocol for routing within a pre-defined *area* in the network, in the Internet today. On the other hand, MPLS based features are only beginning to appear in the networking equipment and the implementations are far from being mature. Deploying MPLS based mechanisms in the commercial IP networks will take billions of dollars worth of investment. Hence, OSPF, with its mature implementation and large deployed base, seems to be an overall more cost-effective choice than MPLS based mechanisms.

In spite of its cost-effectiveness, the current OSPF standard leaves much to be desired when it comes to the speed of recovery from network failures. In its current standardized form, the OSPF protocol takes several tens of seconds to detect a failure and re- route the affected traffic on alternate paths. During the transient period, delivery of packets to the destinations is disrupted, i.e. the packets may be lost or delayed. Such a long disruption in the Internet service is clearly not acceptable. In this chapter, we examine how to reduce the time that OSPF protocol takes to recover from failures. The main reason for the slow failure recovery behavior of OSPF is its overly conservative design. The current standardized version of OSPF has been designed with a view to limit the protocol's processing requirements and ensure stable operation over networks of reasonably large scale. However, the processing power of routers has improved considerably over past several years. Furthermore, service providers generally limit the size of a single OSPF area. Hence, the OSPF protocol specifications, designed for large networks and slow routers, seem overly conservative for current networks. Hence, our focus has been to adapt the parameters of the OSPF protocol with the goal to achieve faster failure recovery.

## 7.2 Failure Detection and Recovery in OSPF

In OSPF, two adjacent routers in the same area periodically exchange *Hello* messages to maintain the link adjacency. If a router does not receive a Hello message from its neighbor within a *RouterDeadInterval* (typically 40 seconds or 4 *HelloIntervals*), it assumes the link between itself and the neighbor to be down and generates a new Router LSA to reflect the changed topology. All such LSAs, generated by the routers affected by the failure, are flooded throughout the network and cause the routers in the network to redo the shortest path first (SPF) calculation and update the next hop information in the forwarding table. Thus, the time required to recover from the failure consists of:

1. the failure detection time
2. LSA flooding time
3. the time to complete the new SPF calculations and update the forwarding tables.

With HelloInterval value 10 seconds and RouterDeadInterval value 40 seconds, the failure detection can take anywhere between 30 to 40 seconds. The LSA flooding times consist of the propagation delays and any pacing delays resulting from the rate-limiting of *LSUpdate* packets sent down an interface. Once a router receives a new LSA, it schedules an SPF calculation. Since SPF calculation using Dijkstra's algorithm [32] constitutes significant processing load, the router waits for some time (*spfDelay* - typically 5 seconds) to let other LSAs arrive before doing an SPF calculation. Moreover, the routers place a limit on the frequency of SPF calculations (governed by *spfHoldTime*, typically 10 seconds, between successive SPF calculations) which can introduce further delays. Tables 5.1 and 5.2 show different standard and vendor introduced delays that affect the OSPF operation in networks of popular commercial routers.

The focus of our investigation lies on reducing the failure detection time which is clearly the main component of the overall failure recovery time in OSPF based networks. While the availability of link layer notifications can help achieve fast failure detection, such mechanisms are often not available. Hence, the routers use the Hello protocol to detect the loss of adjacency with a neighbor. The Hello protocol operates via periodic exchange of Hello messages between neighbor routers. A router declares its adjacency with a neighbor to be down if it does not receive a Hello from the neighbor for more than RouterDeadInterval. This can happen if the link between the router and the neighbor is down or the neighbor router is no longer functional. To avoid a false breakdown of adjacency because of congestion related loss of Hello messages, the RouterDeadInterval is usually set to be four times the HelloInterval - the interval between successive Hello messages sent by a router to its neighbor. The failure detection via Hello protocol can be substantially speeded up by reducing the HelloInterval. However, there is a limit up to which the HelloInterval can be safely reduced. As the HelloInterval becomes smaller, there is an increased chance that the network congestion will lead to loss of several consecutive Hello messages and thereby cause false breakdown of adjacency between routers even though the routers and the link between them are functioning perfectly well. The LSAs generated because of a false alarm will lead to new path calculations, avoiding the supposedly down link, by all the routers in the network. A false alarm is soon corrected by successful Hello exchange between the affected routers which causes new set of LSAs to be generated and possibly new path calculations to be done by the routers in the network. Thus, false alarms cause unnecessary processing load on the routers and some times lead to temporary changes in the network traffic's path which can have a serious impact on the QOS levels in the network. If the false alarms are too frequent, the routers will have to spend a lot of time doing unnecessary

LSA processing and SPF calculations which may significantly delay important tasks such as Hello processing, thereby leading to more false alarms. Persistent overload on router CPUs will ultimately result in complete meltdown of routing operation in the network.

The objective of our investigation is to make a realistic assessment regarding how small the HelloInterval can be, to achieve faster detection and recovery from network failures while limiting the occurrence of false alarms. This assessment is done via simulations on the network topologies of commercial ISPs [91] using a detailed implementation of OSPF protocol in NS2 simulator [107] which models all the protocol features as well as various standard and vendor-introduced delays in the functioning of the protocol (Tables 5.1 and 5.2). We examine the network wide impact of reducing the HelloInterval in terms of number of false alarms under a realistic model of network congestion. We quantify the detrimental effect of these false alarms in terms of unnecessary SPF calculations done by the routers. We examine how the network topology influences the occurrence of false alarms. Finally, we evaluate how much does the faster detection of network failures help in achieving faster recovery from these failures in the operation of OSPF networks.

## 7.3 Related Work

In this section, we briefly survey the existing literature on speeding the recovery from network failures in the operation of OSPF and IS-IS protocols. First, we discuss the previous work on reducing the HelloInterval and the impact of congestion in causing false alarms. Alaettinoglu et al. [3] proposed reducing the HelloInterval to millisecond range to achieve sub-second recovery from network failures but did not consider any side effects of HelloInterval reduction. Shaikh et al. [129] used Markov Chain based analysis of a simple network topology to obtain the expected times before high packet drop rates cause a healthy adjacency to be declared down and then back up again. However, this work did not study the network wide generation of false alarms caused by congestion as the HelloInterval is reduced. Basu and Riecke [15] have also examined using sub-second HelloIntervals to achieve faster recovery from network failures. This work is similar to ours in the sense that it also considered the tradeoff between faster failure detection and increased frequency of false alarms. It reports 275ms to be an optimal value for HelloInterval providing fast failure detection while not resulting in too many false alarms. However, this work did not consider the impact of different levels of network congestion and topology characteristics on the optimal HelloInterval value. We believe these factors impact the setting of the HelloInterval substantially, as we illustrate later.

False alarms can also be generated if the Hello message gets queued behind a huge burst of LSAs and can not be processed in time. The possibility of such an event increases with reduction in RouterDeadInterval. Large LSA bursts can be caused by a number of factors such as simultaneous refresh of a large number of LSAs or several routers going down/coming up simultaneously. Choudhury et al. [26] studied this issue and observed that reducing the HelloInterval lowers the threshold (in terms of number of LSAs) at which an LSA burst will lead to generation of false alarms. However, the probability of such events is quite low. In our experiments with more probable events such as the failure of a single router, the resulting LSA burst was too small to cause false alarms. Similarly, we investigated if frequent update of Traffic Engineering LSAs [75] leads to large enough LSA bursts to cause false alarms. However, we did not observe any such effect even for reasonably high update frequency of such LSAs.

Since the loss and/or delayed processing of Hello messages can result in false alarms, recently there have been proposals to give such packets prioritized treatment at the router interface as well as in the CPU processing queue [26][10]. An additional proposal is to consider the receipt of any OSPF packet (e.g. an LSA) from a neighbor as an indication of the good health of the router's adjacency with the neighbor [10]. This provision can help avoid false loss of adjacency in the scenarios where Hello packets get dropped because of congestion, caused by a large LSA burst, on the control link between two routers. Such mechanisms should help mitigate the false alarm problem significantly. However, it will take some time before these mechanisms are standardized and widely deployed. Since SPF calculation using Dijkstra's algorithm imposes significant processing load on the routers, vendors have introduced delays (spfDelay and spfHoldTime) that limit the frequency of such operations. These delays ultimately result in slowing down the failure recovery process. Alaettinoglu et al. [2] propose eliminating any restrictions on SPF calculations. They argue that the frequency of SPF calculations can be reduced by careful filtering of status changes in the links/routers and the processing time of an SPF calculation can be reduced by using modern algorithms (such as [51][52][117]) instead of Dijkstra's algorithm. In a related work, Thorup [134] proposes the use of data structures that will help routers make a constant time determination of the next hop on the shortest path to a destination avoiding a given failed link. This will help in avoiding the routing loops while the routers recalculate shortest paths after a link failure.

## 7.4 Experimentation Methodology

We implemented substantial extensions to the OSPF routing model [109] currently available in NS2 simulator such as the Hello protocol, LSA generation and flooding, shortest path calculation and adjacency establishment. Our emphasis has been to include in the simulation model various standard (i.e., as per the OSPF specification [102]) and vendor-implemented delays and timers, listed in Tables 5.1 and 5.2, that affect the functioning of OSPF protocol in operational networks of commercial routers. Some of these delays are configurable, some have a fixed value and some depend on the architecture and processing capability of the routers. Values for the delays that depend on the architecture and processing capability of the routers were obtained after extensive experimentation with commercial routers [126][128]. In our experimentation, we used the standard or the typical values for the different delay parameters (except HelloInterval and RouterDeadInterval) as listed in Tables 5.1 and 5.2. This enables us to have a higher degree of confidence in the applicability of our simulation results to real operational networks.

Rather than using actual packet flows to create congestion, we used realistic models to achieve the same effects. This choice was driven by the lack of information about realistic traffic loads as well as a desire to keep the processing and running time of the simulations reasonable. The congestion model used in our simulations tries to emulate the behavior of *Random Early Drop* (RED) [47] and droptail buffer management policies. In RED, the packet drop probability ($p$) at a router interface increases linearly from a value 0 to *max_p* as the average buffer occupancy *qlen* (the ratio of the average queue length to the total buffer size) increases from *min_th* to *max_th*. The packet drop probability remains 0 for *qlen* values less than *min_th* and remains equal to *max_p* as the *qlen* becomes more than *max_th*. If *qlen* exceeds 1, the packet drop probability becomes 1, i.e., all the incoming packets are dropped. We simulate congestion by assigning random *qlen* values between 0 and *max_q* to the router interfaces. The assigned *qlen* value determines the packet loss probability for the OSPF packets arriving at the interface. The *qlen* value assigned to an interface persists for a random duration with in the range {*min_pers*, *max_pers*}. This is

| Topology | Nodes | Links | Topology | Nodes | Links |
|---|---|---|---|---|---|
| A | 9 | 72 | D | 37 | 88 |
| B | 27 | 58 | E | 51 | 176 |
| C | 27 | 116 | F | 116 | 476 |

Table 7.1: Network Topologies Used in Simulations

to emulate the slowly varying average queue length, an exponential moving average, in RED buffers. The *min_th*, *max_th* and *max_p* values used in the RED simulations are 0.25, 0.75 and 0.1 respectively. The congestion level in the simulation is controlled by parameter *max_q* and range $\{$*min_pers, max_pers*$\}$. As the value of *max_q* is increased, higher packet drop rates in the network become possible. The range $\{$*min_pers, max_pers*$\}$ will determine how long high (and low) packet drop rates persist on an interface.

A similar technique is used to emulate the behavior of droptail buffer management. For droptail buffers, *qlen* represents the ratio of instantaneous buffer occupancy and total buffer size. A new value is assigned to the *qlen* associated with each router interface every time a new OSPF packet arrives. The packet drop probability remains 0 unless *qlen* is greater than 1 in which case the packet drop probability is 1. Note that in the droptail simulations, *max_q* value needs to be greater than 1 for packet loss to occur and a given *max_q* value corresponds to the packet drop rate of $\frac{\mathrm{max\_q}-1}{\mathrm{max\_q}}$. The simulations were conducted on a number of topologies obtained from [91]. These topologies correspond to real IP backbones for several commercial ISPs. Table 7.1 lists some characteristics of these topologies. While most of the topologies are irregular, topology A is a pure mesh and topology B has a star-like structure.

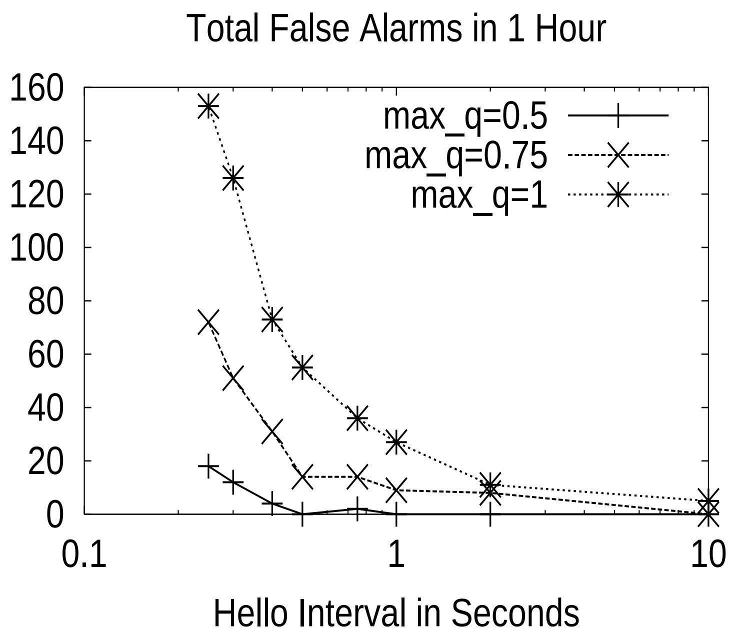


Figure 7.1: False alarm occurrence in topology C for different HelloInterval values and congestion levels (RED simulations, $\{$min_pers, max_pers$\}$ = $\{$0.1s,0.5s$\}$).

## 7.5 Simulation Results

The first set of simulation results examines how reducing the HelloInterval causes more false alarms to take place and how increase in network congestion exacerbates the problem. Figure 7.1 shows the total number of false alarms observed on topology C during 1 hour of failure-free operation for different HelloInterval values. These numbers were obtained from RED simulations assuming that average buffer occupancy persists for 100ms to 500ms. Different curves in the figure correspond to different congestion levels (modeled by parameter *max_q*). As expected, false alarms become more frequent with decrease in the HelloInterval value and increase in network congestion levels. Further, the impact of increased congestion levels seems to be more severe for lower HelloInterval values. Clearly,

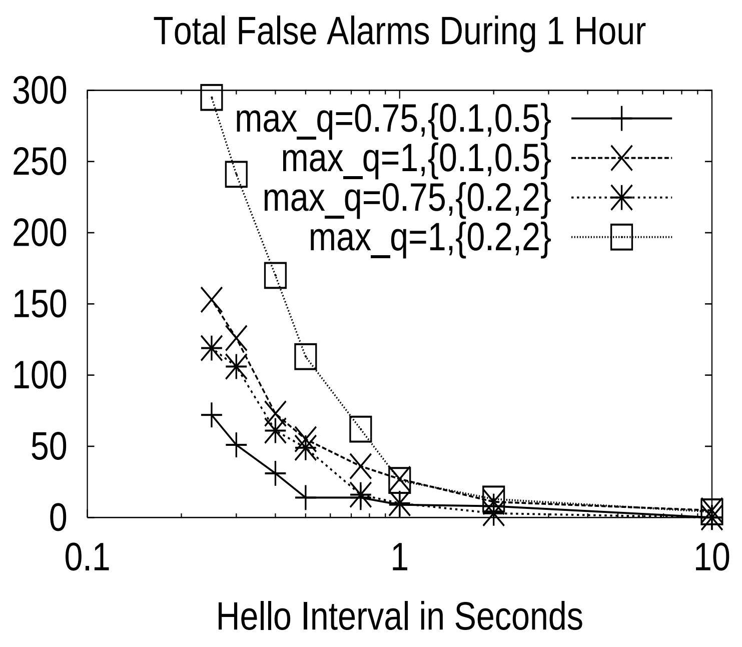


Figure 7.2: Change in false alarm frequency as high drop rates persist for longer durations . (RED simulations)

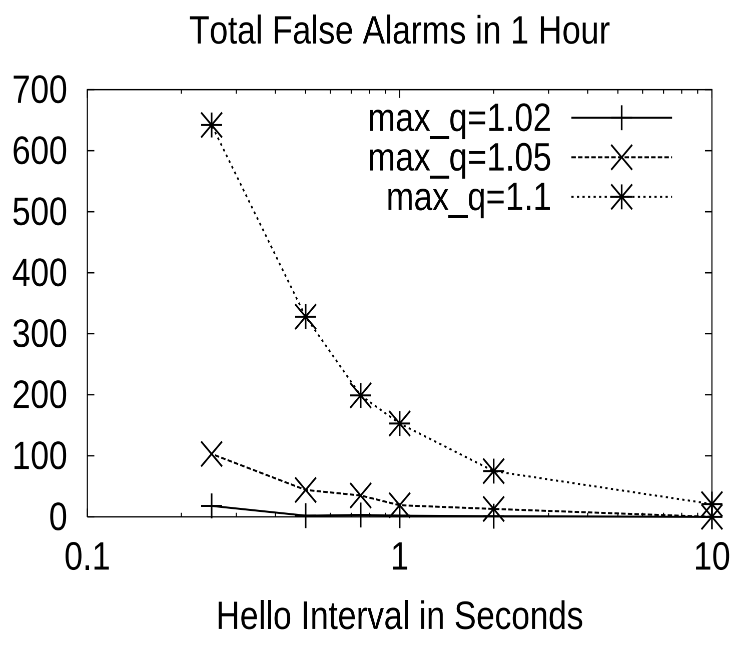


Figure 7.3: False alarm occurrence in topology C for droptail simulations.

the optimal value of HelloInterval depends on the expected congestion levels in the network and an understanding of what constitutes an acceptable limit on false alarm frequency. Assuming that no more than 20 false alarms in an hour can be tolerated and if the average buffer occupancy in the router interfaces will rarely rise above 0.5, the HelloInterval for topology C can be set to be 250ms. However, if the buffer overflows are not uncommon, it will be prudent not to reduce HelloInterval below 1.5 seconds. As shown in figure 7.2, if the congestion persists for longer durations (200ms to 2s, rather than 100ms to 500ms as in figure 7.1), the number of false alarms observed for a given HelloInterval increase further. Again, the increase in false alarms is more severe for lower HelloIntervals; hence there is a need to be conservative while setting HelloInterval value. The results for droptail simulations are shown in figure 7.3. The different curves in figure 7.3 show results for *max_q* values 1.02, 1.05 and 1.1 which correspond to packet drop rates of 1.96%, 4.76% and 9.09% respectively. Note that with around 10% overload on the system, any HelloInterval value less than 10s leads to unacceptable number of false alarms.

False alarms disrupt traffic in the network and cause unnecessary processing load on the routers. The LSAs generated as the result of a false alarm will be flooded throughout the network and lead to new SPF calculation by each router in the network. As the frequency of false alarms increases, routers spend more and more time doing unnecessary SPF calculations; generally one SPF calculation for each false alarm. Some times, for large HelloInterval values, a false alarm causes two SPF calculations to be done in each router; first one in response to adjacency breakdown and second one in response to re-establishment of adjacency following successful exchange of Hello messages between the routers affected by the false alarm. For smaller HelloInterval values, a broken adjacency is generally re-established soon enough so that only one SPF calculation (scheduled

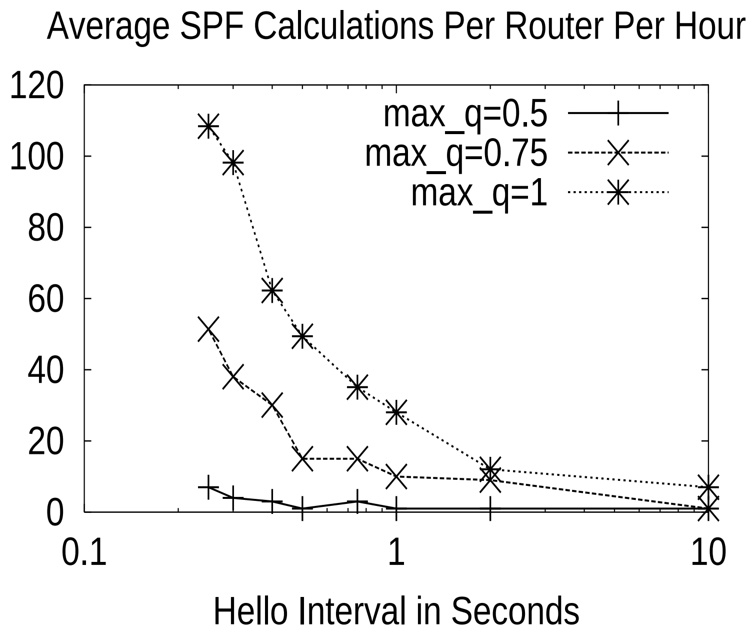


Figure 7.4: Average number of SPF calculations on a router in topology C due to false alarms shown in Figure 7.1 ( RED simulations, $\{$min_pers, max_pers$\}$ = $\{$0.1s,0.5s$\}$).

5 seconds, the spfDelay, after receiving the false alarm) is required to take care of both changes. Thus, for smaller HelloIntervals, since false alarms are corrected soon enough, they may not always lead to changes in routing tables and hence re-routing of network traffic. Nevertheless, smaller HelloInterval values do result in frequent false alarms and thus the processing load on the routers because of SPF calculations can become significant. Persistent overload on router CPUs can potentially lead to total meltdown of routing operation in the network. When the frequency of false alarms in the network becomes very high, spfDelay and spfHoldTime limit the frequency of SPF calculations. This and other previously mentioned effects can be seen in figure 7.4 which shows the average number of SPF calculations done by a router in topology C in response to false alarms in the simulations whose results regarding false alarms were previously shown in figure 7.1. The LSAs

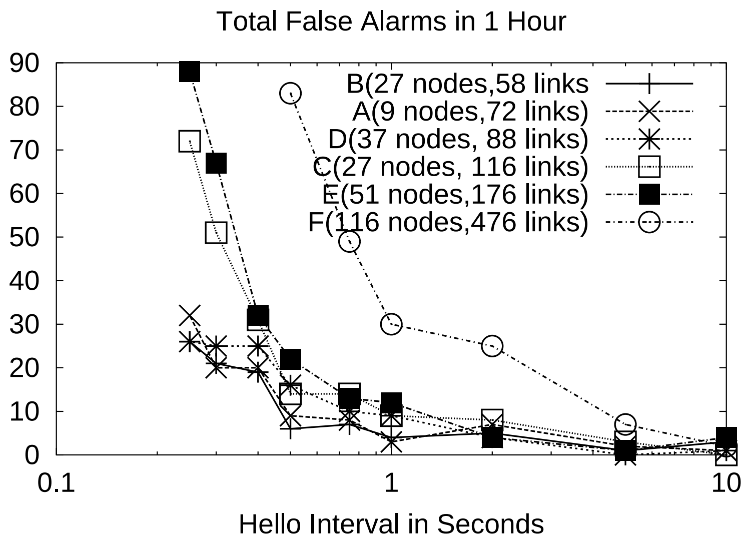
| Hello | Seed 1 | | Seed 2 | | Seed 3 | |
|---|---|---|---|---|---|---|
| Interval | FDT | FRT | FDT | FRT | FDT | FRT |
| 10s | 32.08s | 36.60s | 39.84s | 46.37s | 33.02s | 38.07s |
| 2s | 7.82s | 11.68s | 7.63s | 12.18s | 7.79s | 12.02s |
| 1s | 3.81s | 9.02s | 3.80s | 8.31s | 3.84s | 10.11s |
| 0.75s | 2.63s | 7.84s | 2.97s | 5.08s | 2.81s | 7.82s |
| 0.5s | 1.88s | 6.98s | 1.82s | 6.89s | 1.79s | 6.85s |
| 0.25s | 0.95s | 10.24s | 0.84s | 6.08s | 0.99s | 13.41s |

Table 7.2: Failure Detection Time (FDT) and Failure Recovery Time (FRT) For A Router Failure On Topology C With Different HelloInterval Values. (RED Simulations with *max_q* = 1 and {*min_pers*,*max_pers*} = {0.2s,2s}).

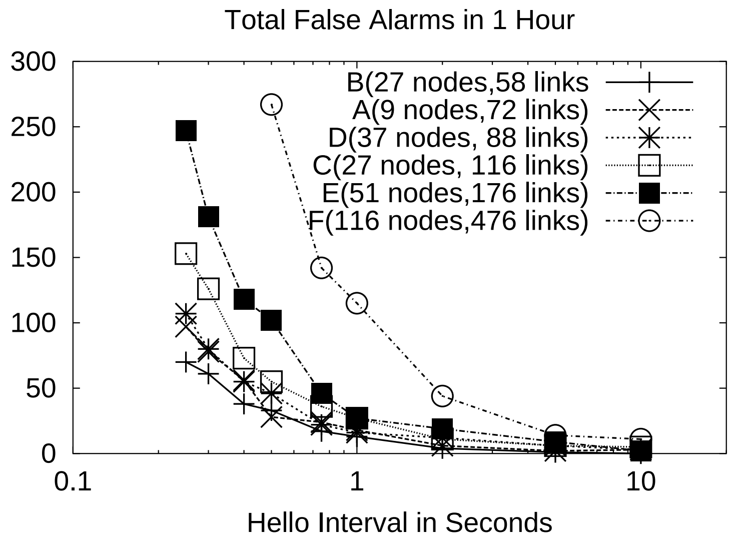
generated because of false alarms also impose unnecessary processing load on every router since each router may have to send and receive an LSA on each one of its interfaces as part of flooding of such LSAs.

Next, we examine the impact of topology characteristics on the optimal value of HelloInterval for a network. The probability of a false alarm occurring in the network increases with the number of links in the network. This trend is clear from figure 7.5 which shows the false alarm occurrence during 1 hour for different topologies for congestion levels created by *max_q* values 0.75 and 1 respectively. In figure 7.6, we plot the optimal HelloInterval value for different topologies assuming that no more than 20 false alarms per hour can be tolerated. It can be seen that the optimal HelloInterval value increases with the number of links in the topology. Further, as observed earlier, expected congestion level in the network plays a significant part in determining the optimal value.

Finally, we examine the impact of lower HelloInterval values on the failure detection and recovery times. For this purpose, we caused a particular router in topology C to fail and observed the failure detection time i.e. the time by when all the neighbors of the failed



(a) *max_q*=0.75



(b) *max_q*=1

Figure 7.5: False alarm occurrence in different topologies for different HelloInterval values and different congestion levels. ( RED simulations, {min_pers, max_pers} = {0.1s,0.5s})

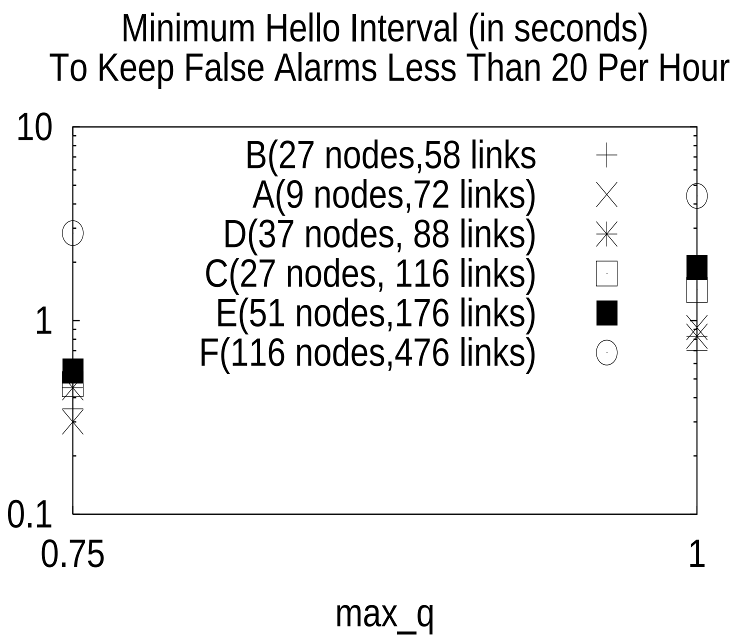


Figure 7.6: Optimal HelloInterval values for different topologies for different congestion levels. (RED simulations,{min_pers,max_pers}={0.1s,0.5s})

router have detected the failure and the failure recovery time i.e. the time by when all the routers in the network have finished SPF calculation and forwarding table update in response to the failure. The simulations used RED packet drop model with *max_q* value 1 and average buffer persistency in the range 0.2s to 2s. The simulations were conducted for several different seed values for the random number generation. In Table 7.2, we report some typical and interesting cases. As expected, the failure detection time is within the range 3 to 4 times the HelloInterval. Once a neighbor detects the router failure, it generates a new LSA about 0.5 seconds after the failure detection. The new LSA is flooded throughout the network and will lead to scheduling of SPF calculation 5 seconds (spfDelay) after the LSA receipt. This is done to allow one SPF calculation to take care of several new LSAs. Once the SPF calculation is done, the router takes about 200ms more to update the forwarding table. After including the LSA propagation and pacing delays, we can expect the failure recovery to take place about 6 seconds after the ’earliest’ failure detection by a neighbor router. Notice that many entries in Table 7.2 show the recovery to take place much sooner than 6 seconds. This is mainly because the reported failure detection times are the ’latest’ ones rather than the ’earliest’. In one interesting case (seed 2, HelloInterval 0.75s), the failure recovery takes place about 2 seconds after the ’latest’ failure detection. This happens because the SPF calculation scheduled by an earlier false alarm takes care of the LSAs generated because of router failure. Many times, the failure recovery can be noticed to take place much later than 6 seconds after the failure detection (notice entries for HelloInterval 0.25s, seeds 1 and 3). Failure recovery can be delayed because of several factors. The SPF calculation frequency of the routers is limited by spfHoldTime (typically 10s) which can delay the new SPF calculation in response to the router failure. The delay caused by spfDelay has already been explained. Finally, the routers with low connectivity may not get the LSAs in the first try because of loss due to congestion. Such routers may have to wait for 5 seconds (RxmtInterval) for the LSAs to be retransmitted. The results in Table 7.2 indicate that a smaller value of HelloInterval speeds up the failure detection but is not effective in reducing the failure recovery times beyond a limit because of other delays like spfDelay, spfHoldTime and RxmtInterval. While it may be possible to further speed up the failure recovery by reducing the values of these delays, eliminating such delays altogether may not be prudent. Eliminating spfDelay and spfHoldTime will result in several SPF calculations to take place in a router in response to a single failure (or false alarm) as different LSAs generated because of the failure arrive one by one at the router. The resulting overload on the router CPUs may have serious consequences for routing stability especially when there are several simultaneous changes in the network topology.

## 7.6 Summary

OSPF based recovery presents the most cost-effective way of providing survivability at the IP layer. However, the current standardized version of the OSPF protocol is unacceptably slow in terms of the speed of recovery. With the current default settings of the OSPF parameters, the network takes several tens of seconds before recovering from a failure. The main component in this delay is the time required to detect the failure using Hello protocol. Failure detection time can be speeded up by reducing the value of HelloInterval. However, too small a value of HelloInterval will lead to too many false alarms in the network which cause unnecessary routing changes and may even lead to routing instability. In our work, we explored the optimal value for the HelloInterval that will lead to fast failure detection in the network while keeping the false alarm occurrence within acceptable limits. Our simulation results indicate that the optimal value for HelloInterval for a network is strongly influenced by the expected congestion levels and the number of links in the topology. While the HelloInterval can be much lower than current default value of 10s, it is not advisable to reduce it to millisecond range as it will lead to too many false alarms. Further, it is difficult to prescribe a single HelloInterval value that will perform optimally in all cases. The network operator should set the HelloInterval conservatively taking in account both the expected congestion levels as well as the number of links in the network topology. The most important task for the future work is analyzing how to achieve still faster failure recovery, without compromising on routing stability, when failure detection is no longer an issue.

# CHAPTER 8

# CONCLUSIONS AND FUTURE WORK

## 8.1 Conclusions

Today, the Internet has become an integral part of our daily life. We have come to depend on Internet service as it provides us with easy and instantaneous access to information which is otherwise available only with significant investment of time and money. The incredible part is that, at the moment, the Internet revolution is only at its beginning stage. Future holds mind-boggling possibilities that will change our lives in manners beyond our imagination. The past few years have witnessed an important and perhaps fundamental change in the nature of Internet. Hitherto *best effort* Internet is getting ready to provide service guarantees. The paradigm shift from best effort to guaranteed quality is still in the progress. As a first stage, the Internet is moving towards a *class of service* architecture where the consumers will have a range of traffic classes, each providing a different statistically different service level, to choose from for their Internet service. Ultimately, the paradigm shift will be complete and the consumers will be able to negotiate a custom-built *service level agreement* with the Internet service providers. As the Internet grows in terms of usage and the complexity of the offered services, the need for evaluating the performance of the Internet as well as optimizing the performance where possible is becoming more critical. Evaluation and optimization of the performance of the Internet has come to be known as Internet traffic engineering. In this thesis, we have examined several aspects of the Internet traffic engineering problem.

Most of the previous literature dealing with Internet traffic engineering as viewed the problem as solely concerning the service providers. We have attempted to correct this anomaly by exploring the user perception of the Internet performance as well. Rather than examining how user application requirements translate into network QoS requirements, we examine how the Internet service available to the user translates in terms of user level application performance. We have argued that translation of network QoS into application level performance is more relevant than the translation other way around. Rather than measuring the application level performance, we have argued in favor of developing sound performance prediction frameworks as a scalable solution to the problem of translating network QoS into application level performance. One of the most important components for the proposed performance prediction framework is a sound analytical model for converting network QoS into a prediction for achieved TCP throughput. We have examined the difficulties associated with realizing a practical and reliable TCP throughput prediction model and presented a new analytical model which performs reasonably well in a vast range of scenarios even with the practical constraints on available input information.

The network performance from the point of view of a service provider is synonymous with the cost of operating the network while taking care of the QoS requirements of the consumers. Given the increasing dependence of the society on the Internet for a variety of purposes, network survivability, i.e., the ability of the network to survive failures without affecting the continuity and quality of the Internet service for the users, has emerged as a prime QoS requirement. Internet service providers have to consider network survivability while designing the network topology or deciding the protocols to be deployed in the network. The cost of survivability and the speed of recovery from the failures constitute two main aspects of the network survivability problem. In this thesis, we have analyzed both aspects of the problem. We have calculated the extra cost of the survivability against single node failures for several current IP backbone topologies for different survivability mechanisms. We have also examined the ways and means of reducing the cost of network operation while taking care of the survivability requirements. Our investigations into the speed of recovery from the failures cover several important aspects of the problem. We have investigated the network management aspect of the problem where the goal is to achieve fast detection of the failures by the network administrators so that they can take prompt actions to correct the problem. The other aspect of the problem deals with ensuring fast re-routing of the traffic affected by the failure while the failure has not yet been corrected. In this respect, we have analyzed the operation of cost-effective survivability mechanisms at the optical and IP layers with the goal of improving their speed of recovery behavior.

## 8.2 Future Work

In this thesis, we have explored several problems associated with Internet traffic engineering. Given the magnitude of the problems at hand, our efforts so far should be viewed as representing only a few steps towards the solutions. In this thesis, we propose the idea of building a performance prediction framework for scalable translation of network QoS in terms of the performance of user level applications. We have also worked on developing one important building block for the purpose - specifically translation of the network QoS into a prediction for TCP throughput. However, realizing the proposed performance prediction framework requires significantly more work. We need reliable, scalable and non-invasive techniques to estimate network level QoS as experienced by user level applications. Significant research effort will be required in the design of other building blocks of the framework. Finally, the framework needs to be implemented and tested for usefulness in a variety of different scenarios and for different applications. Implementation of a prototype and operational experience with it will undoubtedly reveal new research problems and provide further understanding into making the performance prediction framework a useful reality.

In our work on the cost of survivability, we observed that the comparative cost-effectiveness of OSPF and MPLS end-to-end recovery approach depends on the network topology. The network topology was also observed to be the main factor determining the cost of survivability regardless of the survivability mechanism used or the traffic matrix prevalent in the network. However, we do not yet have a complete understanding of why the topology has such a great significance. Understanding the impact of network topology on the cost of survivability constitutes the most important piece of future work in our opinion. Gaining such an understanding can have several beneficial consequences including designing the network topology in a manner that minimizes the cost of survivability for the particular survivability mechanism used. Also we need to have a better understanding of the impact of traffic matrix variations on the utility of the topological modifications designed to reduce the cost of survivability. Understandably, the impact of traffic matrix variation will become more clear once we have a good understanding of the impact of the network topology. However, it is very much possible that there are scenarios where traffic matrix plays a more dominant role than the topology. It is important to identify such scenarios and use them to gain an understanding of the impact of traffic matrix on the cost of survivability.

Similarly, there remains several unexplored dimensions of the speed of recovery problem. From the network administrator's perspective, it is not sufficient to track just IP layer failures. The failures occurring at layer 2 and below are also very important. In fact, many times, the failures occurring below the IP layer are responsible for the failures manifesting themselves at the IP layer. Thus, fast detection of layer 2 and layer 1 failures is as important as fast detection of IP layer failures. As in IP layer, there exists several mechanisms to choose from for layer2/1 failure detection. Comparing different alternative mechanisms with a comprehensive evaluation of their advantages and disadvantages constitutes an important research problem.

In our work on shared mesh restoration, we proposed several *aggregated signaling* schemes to allow fast failure recovery at the optical layer using shared mesh restoration. These schemes need to be implemented in software and tested for their performance on a working system. Prototype implementation and testing will not only help us have a better understanding of the advantages of such schemes but also it will encourage the industry to realize practical benefits of such schemes.

Finally, the problem of re-designing the OSPF protocol with the goal of improving the failure recovery times offers several avenues for further exploration. Our work so far has focussed on speeding up the failure detection time of the protocol, which is arguably the most time consuming factor in speeding up the overall failure recovery process. Adjusting different parameters of the OSPF protocol once failure detection time is no longer as issue remains an open research problem. The challenge is to do the adjustments without endangering the network stability. The other aspects of the OSPF protocol that demand careful attention include investigating the occurrence of routing loops while the protocol is recovering from a failure.

# APPENDIX A

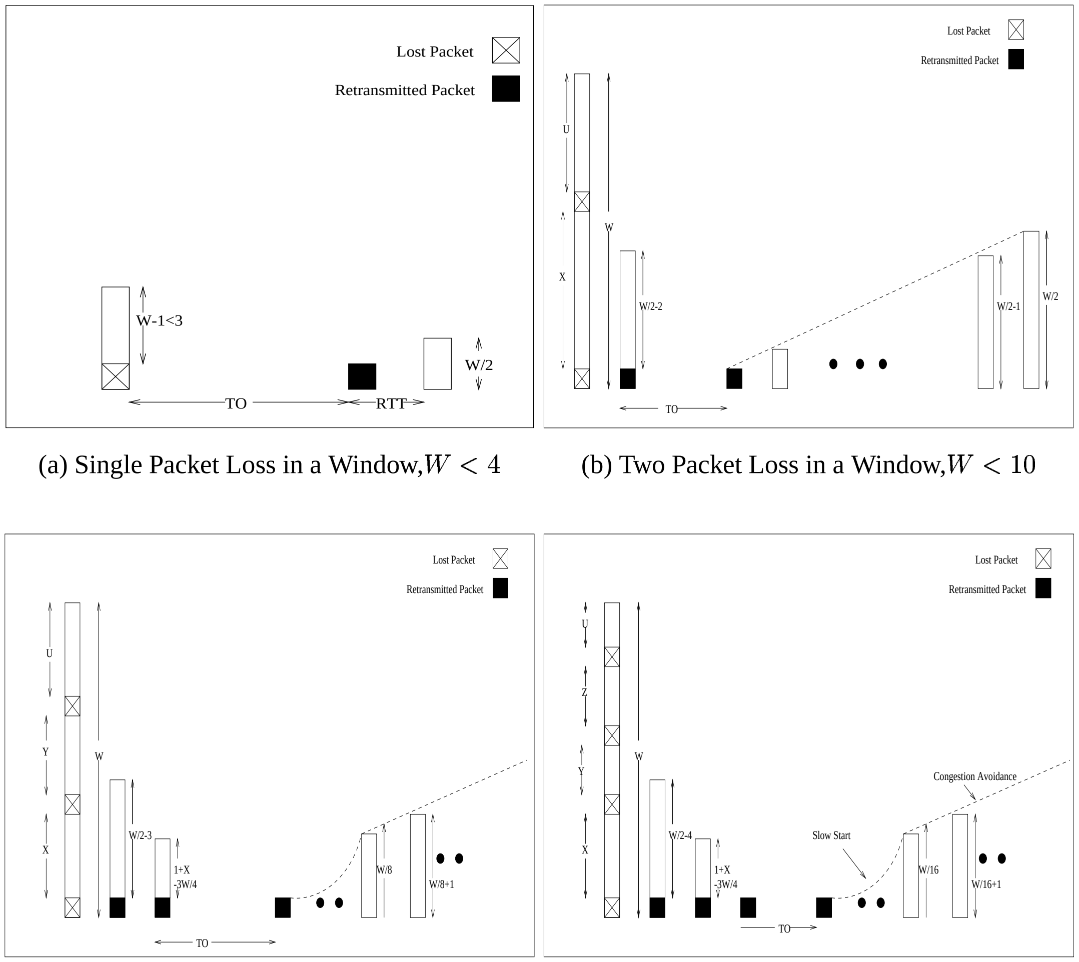
# RETRANSMISSION TIMEOUT SCENARIOS IN RENO TCP

In this appendix, we step through the possible loss scenarios for a TCP flow, in order to precisely identify those cases that will lead to a retransmission timeout.

A TCP flow detects packet losses either by receiving triple duplicate ACKs ($dupacks$) for the lost packet, or because a retransmission timeout timer expires before three $dupacks$ are received. When the timer expires, the packet is retransmitted. Whether a retransmission timeout occurs or not depends on the number of packets lost in the window and on the congestion window ($cwnd$) size at the time of the loss. In this appendix, we step through the different scenarios that trigger a retransmission timeout in a Reno TCP flow. This will help us determine the probability that an epoch ends with a timeout.

Let $W$ be the size of the $cwnd$ at the time the first packet is lost. After the first lost packet, a flow sends $W - 1$ more packets in response to acks of earlier successfully transmitted packets. If the $cwnd$ $W$ is less than 4, then even a single packet loss will lead to a timeout as sufficent number of packets are not transmitted after the lost packet to generate 3 $dupacks$ (Figure A.1a).

In the case where two packets are lost within a window of size $W$ (Figure A.1b), $W - 2$ $dupacks$ will be received for the first lost packet. Assuming that $W - 2$ $dupacks$ are sufficient to trigger the fast retransmit of the first lost packet, the $cwnd$ will be reduced



(a) Single Packet Loss in a Window,$W < 4$

(b) Two Packet Loss in a Window,$W < 10$

(c) Three Packet Loss in a Window,$W < 20$

(d) Four Packet Loss in a Window

Figure A.1: Retransmission timeout scenarios.

to $W/2$. Since $W - 2$ $dupacks$ are received for the first lost packet, the $cwnd$ will be inflated to $W/2 + W - 2$. At the time the first lost packet is retransmitted $W$ packets stand unacknowledged. Thus, in the fast recovery phase subsequent to the fast retransmit of the first lost packet, the flow will send $W/2 - 2$ packets. These packets constitute the $dupacks$ for the second lost packet. In case $W < 10$, $W/2 - 2$ $dupacks$ won't be sufficient to trigger the fast retransmit of the second lost packet and a timeout will occur. Thus, if $W < 10$, a loss of two packets within the window leads to retransmission timeout.

In the case of 3 losses in a window (Figure A.1c), $W - 3$ $dupacks$ will be received for the first lost packet. Assuming that $W - 3$ $dupacks$ are sufficient to avoid a retransmission time out and successfully trigger retransmission of the first lost packet, the $cwnd$ will be reduced to $W/2$ and $W/2 - 3$ packets will be sent in the accompanying fast recovery phase. These packets in turn generate $W/2 - 3$ $dupacks$ for the second lost packet. Further assuming that these $dupacks$ are also sufficient to trigger the fast retransmit of the second lost packet, the $cwnd$ will be reduced to $W/4$. This congestion window is then inflated by the $W/2 - 3$ $dupacks$ received for the second lost packet, and reaches $W/4 + W/2 - 3$. However the number of unacknowledged packets at this stage is $W - X - 1 + W/2 - 3$ where $X$ is the number of successfully transmitted packets between the first and second lost packets. Therefore, in the fast recovery phase following the retransmission of the second lost packet $1 + X - 3W/4$ packets will be sent. As the maximum value of $X$ is $W - 3$, a maximum of $W/4 - 2$ packets can be sent in the fast recovery phase following the retransmission of the second lost packet. These packets constitute $dupacks$ for the third lost packet. If $W$ is less than 20, there won't be enough $dupacks$ to trigger a fast retransmit of the third lost packet and hence a timeout will occur. Since $W \geq 10$ (otherwise the flow would have timed out for the second lost packet itself), the minimum value of $X$ required to get 3 $dupacks$ for the

3rd lost packet is $2 + 3W/4 = 9.5$. Hence, with three packet losses in a window, a timeout will always occur if the number of successfully transmitted packets between the first and second lost packets is less than 10.

Finally, let us consider the case of the loss of 4 packets in a window (Figure A.1d). Let us assume that $1 + X - 3W/4$ $dupacks$ are sufficient to trigger the fast retransmit of the third lost packet, so that the $cwnd$ is reduced to $W/8$. $1 + X - 3W/4$ duplicate acks will then inflate this $cwnd$ to $W/8 + 1 + X - 3W/4$ ($= A$). However, the number of unacknowledged packets at this point is $U + Z + 2 + W/2 - 4 + 1 + X - 3W/4$ ($= B$), where $Z$ is the number of successfully transmitted packets between the third and fourth lost packets and $U$ is the number of successfully transmitted packets between the fourth lost packet and the end of the corresponding round (Figure A.1d). Thus, the number of packets sent in the fast recovery phase after the retransmission of the third lost packet is $A - B = 2 - 3W/8 - U - Z$. Since, the minimum values of $U$ and $Z$ is zero and $W \geq 20$, the quantity $2 - 3W/8 - U - Z$ is always negative and hence no packet is sent in the fast recovery phase after the retransmission of the third lost packet. Thus, no duplicate ack is received for the fourth lost packet and a timeout occurs. Therefore, when 4 or more packets are lost in a window, a retransmission timeout *always* occurs.

In our analysis, we will assume for simplicity that a timeout always occurs in case 3 or more packets are lost in a window. Let $P_{TO}(W)$ denote then the timeout probability, when $W$ is the $cwnd$ size at the time of the packet loss, and let $P(n, W)$ be the probability of losing $n$ out of $W$ packets starting from the first lost packet for a given loss model. We then have:

$$P_{TO}(W) = \begin{cases} 1 & \text{if } W < 4 \\ 1 - P(1, W) & \text{if } W < 10 \\ 1 - P(1, W) - P(2, W) & \text{otherwise} \end{cases} \tag{A.1}$$

# APPENDIX B

# CONGESTION WINDOW AT THE BEGINNING OF AN EPOCH

The congestion window ($cwnd$) at the beginning of an epoch depends on how many times the *ssthresh* was halved in the previous epoch. The events causing *ssthresh* to be halved include *fast retransmission* of a lost packet ($retrans$) and the *retransmission timeout* ($rto$). Let us assume that $n$ packets are lost within a window of $W$ packets at the end of an epoch. If $W < 4$, there are not sufficient duplicate acks ($dupacks$) to trigger the $retrans$ of the first lost packet. Hence the $rto$ takes place, *ssthresh* will be set to $W/2$ and after the slow start phase, the $cwnd$ at the beginning of the next epoch will be $W/2$. Otherwise, if $n \geq W - 2$, again there are not sufficient $dupacks$ to trigger the $retrans$ of the first lost packet and hence $rto$ will occur and the $cwnd$ at the beginning of the next epoch will be $W/2$. Let $Q(i, W)$ be the probability that $cwnd$ at the beginning of the next epoch will be $W/2^i$ given that $W$ is the $cwnd$ size at the time of packet loss in the previous epoch. Then,

$$Q(1, W) = \begin{cases} 1 & \text{if } W < 4 \\ P(1, W) + \sum_{n=W-2}^{W} P(n, W) & \text{otherwise} \end{cases} \tag{B.1}$$

We have seen in Appendix A that if $W < 10$ and there are sufficient $dupacks$ to $retrans$ the first lost packet, the flow will still $rto$ before retransmitting the second lost packet. In this case, the $cwnd$ at the beginning of the next epoch will be $W/4$. Hence, for $W < 10$, $Q(2, W) = 1 - Q(1, W)$. If $W \geq 10$, the $cwnd$ at the end of the loss recovery will be

$W/4$, if 2 packets are lost or if there are not sufficient $dupacks$ to trigger the $retrans$ of the second lost packet. After the $retrans$ of the first lost packet, the flow sends $W/2 - n$ packets in the fast recovery phase. If $W/2 - n < 3$ i.e. $n > W/2 - 3$, the second lost packet will be retransmitted via $rto$. Note that $n$ should be less than or equal to $W - 3$, otherwise even the first lost packet will be retransmitted via $rto$. Thus,

$$Q(2, W) = \begin{cases} 1 - Q(1, W) & \text{if } W < 10 \\ P(2, W) + \sum_{n=W/2-2}^{W-3} P(n, W) & \text{otherwise} \end{cases} \tag{B.2}$$

The terms $\sum_{n=W-2}^{W} P(n, W)$ and $\sum_{n=W/2-2}^{W-3} P(n, W)$ in $Q(1, W)$ and $Q(2, W)$ respectively make the evaluation of $Q(1, W)$ and $Q(2, W)$ cumbersome. Further, it becomes difficult to prove the suitability of different loss models from the point of view of uniquely determining $cwnd$ value at the end of an epoch for a given loss rate. Thus, for simplicity we assume that

$$Q(1, W) = \begin{cases} 1 & \text{if } W < 4 \\ P(1, W) & \text{otherwise} \end{cases} \tag{B.3}$$

$$Q(2, W) = \begin{cases} 1 - Q(1, W) & \text{if } W < 10 \\ P(2, W) & \text{otherwise} \end{cases} \tag{B.4}$$

Since, we assume that three packets lost within a window always cause $rto$, the $cwnd$ at the beginning of the next epoch is never less than $W/8$ and hence $Q(3, W) = 1 - Q(1, W) - Q(2, W)$. As per the cyclical model, the steady state of the flow is characterized by a sequence of epochs with same initial and final window sizes. Thus, if $W_f$ is the $cwnd$ at the end of an epoch in steady state, $W_i$ is given by:

$$W_i = \sum_{m=1}^{3} Q(m, W_f) \frac{W_f}{2^m} \tag{B.5}$$

# APPENDIX C

## UNIQUENESS OF $W_F$ FOR A GIVEN $P$ FOR RANDOM LOSS MODEL

We first determine the conditions that guarantee the uniqueness of $W_f$ for a given $p$ as determined by equation 3.29 which is repeated next:

$$W_f = \frac{b}{2}(W_f + W_i)(W_f - W_i + 1)(1 - (1-p)^{W_f}) \tag{C.1}$$

It is clear that the *left hand side* ($W_f$) and factor $(1 - (1-p)^{W_f})$ of the *right hand side* of the equation above increase monotonically with $W_f$. If it can be shown that the *right hand side* as a whole also increases monotonically with $W_f$, the uniqueness of $W_f$ for a given $p$ is guaranteed. Note that in our model, $W_f$ takes only positive integer values. Thus, our goal is to show that $\frac{b}{2}(W_f + W_i)(W_f - W_i + 1)$ (say $A$) increases monotonically for integer values of $W_f$. Note that $A$ represents the number of packets sent in the congestion avoidance phase excluding the packets sent in the 'short' round. Let the value of $W_f$ be increased from $W_{f1}$ to $W_{f2}$. Let $W_{i1}$ and $W_{i2}$ be the corresponding values of $W_i$ and $A1$ and $A2$ be the corresponding values of $A$. There are two possibilities: $W_{i2} \leq W_{i1}$ and $W_{i2} > W_{i1}$. As the rate of increase in window size is constant at 1 packet per $b$ rounds, the two scenarios can be depicted as in figure C.1. It is clear that if $W_{i2} \leq W_{i1}$, $A2$ is larger than $A1$. In the case $W_{i2} > W_{i1}$, in order for $A2$ to be greater than $A1$,

$$\frac{b}{2}(W_{f2}+W_{i2})(W_{f2}-W_{i2}+1) > \frac{b}{2}(W_{f1}+W_{i1})(W_{f1}-W_{i1}+1) \tag{C.2}$$

Since $W_{f2} > W_{f1}$ and $W_{i2} > W_{i1}$, the relation above will hold if

$$\begin{aligned} W_{f2}-W_{i2} &> W_{f1}-W_{i1} && \text{(C.3)}\\ \Rightarrow W_{f2}-W_{f1} &> W_{i2}-W_{i1} && \text{(C.4)} \end{aligned}$$

i.e. $A$ will increase monotonically with $W_f$ if it can be proved that the increase in $W_i$ is less than the increase in $W_f$. Plugging in the values of $W_{i2}$ and $W_{i1}$ using equation 3.21, we get

$$W_{f2}-W_{f1} > \sum_{m=1}^{3}(Q(m,W_{f2})\frac{W_{f2}}{2^m} - Q(m,W_{f1})\frac{W_{f1}}{2^m}) \tag{C.5}$$

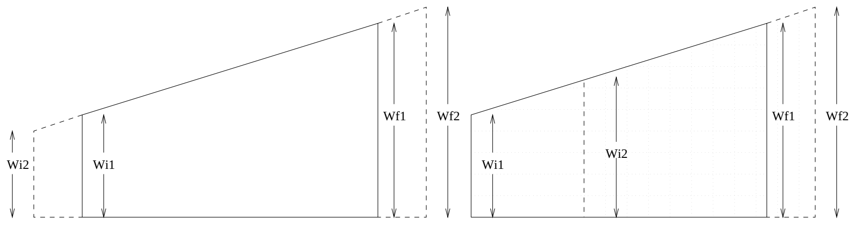
Let $W_{f2}-W_{f1} = \delta$. Thus

$$\begin{aligned} \delta &> \sum_{m=1}^{3}(Q(m,W_{f2})\frac{W_{f1}+\delta}{2^m} - Q(m,W_{f1})\frac{W_{f1}}{2^m}) && \text{(C.6)}\\ \Rightarrow \delta &> \sum_{m=1}^{3}((Q(m,W_{f2})-Q(m,W_{f1}))\frac{W_{f1}}{2^m} + Q(m,W_{f2})\frac{\delta}{2^m}) && \text{(C.7)} \end{aligned}$$

Let $Q(m,W_{f2})-Q(m,W_{f1}) = \alpha_m$.

$$\begin{aligned} \delta &> \sum_{m=1}^{3}((\alpha_m)\frac{W_{f1}}{2^m} + Q(m,W_{f2})\frac{\delta}{2^m}) && \text{(C.8)}\\ \Rightarrow \sum_{m=1}^{3}\alpha_m\frac{W_{f1}}{2^m} &< \delta(1-\sum_{m=1}^{3}Q(m,W_{f2})\frac{1}{2^m}) && \text{(C.9)} \end{aligned}$$

Since $\delta(1-\sum_{m=1}^{3}Q(m,W_{f2})\frac{1}{2^m}) > 0$, the relation above will still be true if



(a) $W_{i2} \leq W_{i1}$ (b) $W_{i2} > W_{i1}$

Figure C.1: As $W_f$ is increased, $W_i$ can increase or decrease

$$\sum_{m=1}^{3} \alpha_m \frac{W_{f1}}{2^m} < 0 \tag{C.10}$$

The relation above constitutes a sufficient (but not necessary) condition to ensure that $W_f$ obtained for a given $p$ using equation 3.29 is unique. Next, we show that random loss model satisfies this condition.

## C.1 Proving uniqueness of $W_f$ for a given $p$ in random loss model

For random loss model, $P(n, W)$, the probability of losing $n$ packets out of a window of $W$ packets starting with the first lost packet is given by equation 3.16. Let $Q(i, W)$ be the probability that congestion window at the beginning of the next epoch will be $W/2^i$ given that $W$ is the congestion window size at the time of packet loss in the previous epoch. Using equation 3.16, $Q(i, W)$ terms for random loss model are given by:

$$Q(1, W) = \begin{cases} 1 & \text{if } W < 4 \\ (1-p)^{W-1} & \text{otherwise} \end{cases} \tag{C.11}$$

$$Q(2, W) = \begin{cases} 1 - Q(1, W) & \text{if } W < 10 \\ (W-1)p(1-p)^{W-2} & \text{otherwise} \end{cases} \tag{C.12}$$

and

$$Q(3,W) = 1 - Q(1,W) - Q(2,W) \tag{C.13}$$

As mentioned earlier, $\alpha_m$ is $Q(m, W_{f2}) - Q(m, W_{f1})$ where $W_{f2} > W_{f1}$. Note that $Q(1,W)$ is a monotonically decreasing function of $W$. Thus $\alpha_1 = Q(1, W_{f2}) - Q(1, W_{f1})$ is always negative. Since $\alpha_1 + \alpha_2 + \alpha_3 = 0$, $\alpha_2 + \alpha_3 > 0$ and the condition C.10 can be written as

$$\begin{aligned}
\alpha_1 \frac{W}{2} + \alpha_2 \frac{W}{4} + \alpha_3 \frac{W}{8} &< 0 \\
\Rightarrow \alpha_2 (\frac{W}{4} - \frac{W}{2}) + \alpha_3 (\frac{W}{8} - \frac{W}{2}) &< 0 \\
\Rightarrow (\alpha_2 + \alpha_3)(\frac{W}{4} - \frac{W}{2}) - \alpha_3 \frac{W}{8} &< 0 \\
\Rightarrow -(\alpha_2 + \alpha_3)\frac{W}{4} - \alpha_3 \frac{W}{8} &< 0 \\
\Rightarrow (\alpha_2 + \alpha_3)\frac{W}{4} + \alpha_3 \frac{W}{8} &> 0
\end{aligned}$$

Since $\alpha_2 + \alpha_3 > 0$, the condition above will be satisfied if $\alpha_3 \geq 0$. In the following, we prove that $\alpha_3 \geq 0$ as $W_f$ is increased from $W$ to $W + 1$. We start with the assumption that $\alpha_3 < 0$.

$$\begin{aligned}
\alpha_3 &< 0 \\
\Rightarrow -\alpha_1 - \alpha_2 &< 0 \\
\Rightarrow \alpha_1 + \alpha_2 &> 0 \\
\Rightarrow Q(1, W+1) - Q(1,W) + Q(2, W+1) - Q(2,W) &> 0 \qquad \text{(C.14)}
\end{aligned}$$

which means that, for $W + 1 < 10$,

$$\begin{aligned}
Q(1, W+1) - Q(1, W) + 1 - Q(1, W+1) - 1 + Q(1, W) &> 0 \\
\Rightarrow 0 &> 0 \\
\Rightarrow \quad &False
\end{aligned}$$

For $W + 1 = 10$, Equation C.14 can be written as

$$\begin{aligned}
Q(1, W+1) - Q(1, W) + Q(2, W+1) - 1 + Q(1, W) &> 0 \\
\Rightarrow Q(1, W+1) + Q(2, W+1) &> 1 \\
\Rightarrow \quad &False
\end{aligned}$$

For other values of $W$, Equation C.14 can be written as

$$\begin{aligned}
(1-p)^W - (1-p)^{W-1} + Wp(1-p)^{W-1} - (W-1)p(1-p)^{W-2} &> 0 \\
\Rightarrow (1-p)^2 - (1-p) + Wp(1-p) - (W-1)p &> 0 \\
\Rightarrow -(1-p)p + Wp(1-p) - (W-1)p &> 0 \\
\Rightarrow -(1-p) + W(1-p) - (W-1) &> 0 \\
\Rightarrow (W-1)(1-p) - (W-1) &> 0 \\
\Rightarrow 1 - p - 1 &> 0 \\
\Rightarrow p &< 0 \\
\Rightarrow \quad &False
\end{aligned}$$

Thus, we have shown that each time $W_f$ is incremented by 1, the increase in $W_i$ is less than 1 and hence the $W_f$ determined from equation 3.29 is unique for a given $p$.

# APPENDIX D

# NON-INVASIVE ESTIMATION OF CORRELATION IN LOSSES AND THROUGHPUT PREDICTION BASED ON A CORRELATED LOSS MODEL

In this appendix, we explore the problems of estimating the “correlation” in packet losses using non-invasive means and also develop a throughput prediction model that uses the information about correlation in packet losses. In the results presented earlier, we have seen that throughput prediction using basic average loss rate does not always perform very well, especially in the scenarios where significant “correlation” among packet losses is expected. Developing a throughput prediction model that uses this “correlation” information can perform better in these scenarios. However, the complications emerge as we consider the logistic of estimating the “correlation” in packet losses in a non-invasive manner. It is not clear how should “correlation” metric values observed at individual routers be combined to produce end-to-end value. In the absence of an answer, we are reduced to using only end-to-end ping probes for estimating the correlation in packet losses. Still, one has to devise a manner for doing the estimation. Infact, the limitations imposed by non-invasive estimation means should determine the correlated loss model used for throughput prediction. Typical models assume the network to toggle between *non-congested* and *congested* states which are characterized by different loss rates and durations. The popular choices are to assume that the duration of the *non-congested* state of the network is a function of the prevalent loss probability ($p$) and the first packet loss event causes the network to toggle to the *congested* state which lasts for an assumed duration (say one RTT or for remaining "window") and is characterized by a different higher loss probability ($p'$). Limiting ourselves to non-invasive estimation means, we developed a correlated loss model that does not depend on any "flow-level" characteristics. The basic feature of the model is that the applicable loss probability for a packet is determined by the fate of the previous packet, i.e., if the previous packet was dropped, the applicable loss rate for the current packet is higher than otherwise. As presented in the next section, the loss parameters $p$ and $p'$ for the model can be estimated with ping probes. This is followed by the incorporation of the correlated loss model in the throughput prediction framework developed earlier. Finally, we compare the performance of the correlated loss model with that of random loss model in terms of throughput prediction with actual loss rates.

## D.1 Estimating "Correlated" Loss Parameters

Estimating the parameters ($p$ and $p'$) of the correlated loss model can be carried out using the *three counter technique* discussed next. As the name indicates, it relies on three counters: $success$, $p_{dropped}$ and $p'_{dropped}$. For ping probe based estimation, the counters are updated as follows: the $success$ counter is incremented by 1 for each acked ping probe; if $n$ probes are lost between two successfully transmitted ones, the $p_{dropped}$ counter is incremented by 1 and the $p'_{dropped}$ counter is incremented by $n-1$. Values for the loss parameters $p$ and $p'$ can then be computed from the following two expressions:

$$p = \frac{p_{dropped}}{success} \tag{D.1}$$

$$p' = \frac{p'_{dropped}}{p_{dropped} + p'_{dropped}} \tag{D.2}$$

Note that $p'$ should be greater than $p$, and if the computed value does not satisfy this relation (this can happen due to the small number of samples involved), the transient congestion model is then defaulted to the random loss model, i.e., $p = p'$.

Figures D.1 and D.2 show the simulation results regarding the accuracy of estimates obtained using *three counter technique* with *back-to-back* and *ack-paced* probing. The actual values for $p$ and $p'$ were obtained by using the *three counter technique* at the destination of the TCP flows and taking the average of the values thus obtained across all the flows in a group. As in our earlier presented results, the probing application used in these simulations, sent 4 probes (of same size as the packets of TCP flows) in each round of probing and was frequent enough to consume 0.5% of the bottleneck link bandwidth. These results are representative results in the sense that varying the number of probes per round or the probing frequency did not result in significant improvements. The results shown for single router, heterogeneous flow simulations refer to the set of flows with larger RTT. It is clear from the figures that the estimates obtained for $p$ and $p'$ with ping probing are in general not good. This is especially true for droptail scenarios where one expects to find significant correlation in packet losses and hence where the correlated loss model will be really useful. On examining the curves for actual average values for $p$ and $p'$, we find that the difference between $p$ and $p'$ values becomes significant as the bottleneck bandwidth increases. This is understandable as with higher bandwidth, the TCP flows will be characterized by bigger packet bursts. This will happen because larger bottleneck bandwidth means decreased spacing between the acks and hence between data packets and increased window sizes (since more bandwidth is available per flow). The contrasts are particularly visible for droptail scenarios where with increase in bottleneck bandwidth, the $p$ values decrease rapidly but the $p'$ values remain significantly high. Going back to figure 3.12 which shows the performance of throughput prediction based on random loss model with actual loss rates, we observe that the deterioration in the performance of the model can be linked to the increased correlation in packet losses. However, it can be argued that a TCP flow on Internet typically has small window sizes and hence might not see much correlation in packet losses. The bigger unanswered question is even if significant correlation exists, how to estimate it in a non-invasive and scalable manner. Without any answer to this question, the benefits of developing a throughput prediction model that takes cognizance of the correlation in packet losses remain limited.

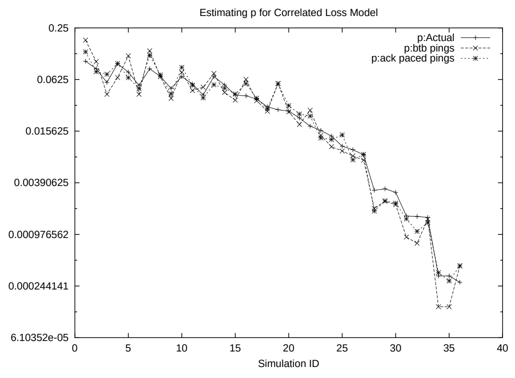
Having presented a correlated loss model for which the non-invasive estimation is possible atleast in theory, we move on to incorporate the model in the throughput prediction framework developed in Section 3.5.

## D.2 Throughput Prediction using a *Correlated* Loss Model

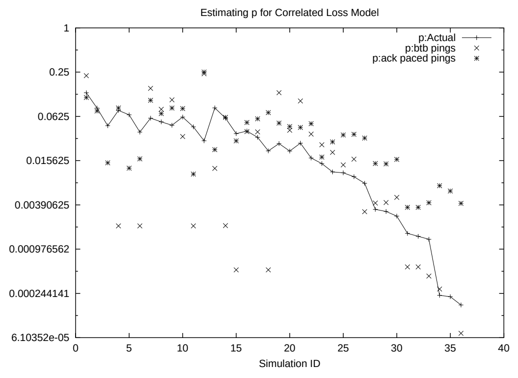
It turns out that it is surprisingly easy to replace the random loss model used in our throughput prediction framework with the correlated loss model presented above. The whole transformation can be performed with following modifications:

1. Replace average loss rate used in the framework with the loss probability in effect during the *non-congested* state of the network.

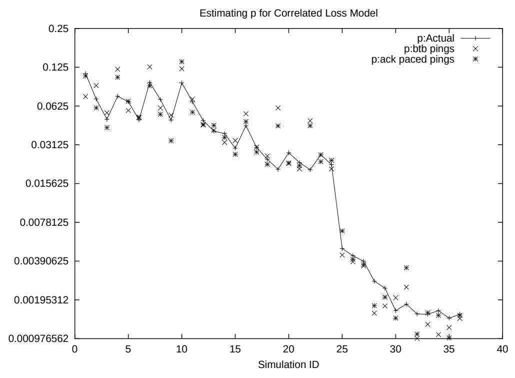
2. Derive the expressions for the probability $P(i, W)$, that after the first packet loss in an epoch, a total of $i$ packets including the first loss are lost in the following window of size $W$, under the new model and use it in the throughput prediction framework



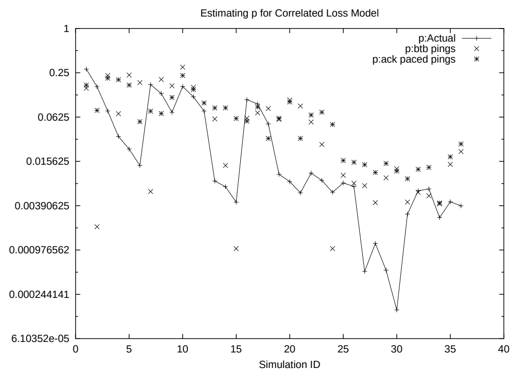
(a) Single RED router, homogeneous flows



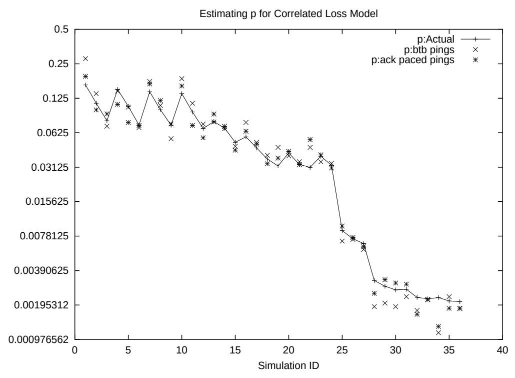
(b) Single Droptail router, homogeneous flows



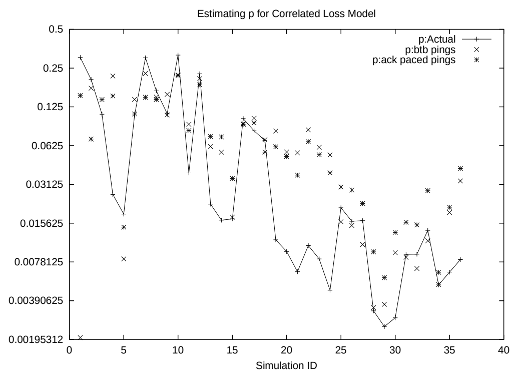
(c) Single RED router, heterogeneous flows



(d) Single Droptail router, heterogeneous flows

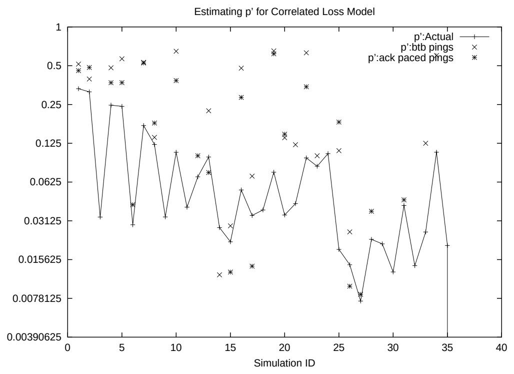


(e) Two RED routers

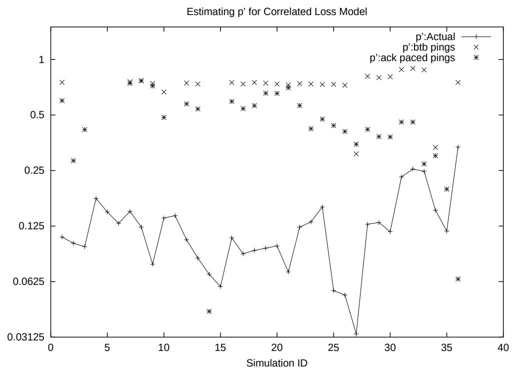


(f) Two Droptail routers

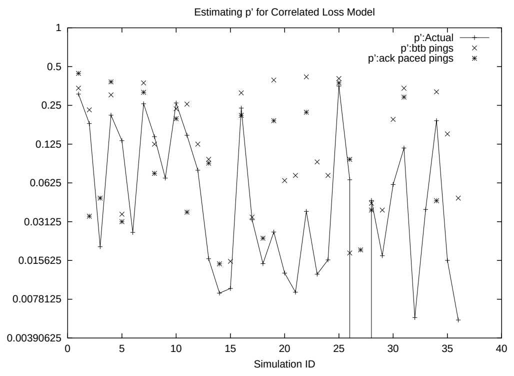
Figure D.1: Estimating $p$ for the correlated loss model



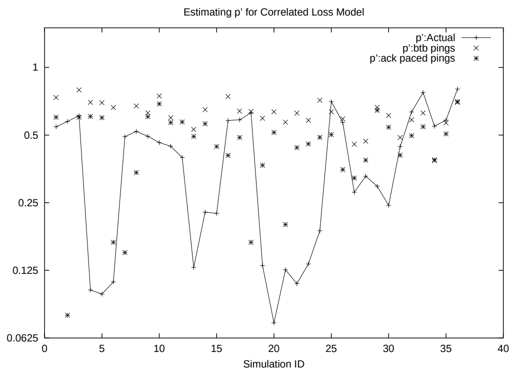
(a) Single RED router, homogeneous flows



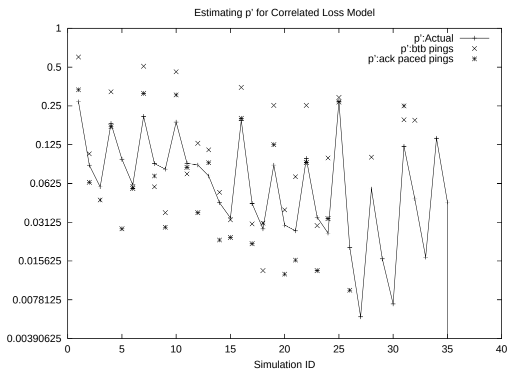
(b) Single Droptail router, homogeneous flows



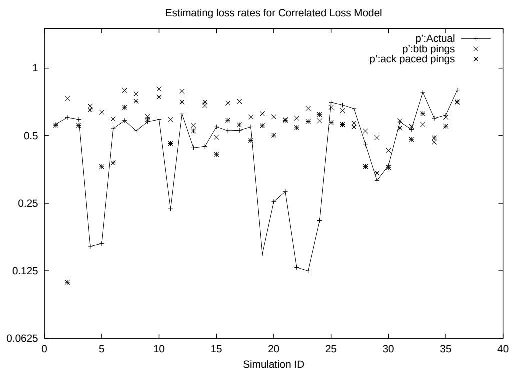
(c) Single RED router, heterogeneous flows



(d) Single Droptail router, heterogeneous flows



(e) Two RED routers



(f) Two Droptail routers

Figure D.2: Estimating $p'$ for the correlated loss model

in place of equation 3.16. Further it suffices to provide expressions for $P(1, W)$ and $P(2, W)$ only as these are the only ones used in the throughput prediction procedure.

3. Prove that the value of $W_f$ obtained with equation 3.29 is still unique for a give $p$ (the loss probability during the *non-congested* state) under the new model.

In the correlated loss model we consider, a packet is lost with probability $p'$ $(p)$, if the previous packet from the flow was (not) lost. In this case, the probability $P(i, W), i = 1, 2$ is given by:

$$P(1, W) = \begin{cases} 1 & \text{if } W = 1 \\ (1-p')(1-p)^{W-2} & \text{otherwise} \end{cases} \tag{D.3}$$

$$P(2, W) = \begin{cases} 0 & \text{if } W = 1 \\ p' & \text{if } W = 2 \\ (1-p')(p+p') & \text{if } W = 3 \\ (p'+p)(1-p')(1-p)^{W-3} + (W-3)p(1-p')^2(1-p)^{W-4} & \text{otherwise} \end{cases} \tag{D.4}$$

Next we prove the uniqueness of $W_f$ obtained with equation 3.29 for a give $p$ (the loss probability during the *non-congested* state) under the new model. For this purpose, we will show that the condition mentioned in equation C.10 in Appendix C is satisfied by our correlated loss model.

Using equations D.3 and D.4, the terms for $Q(i, W)$, the probability that initial window size in an epoch is $\frac{W}{2^i}$ with $W$ being the window size at the time of packet loss in previous epoch, can be derived to be:

$$Q(1, W) = \begin{cases} 1 & \text{if } W < 4 \\ (1-p')(1-p)^{W-2} & \text{otherwise} \end{cases} \tag{D.5}$$

$$Q(2,W) = \begin{cases} 1 - Q(1,W) & \text{if } W < 10 \\ p'(1-p')(1-p)^{W-3} + (1-p')(1-p)^{W-3}p & \\ +(W-3)p(1-p')^2 * (1-p)^{W-4} & \text{otherwise} \end{cases} \tag{D.6}$$

and

$$Q(3,W) = 1 - Q(1,W) - Q(2,W) \tag{D.7}$$

Note that $Q(1,W)$ is a monotonically decreasing function of $W$. Thus $\alpha_1 = Q(1,W_{f2}) - Q(1,W_{f1})$ where $W_{f2} > W_{f1}$ is always negative. As in *Random Loss Model,* a sufficient criterion for uniqueness of $W_f$ for a given $p$ can be derived to be $\alpha_3 \geq 0$. As in *Random Loss Model*, we prove that as $W_f$ is increased from $W$ to $W+1$, $\alpha_3$ is always greater than or equal to 0.

Let

$$\begin{aligned} \alpha_3 &< 0 \\ \Rightarrow -\alpha_1 - \alpha_2 &< 0 \\ \Rightarrow \alpha_1 + \alpha_2 &> 0 \\ \Rightarrow Q(1,W+1) - Q(1,W) + Q(2,W+1) - Q(2,W) &> 0 \end{aligned} \tag{D.8}$$

For $W+1 < 10$, the above inequality can be written as

$$\begin{aligned} Q(1,W+1) - Q(1,W) + 1 - Q(1,W+1) - 1 + Q(1,W) &> 0 \\ \Rightarrow 0 &> 0 \\ \Rightarrow & \; False \end{aligned}$$

For $W+1=10$, Equation D.8 can be written as

$$\begin{aligned}
Q(1,W+1)-Q(1,W)+Q(2,W+1)-1+Q(1,W) &> 0\\
\Rightarrow Q(1,W+1)+Q(2,W+1) &> 1\\
&\Rightarrow \quad False
\end{aligned}$$

For other values of $W$, Equation D.8 can be written as

$(1-p')(1-p)^{W-1}-(1-p')(1-p)^{W-2}+(p'+p)(1-p')(1-p)^{W-2}+(W-2)p(1-p')^2(1-p)^{W-3}-(p'+p)(1-p')(1-p)^{W-3}-(W-3)p(1-p')^2(1-p)^{W-4}>0$

$\Rightarrow (1-p)^3-(1-p)^2+(p'+p)(1-p)^2+(W-2)p(1-p')(1-p)-(p'+p)(1-p)-(W-3)p(1-p')>0$

$\Rightarrow -p(1-p)^2-p(p'+p)(1-p)+(W-2)p(1-p')(1-p)-(W-3)p(1-p')>0$

$\Rightarrow -(1-p)^2-(p'+p)(1-p)+(W-2)(1-p')(1-p)-(W-3)(1-p')>0$

$\Rightarrow (-1+p-p'-p+(W-2)(1-p'))(1-p)>(W-3)(1-p')$

$\Rightarrow (-1-p'+1-p'+(W-3)(1-p'))(1-p)>(W-3)(1-p')$

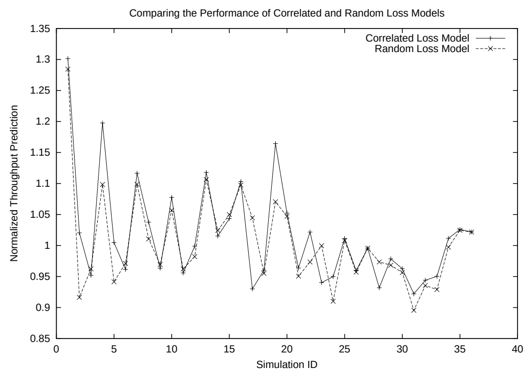
$\Rightarrow (-2p'+(W-3)(1-p'))(1-p)>(W-3)(1-p')$

$\Rightarrow False$

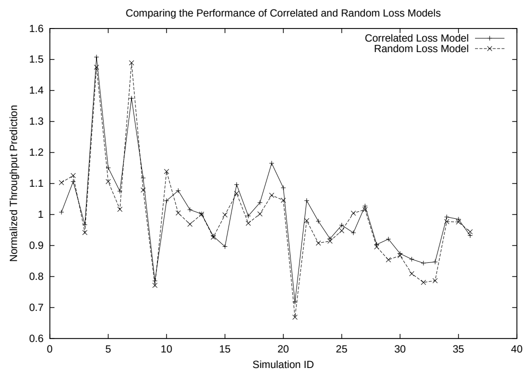
Thus, we have shown that each time $W_f$ is incremented by 1, the increase in $W_i$ is less than 1 and hence the $W_f$ determined from Equation 3.29 is unique for a given $p$ under our correlated loss model.

Next, we evaluate the throughput prediction accuracy achieved with the correlated loss model and compare it against that of the random loss model. As we saw earlier, the non-invasive estimates for the correlated loss parameters are not satisfactory. Hence, the performance results presented here are the ones obtained with accurate information of network parameters only. Figure D.3 shows the normalized (with respect to actual average throughputs) predictions made by correlated and random loss models with accurate network parameter information as input. As expected the correlated loss model performs better than the random loss model in the scenarios with significant correlation in packet losses. In general, correlated loss model tends to give higher throughput estimates than the random loss model. This is understandable as the values for the applicable loss rates in the *non-congested* state used in correlated loss model will be lower than the average loss rates used in random loss model. This also means that while improved througput estimates are obtained for high correlation scenarios, a few instances of overestimates can be observed otherwise.

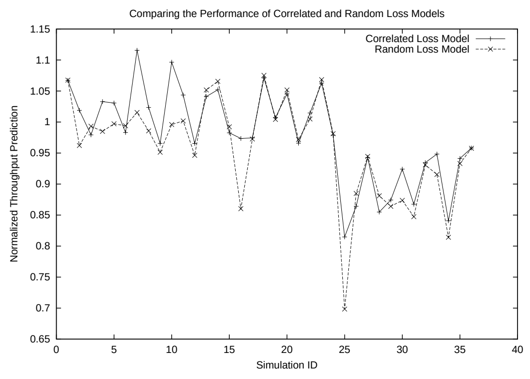
To conclude, we reiterate the limited usefulness of the correlated loss model in the absence of reliable non-invasive means to estimate the correlation as observed by the TCP flows. Investigating scalable non-invasive estimation methods that can reliably measure application centric network properties is an important area of research which we hope to explore further in our future works.



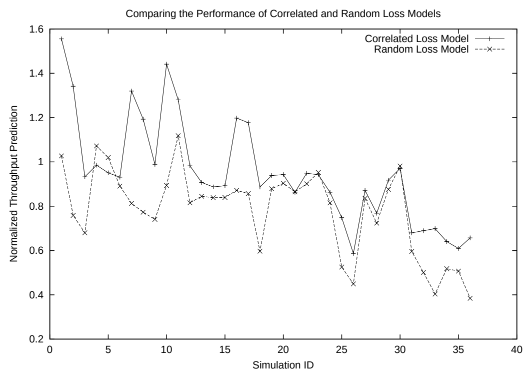
(a) Single RED Router, Homogeneous RTT Flows



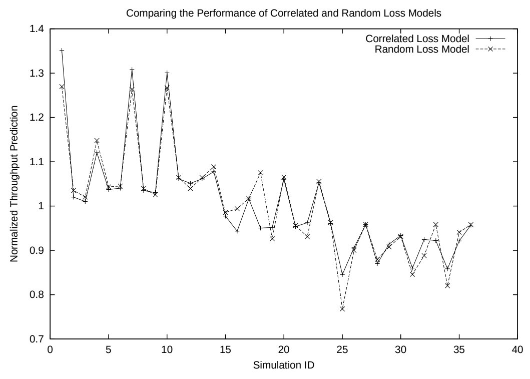
(b) Single Droptail Router, Homogeneous RTT Flows



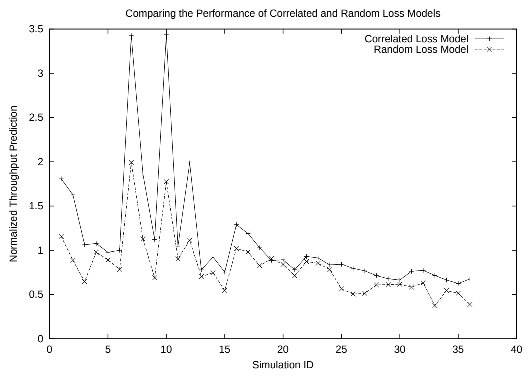
(c) Single RED Router, Heterogeneous RTT Flows



(d) Single Droptail Router, Heterogeneous RTT Flows



(e) Two RED Routers



(f) Two Droptail Routers

Figure D.3: Simulation results comparing the performance of the correlated and random loss models.

# BIBLIOGRAPHY

[1] ETSI TR 003. Network aspects (na) ; general aspects of quality of service (qos) and network performance (np).

[2] C. Alaettinoglu and S. Casner. Detailed analysis of is-is routing protocol on the qwest backbone. *NANOG 24*, February 2002.

[3] C. Alaettinoglu, V. Jacobson, and H. Yu. Towards millisecond igp convergence. In *NANOG 20*, October 2000.

[4] M. Allman. A web server's view of the transport layer. *ACM Computer Communication Review*, 30(5), October 2000.

[5] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. Request For Comments (Standards Track) RFC 2581, Internet Engineering Task Force, April 1999.

[6] D. P. Anderson, R. Govindan, and G. Homsy. Design and implementation of a continuous media i/o server. In *Proc. First International Workshop on Network and Operating System Support for Digital Audio and Video*, November 1990.

[7] D. P. Anderson, S.-Y. Tzou, R. Wahbe, R. Govindan, and M. Andrews. Support for continuous media in the dash system. In *Proc. 10th International Conference on Distributed Computer Systems*, pages 54–61, May 1990.

[8] J. Anderson, B. Doshi, S. Dravida, and P. Harshavardhana. Fast restoration of atm networks. *IEEE Journal on Selected Areas in Communications*, 12(1):128–138, January 1994.

[9] M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne. Real-time communications in packet-switched networks. *Proceedings of the IEEE*, 82:122–139, January 1994.

[10] J. Ash, G. Choudhury, V. Sapozhnikova, M. Sherif, V. Manral, and A. Maunder. Congestion avoidance and control for ospf networks. INTERNET-DRAFT, draft-ash-manral-ospf-congestion-control-00.txt, April 2002. (work in progress).

[11] P. Aukia, M. Kodialam, P. Koppol, T. Lakshman, H. Sarin, and B. Suter. RATES: A server for MPLS traffic engineering. *IEEE Network*, pages 34–41, March/April 2000.

[12] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principals of internet traffic engineering. Request For Comments (Informational) RFC 3272, Internet Engineering Task Force, May 2002.

[13] E. Baccelli and R. Rajan. Monitoring ospf routing. In *IM 2001: IFIP/IEEE International Symposium on Integrated Network Management*, May 2001.

[14] F. Baker and R. Coltun. OSPF Version 2 Management Information Base. Request for Comments 1850, November 1995.

[15] A. Basu and J. Riecke. Stability issues in ospf routing. In *Proc. SIGCOMM'2001*, pages 225–236, August 2001.

[16] Y. Birk and N. Bloch. Improving network performance with prioritized dispersal. In *Proc. INFOCOM'2000*, pages 1817–1826, March 2000.

[17] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Request For Comments (Proposed Standard) RFC 2475, Internet Engineering Task Force, December 1998.

[18] A. Bouch and M. Sasse. Why value is everything: A user-centred approach to network quality of service and pricing. In *Proc. IWQoS'2001*, pages 59–74, June 2001.

[19] B.V. Caenegem, W.V. Parys, F.D. Turck, and P.M. Demeester. Dimensioning of survivable wdm networks. *IEEE Journal on Selected Arreas in Communication*, 16(7):1146–1157, September 1998.

[20] R. Callon. Use of OSI IS-IS for Routing in TCP/IP and Dual Enviornments. RFC1195, December 1990.

[21] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for internet load balancing. In *Proc. INFOCOM'2000*, pages 332–341, March 2000.

[22] C. Casetti and M. Meo. A new approach to model the stationary behavior of TCP connections. In *Proc. INFOCOM'2000*, pages 367–375, March 2000.

[23] ISO/IEC 13818-2 CD. Information technology- generic coding of moving pictures and associated audio recommendations h.262, November 1993.

[24] T. Chiang and D. Anastassiou. Hierarchical coding of digital television. *IEEE Communications Magazine*, pages 38–45, May 1994.

[25] H. Choi, S. Subramaniam, and H. Choi. On double-link failure recovery in wdm optical networks. In *Proc. INFOCOM'2002*, June 2002.

[26] G. Choudhury, V. Sapozhnikova, A. Maunder, and V. Manral. Explicit marking and proritized treatment of specific igp packets for faster igp convergence and improved network scalability and stability. INTERNET-DRAFT, draft-ietf-ospf-scalability-01.txt, April 2002. (work in progress).

[27] D. Cohen. Issues in transnet packetized voice communication. In *Proc. Fifth Data Communications Symposium*, pages 6–13, September 1977.

[28] MPGEG-7: Context and Objectives. http://drogo.cselt.stet.it/mpeg/standards/mpeg-7/mpeg-7.htm.

[29] O. Crochat and J.L. Boudec. Design protection for wdm optical networks. *IEEE Journal on Selected Arreas in Communication*, 16(7):1158–1165, September 1998.

[30] A. Dacomo, S.D. Patre, G. Maier, A. Pattavina, and M. Martinelli. Design of static resilient wdm mesh networks with multiple heuristic criteria. In *Proc. INFOCOM'2002*, pages 1793–1802, June 2002.

[31] M. Daneshmand and C. Savolaine. Measuring outages in telecommunications switched networks. *IEEE Communications Magazine*, pages 34–38, June 1993.

[32] E. Dijkstra. A note on two problems in connection with graphs. *Numerische mathematik, 1:269-271*, 1959.

[33] R. Droms. Dynamic host configuration protocol. RFC 2131, March 1997.

[34] P. Drucker. Management. Butterworth-Heinemann, 1999.

[35] D. Dunn, W. Grover, and M. MacGregor. Comparison of k-shortest paths and maximum flow routing for network facility restoration. *IEEE Journal on Selected Arreas in Communication*, 12(1):88–99, January 1994.

[36] ITU-T Recommendation E.800.

[37] G. Ellinas, A.G. Hailemariam, and T.E. Stern. Protection cycles in mesh wdm networks. *IEEE Journal on Selected Arreas in Communication*, 18(10):1924–1937, October 2000.

[38] D. Awduche et. al. Overview and principles of internet traffic engineering. RFC 3272, May 2002. (Informational).

[39] D. Colle et. al. Data-centric optical networks and their survivability. *IEEE Journal on Selected Arreas in Communication*, 20(1):6–20, January 2002.

[40] P. Demeester et. al. Resilience in multi-layer networks. *IEEE Communications Magazine*, pages 70–75, August 1999.

[41] E. Mannie et.al. Recovery protection and restoration terminology for gmpls. INTERNET-DRAFT, draft-ietf-ccamp-gmpls-recovery-terminology-01.txt, November 2002. (work in progress).

[42] G. Li et.al. Rsvp-te extensions for shared-mesh restoration in transport networks. INTERNET-DRAFT, draft-li-shared-mesh-restoration-01.txt, November 2001. (work in progress).

[43] L. Berger et.al. Generalized mpls - signaling functional description. INTERNET-DRAFT, draft-ietf-mpls-generalized-signaling-09.txt, August 2002. (work in progress).

[44] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexfod, and F. True. Deriving traffic demands for operational ip networks: methodology and experience. *IEEE/ACM Transactions on Networking*, 9(3):265–279, June 2001.

[45] Anja Feldmann and Jennifer Rexford. IP network configuration for traffic engineering. Technical Report 000526-02, AT&T Labs – Research, May 2000.

[46] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. SIGCOMM'2000*, pages 43–56, Stockholm, Sweden, August 2000.

[47] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[48] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional ip routing protocols. *IEEE Communications Magazine*, pages 118–124, October 2002.

[49] B. Fortz and M. Thorup. Internet traffic engineering by optimizing ospf weights. In *Proc. INFOCOM'2000*, pages 519–528, March 2000.

[50] B. Fortz and M. Thorup. Optimizing ospf/is-is weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, May 2002.

[51] P. Fraciosa, D. Frigioni, and M. Giaccio. Semi-dynamic shortest paths and breadth-first search in digraphs. In *Proc. 14th Symp. Theoratical Aspects of Computer Science*, pages 33–46, March 1997.

[52] D. Frigioni, M. Ioffreda, U. Nanni, and G. Pasqualone. Experimental analysis of dynamic algorithms for the single-source shortest path problem. *ACM Journal of Experimental Algorithmics, 3:5*, 1998.

[53] A. Fumagalli and L. Valcarenghi. Ip restoration vs. wdm protection: Is there an optimal choice? *IEEE Network Magazine*, pages 34–41, November/December 2000.

[54] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A framework for ip based virtual private networks. RFC 2764, February 2000.

[55] R. Govindan and H. Tangmunarunkit. Heuristics for internet map discovery. In *Proc. IEEE INFOCOM*, pages 1371–1380, March 2000.

[56] M. Goyal, K.K. Ramakrishnan, and W. Feng. Achieving faster failure detection in ospf networks. In *To Appear in Proc. ICC'2003*, May 2003.

[57] M. Goyal, J. Yates, and G. Li. Benefits of restoration signaling message aggregation. In *To Appear in Proc. OFC'2003*, March 2003.

[58] S. Han and K. Shin. Fast restoration of real-time communication service from component failures in multi-hop networks. In *Proc. SIGCOMM'1997*, pages 77–88, September 1997.

[59] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. Sip: session initiation protocol. RFC 2543, March 1999.

[60] B. Haskell, A. Puri, and A. Netravali. Digital video: An introduction to mpeg-2. Digital Multimedia Standards Series, Chapman and Hall, 1997.

[61] O. Hauser, M. Kodialam, and T.V. Lakshman. Capacity design of fast path restorable optical networks. In *Proc. INFOCOM'2002*, June 2002.

[62] M. Herzberg, S. Bye, and A. Utano. The hop-limit approach for spare capacity assignment in survivable networks. *IEEE/ACM Transactions on Networking*, 3(6):775–784, December 1995.

[63] P. Ho and H.T. Mouftah. A framework for service-guaranteed shared protection in wdm mesh networks. *IEEE Communications Magazine*, pages 97–103, February 2002.

[64] http://www.erlang.com/bandwidth.html.

[65] C. Huitema. *Routing in the Internet*. Prentice Hall, 2000.

[66] ITU-T Recomm. I.350. General aspects of quality of service and network performance in digital networks, including isdns, March 1993.

[67] R. Iraschko, M. MacGregor, and W. Grover. Optimal capacity placement for path restoration in stm or atm mesh survivable networks. *IEEE/ACM Transactions on Networking*, 6(3):325–336, June 1998.

[68] ITU. One-way transmission time. Recommendation G.114, February 1996.

[69] V. Jacobson. Berkeley TCP evolution from 4.3-Tahoe to 4.3-Reno. In *Proceedings of the Eighteenth Internet Engineering Task Force*, page 365, Vancouver, B.C., 1990. University of British Columbia.

[70] JTC1/SC2/WG10. Digital compression and coding of continuous-tone still images.

[71] JTC1/SC2/WG11. Coding of moving pictures and associated audio for digital storage media up to about 1.5 mbit/s. ISO/IEC DIS 11172, 1992.

[72] E. Altman K. Avrachenkov and C. Barakat. A stochastic model of TCP/IP with stationary random losses. In *Proc. SIGCOMM'2000*, pages 231–242, August/September 2000.

[73] H. Kalva, A. Eleftheriadis, A. Puri, and R. Schmidt. Stored file formats for mpeg-4. Contribution ISO-IEC JTC1/SC29/WG11 MPEG97/2062, 39th MPEG meeting, April 1997.

[74] K. Kar, M. Kodialam, and T.V. Lakshman. Routing restorable bandwidth guaranteed connections using maximum 2-route flows. In *Proc. INFOCOM'2002*, June 2002.

[75] D. Katz, D. Yeung, and K. Kompella. Traffic engineering extensions to ospf version 2. INTERNET-DRAFT, draft-katz-yeung-ospf-traffic-09.txt, October 2002. (work in progress).

[76] R. Kawamura, K. Sato, and I. Tokizawa. Self-healing atm networks based on virtual path concept. *IEEE Journal on Selected Arreas in Communication*, 12(1):120–127, January 1994.

[77] S. Kent and R. Atkinson. Security architecture for the internet protocol. RFC 2401, November 1998. (Standards Track).

[78] G. Kessler and S. Shepard. A primer on internet and tcp/ip tools and utilities. Request For Comments (Informational) RFC 2151, Internet Engineering Task Force, October 1996.

[79] M. Kodialam and T.V. Lakshman. Dynamic routing of bandwidth guaranteed tunnels with restoration. In *Proc. INFOCOM'2000*, pages 902–911, March 2000.

[80] M. Kodialam and T.V. Lakshman. Minimum interference routing with applications to mpls traffic engineering. In *Proc. INFOCOM'2000*, pages 884–893, March 2000.

[81] M. Kodialam and T.V. Lakshman. Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information. In *Proc. INFOCOM'2001*, pages 376–385, April 2001.

[82] M. Kodialam and T.V. Lakshman. Integrated dynamic ip and wavelength routing in ip over wdm networks. In *Proc. INFOCOM'2001*, pages 358–366, April 2001.

[83] A. Kumar. Comparative performance analysis of versions of TCP in a local network with a lossy link. *IEEE/ACM Trans. Netw.,* 6(4):485–498, August 1998.

[84] C. Labovitz, A. Ahuja, and F. Jahanian. Experimental study of internet stability and wide-area backbone failures. Technical Report CSE-TR-382-98, 16, 1998.

[85] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high delay-bandwidth products. *IEEE/ACM Trans. Netw.*, 5(3):336–350, June 1997.

[86] K.C. Lee and V. Li. A wavelength-convertible optical network. *IEEE/OSA Journal of Lightwave Technology, 11(5/6):962-970*, May/June 1993.

[87] G. Li, D. Wang, C. Kalmanek, and R. Doverspike. Efficient distributed path selection for shared restoration connections. In *Proc. INFOCOM'2002*, 2002.

[88] G. Li, J. Yates, R. Doverspike, and D. Wang. Experiments in fast restoration using gmpls in optical/electronic mesh networks. In *Proc. OFC'2001*, March 2001.

[89] G. Li, J. Yates, R. Doverspike, and D. Wang. Experiments in fast restoration using gmpls in optical/electronic mesh networks. In *Proc. Optical Fiber Communications Conference 2001*, March 2001.

[90] S. Liew and K. Lu. A framework for characterizing disaster-based network survivability. *Journal on Selected Areas in Communications*, 12(1):52–58, January 1994.

[91] Mapnet: macroscopic internet visualization and measurement tool. *http://www.caida.org/tools/visualization/mapnet.*

[92] V. Marziale and A. Vitaletti. A framework for internet qos requirements definition and evaluation: an experimental approach. http://www.mobilesummit2001.org/mcs2001/papers/MOBCS4VYK6Y.pdf.

[93] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgement options. Request For Comments (Informational) RFC 2018, Internet Engineering Task Force, October 1996.

[94] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Commun. Rev.*, 27(3), July 1997.

[95] K. McCloghrie and M. Rose. Management information base for network management of TCP/IP-based internets: MIB-II. Request For Comments (Informational) RFC 1213, Internet Engineering Task Force, March 1991.

[96] J. McDonald. Public networks - dependable? *IEEE Communications Magazine*, pages 110–112, April 1992.

[97] D. Mills. Simple network time protocol (sntp) version 4 for ipv4, ipv6 and osi. RFC 2030, October 1996.

[98] P. Mockapetris. Domain names - concepts and facilities. RFC 1034, November 1987.

[99] E. Modiano and A. Narula-Tam. Survivable routing of logical topologies in wdm networks. In *Proc. INFOCOM'2001*, pages 348–357, April 2001.

[100] R.H. Moffett. Echo and delay problems in some digital communication systems. *IEEE Communications Magazine*, 25:41–47, August 1987.

[101] J. T. Moy. *OSPF : Anatomy of an Internet Routing Protocol*. Addison-Wesley, January 1998.

[102] John Moy. OSPF Version 2. Request for Comments 2328, April 1998.

[103] K. Murakami and H. Kim. Optimal capacity and flow assignment for self-healing atm networks based on line and end-to-end restoration. *IEEE/ACM Transactions on Networking*, 6(2):207–221, April 1998.

[104] A. Narula-Tam and E. Modiano. Dynamic load balancing for wdm packet networks. In *Proc. INFOCOM'2000*, pages 1010–1019, March 2000.

[105] NetFlow - Cisco IOS software. Cisco Technical Documents.

[106] S. Nojo and H. Watanabe. Incorporating reliability specifications in the design of telecommunication networks. *IEEE Communications Magazine*, pages 40–43, June 1993.

[107] The network simulator - NS2. http://www.isi.edu/nsnam/ns/.

[108] Quality of Service Requirements for IP-SAN Networks. http://www.sanvalley.com/pdf/qos_whitepaper.pdf.

[109] http://networks.ecse.rpi.edu/ sunmin/rtprotols/.

[110] Deploying H.323 Conferencing on your IP Network. Technology white paper, first virtual communications. http://www.fvc.com/software/pdf/whitepaper_deployingh323.pdf.

[111] D. Oran. Osi is-is intra-domain routing protocol. Request For Comments RFC 1142, Internet Engineering Task Force, February 1990.

[112] A. Orda and A. Sprintson. Qos routing: The precomputation perspective. In *Proc. INFOCOM'2000*, pages 128–136, March 2000.

[113] C. Ou, H. Zang, and B. Mukherjee. Sub-path protection for scalability and fast recovery in wdm mesh networks. In *Proc. OFC'2002*, March 2002.

[114] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno performance: A simple model and its empirical validation. *IEEE/ACM Trans. Netw.*, 8(2):133–145, April 2000.

[115] V. Paxson and M. Allman. Computing TCP's retransmission timer. Request For Comments (Standards Track) RFC 2988, Internet Engineering Task Force, November 2000.

[116] J. Postel. User datagram protocol. RFC 768, August 1980.

[117] G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms, 21(2):267-305*, 1996.

[118] S. Ramamurthy and B. Mukherjee. Survivable wdm mesh networks, part 1 - protection. In *Proc. INFOCOM'1999*, pages 744–751, March 1999.

[119] S. Ramamurthy and B. Mukherjee. Survivable wdm mesh networks, part 2 - restoration. In *Proc. ICC'1999*, pages 2023–2030, June 1999.

[120] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communcation Review*, 27(1):31–41, January 1997.

[121] I. Rubin and J. Ling. Failure protection methods for optical meshed-ring communications networks. *IEEE Journal on Selected Arreas in Communication*, 18(10):1950–1960, October 2000.

[122] A. Abouzeid S. Roy and M. Azizoglu. Stochastic modeling of TCP over lossy links. In *Proc. INFOCOM'2000*, pages 1724–1733, March 2000.

[123] N. Cardwell S. Savage and T. Anderson. Modeling TCP latency. In *Proc. INFOCOM'2000*, pages 1742–1751, March 2000.

[124] L. Sahasrabuddhe, S. Ramamurthy, and B. Mukherjee. Fault management in ip-over-wdm networks: Wdm protection versus ip resotration. *IEEE Journal on Selected Arreas in Communication*, 20(1):21–33, January 2002.

[125] H. Schulzrinne. Internet services: from electronic mail to real-time multimedia. In *Proc. of KIVS (Kommunikation in Verteilten Systemen) (Klaus Franke, Uwe Hobner, and Winfried Kalfa, eds.), Informatik aktuell, (Chemnitz, Germany),* pages 21–34, February 1995.

[126] A. Shaikh, M. Goyal, A. Greenberg, R. Rajan, and K. K. Ramakrishnan. An ospf topology server: design and evaluation. *IEEE Journal on Selected Areas in Communications (JSAC), Vol. 20, no 4*, May 2002.

[127] A. Shaikh, M. Goyal, A. Greenberg, R. Rajan, and K.K. Ramakrishnan. An OPSF Topology Server: Design and Experience. Technical report, AT&T Research, March 2001. full version.

[128] A. Shaikh and A. Greenberg. Experience in black-box ospf measurement. In *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW)*, pages 113–125, November 2001.

[129] A. Shaikh, L. Kalampoukas, R. Dube, and A. Varma. Routing stability in congested networks: Experimentation and analysis. In *Proc. SIGCOMM'2000*, pages 163–174, Stockholm, Sweden, August 2000.

[130] R. Siamwalla, R. Sharma, and S. Keshav. Discovering internet topology. In *Proc. IEEE INFOCOM*, July 1999.

[131] William Stallings. *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*. Addison-Wesley, 1999.

[132] R. Steinmetz and C. Engler. Human perception of media synchronization. Technical Report 43.9310, IBM European Networking Center Heidelberg, 1993.

[133] B. Teitelbaum and T. Hanss. Qos requirements for internet2. http://www.internet2.edu/qos/may98Workshop/html/requirements.html.

[134] M. Thorup. Fortifying ospf/is-is against link-failure. *Unpublished*.

[135] L. Torres and M. Kunt. Video coding. Kluwer Academic Publishers, 1996.

[136] G. Trewitt. Topological analysis of local area internetworks. *Computer Commun. Rev.*, 18(4):1–12, August 1988.

[137] International Telecommunication Union. Packet based multimedia communication systems. Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, February 1998.

[138] N. Venkatasubramaniam and K. Nahrstedt. An integrated metric for video qos. In *Proc. ACM International Multimedia Conference*, pages 371–381, November 1997.

[139] O. Verscheure, P. Frossard, and M. Hamdi. Mpeg-2 video services over packet networks: Joint effect of encoding rate and data loss on user-oriented qos. In *Proc. NOSSDAV'98*, July 1998.

[140] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (version 3). RFC 2251, December 1997.

[141] Y. Wang, Z. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *Proc. INFOCOM'2001*, pages 565–571, April 2001.

[142] C. Weinstein and J.W. Forgie. Experience with speech communication in packet networks. *IEEE J. on Selected Areas in Communications*, 6(1):963–980, December 1983.

[143] G. Wright and W. Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Addison–Wesley, Reading, Massachusetts, 1996.

[144] Y. Xiong and L. Mason. Restoration strategies and spare capacity requirements in self-healing atm networks. *IEEE/ACM Transactions on Networking*, 7(1):98–110, February 1999.

[145] Y. Ye, C. Assi, S. Dixit, and M. Ali. A simple dynamic integrated provisioning/protection scheme in ip over wdm networks. *IEEE Communications Magazine*, pages 174–182, November 2001.

[146] Y. Ye and S. Dixit. On joint protection/restoration in ip-centric dwdm-based optical transport networks. *IEEE Communications Magazine*, pages 174–183, June 2000.

[147] I. Yeom and A.L. Narasimha Reddy. Modeling TCP behavior in a differentiated services network. *IEEE/ACM Trans. Netw.*, 9(1):31–46, February 2001.