Efficient Provisioning Algorithms for Network Resource Virtualization with QoS Guarantees

A DISSERTATION PRESENTED

BY

KARTIK GOPALAN

TO

THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

STONY BROOK UNIVERSITY

August 2003

Abstract of the Dissertation

Efficient Provisioning Algorithms for Network Resource Virtualization With QoS Guarantees

by

Kartik Gopalan

Doctor of Philosophy

in

Computer Science

Stony Brook University

2003

In recent years, large commercial and non-commercial customers of network service providers (NSP) are increasingly demanding dedicated and fault-tolerant wide-area network connectivity between their remote endpoints with the ability to exercise fine-grained control over their share of physical network resources. This trend is being coupled with a rapid surge in the amount of real-time network traffic that the NSP's network infrastructure is required to carry. In other words, customers require a *virtualized* share of the NSP's physical network infrastructure with performance and isolation guarantees. At the same time, NSPs themselves have an inherent need to maximize their own revenue base by accommodating the requirements of as many customers as possible. These three competing forces have created an urgent requirement for resource provisioning techniques that enable network resource virtualization with performance and isolation guarantees and at the same time maximize the utilization efficiency of the network infrastructure.

This dissertation presents a comprehensive set of resource provisioning techniques to satisfy these competing requirements on the network infrastructure that carries traffic with performance constraints, also popularly known as Quality of Service (QoS) constraints. We propose three-levels of inter-dependent resource provisioning algorithms for real-time traffic flows that require long-term bandwidth guarantees and deterministic or statistical end-to-end delay bounds.

At the *link-level*, we have developed a novel measurement-based algorithm called *Delay Distribution Measurement* (DDM) based admission control, that provides distinct per-flow statistical delay guarantees. DDM can increase link utilization for voice traffic

by up to a factor of 3 compared to deterministic admission control when tolerance to delay violation is as low as 10^{-5}.

At the *path-level*, we have developed an algorithm called *Load-based Slack Sharing* (LSS), that partitions the end-to-end QoS requirement of a flow into QoS requirements at each hop along the path. The goal of partitioning is to keep the loads on different hops as balanced as possible. Compared to best previously known approach, LSS can admit up to 1.2 times more traffic with deterministic delay bounds and up to 2.8 times with statistical delay bounds.

At the *network-level*, we propose a route-selection algorithm called *Link Criticality Based Routing* (LCBR). LCBR algorithm selects primary and backup routes between a given source and destination under end-to-end delay and bandwidth constraints. It applies a simple notion of link importance to select primary and backup routes that satisfy the end-to-end QoS while balancing the load across the network. LCBR can improve the amount of admitted network traffic by up to a factor of 2 compared to its variants that use route selection criteria from earlier traffic engineering approaches.

To My Family

Contents

List of Figures

Acknowledgments

First, I would like to sincerely thank my advisor Prof. Tzi-cker Chiueh, who has helped me develop a broad interest in all aspects of systems research. Discussions with him, as a norm, always spark fresh new ideas and yield new insights into difficult problems. He is among the most productive and hard-working persons I have known and working with him has been an inspiring experience.

I would like to thank Dr. Yow-Jian Lin who took a keen interest in my work, provided numerous suggestions and extensively reviewed my paper and thesis drafts, Prof. Michael Bender who provided feedback on algorithmic aspects of my dissertation, and Prof. Alex Mohr who gave numerous suggestions on my dissertation and valuable advice on career choices. I would also like to acknowledge my other current and past committee members Prof. Yuanyuan Yang, Prof. Scott Stoller, Prof. R. Sekar and Prof. Philip Lewis. Prof. Tzi-cker Chiueh and Sheng-I Doong provided me with the opportunity to work at Rether Networks Inc. and gain a wide exposure to real-world systems problems that finally helped crystallize my dissertation topic.

Special thanks are due to my Masters thesis advisor Prof. C. Siva Ram Murthy for initiating me into the world of research, Dr. K.N. Balasubramanya Murthy for his guidance during my formative years in research, and Prof. Govindarasu Manimaran for his encouragement and advice throughout my graduate studies.

I would like to acknowledge my collaborators in various projects; Srikant Sharma, Ningning Zhu, Pradipta De, and Gang Peng in Wireless Rether; Anindya Neogi and Prashant Pradhan in Wired Rether; Ashish Raniwala in Multi-channel Wireless Mesh Networks; Prashant Pradhan in Route Lookup Cache; Srikant Sharma, Anindya Neogi,

Jiawu Chen and Wei Li in Internet Service Management Device; Srikant Sharma in Viking and RmiLocalize; Ganesh Venkitachalam in Web Service Proxy; Lan Huang in Real-time Disk Scheduling; Gang Peng and Chang Li in Gage.

Additionally, I would like to acknowledge the following people who contributed directly and indirectly to my research and were sounding boards for my thoughts. Ganesh Venkitachalam was a great source of insights in whose company I delved into details of operating systems and Linux kernel internals; Prashant Pradhan initiated interesting discussions in scheduling and OS issues and provided the base results, simulation code and traffic traces that I developed upon in the Route Lookup Cache project; Saurabh Sethia helped me with key discussions in the multi-resource scheduling and QoS routing problems and provided valuable feedback on my graduate studies and career choices; Srikant Sharma helped me apply the LCBR algorithm in the context of Viking project and always had a willing ear for my useful and useless lines of thoughts on dissertation; Pradipta De and Ashish Raniwala proof-read several paper drafts and provided valuable feedback and discussions; Tulika Mitra was the encyclopedia for my questions on architecture and a model for consistent hard work; Pedro Souto provided me valuable insights into the DSSCAN real-time disk scheduling algorithm; Srinidhi Varadarajan provided valuable feedback on career choices and was always opening my eyes to cool things that could be done in systems; Srinidhi, Prashant, Tulika and Pedro greatly helped with systems related problems during my initial days in the lab.

I also wish to acknowledge all the past and present members of the ECSL who made Ph.D. such a rewarding and enriching experience. In particular: Praveen Arora, Pradipta De, Lan Huang, Fu-Hau Hsu, Tulika Mitra, Anindya Neogi, Gang Peng, Manish Prasad, Prashant Pradhan, Ashish Raniwala, Srikant Sharma, Pedro Souto, Srinidhi Varadarajan, Ganesh Venkitachalam, Chuan-Kai Yang, and Ningning Zhu.

Finally, I owe all my achievements to my parents and family members, without whom this thesis would not have been possible. Appa and Amma supported my pursuit of dreams, even if it meant staying a world away. Venkat, Veena, Padma and Ramesh have been a source of guidance and affection all through.

This research has been supported by NSF awards EIA-9818342, ANI-9814934, and ACI-9907485, USENIX student research grants, as well as funding from Computer As-

sociates/Cheyenne Inc., National Institute of Standards and Technologies, Siemens, and Rether Networks Inc.

Chapter 1

Introduction

Growth of the Internet in recent years is being increasingly driven by requirements of large commercial and non-commercial entities. A number of such organizations are moving their mission-critical services from local networks to remote server-centers. As a result, there is a surge in demand for dedicated wide-area data network connectivity between remote branch locations of these organizations with a fine degree of control over their share of physical network resources. Performance sensitive network traffic, such as Voice over IP (VoIP), video conferencing, online trading and interactive media, constitute a growing proportion of the traffic generated by such entities. The performance requirements, also popularly known as Quality of Service (QoS) requirements, include long-term bandwidth, end-to-end delay, delay jitter, level of traffic loss, and protection against network failures. Network Service Providers (NSP), who cater to the connectivity needs of such large entities, are increasingly turning to *virtualization* as a means of managing the heterogenous customer requirements along performance, isolation, and control dimensions.

1.1 Resource Virtualization and QoS Guarantees

Network resource virtualization is a powerful paradigm that is driving large scale deployment of QoS in commercial networks. Resource virtualization refers to the ability to view multiple physical network resources as one *virtual* resource and to allocate arbi-

trary shares of the virtual resource among multiple customers. Each such virtual share of physical network infrastructure can be associated with exactly the same attributes as a corresponding physical network resource. For instance, one can build a virtual link that is composed of several underlying physical network links and is associated with virtual capacity, latency, and other QoS attributes; or one can even build a virtual network whose topology consists of several QoS-guaranteed virtual links overlaid on top of the NSP's physical network infrastructure. This is analogous to the manner in which an operating system transparently provides a dedicated virtual share of physical system resources, such as CPU, memory and disk, to each application. Network resource virtualization enables each customer to exercise fine-grained control over an allocated share of heterogeneous network resources. There is tremendous appeal in the conceptual simplicity of network resource virtualization as a means of managing customer's QoS requirements. This dissertation tackles the challenges in developing efficient provisioning algorithms for network resource virtualization with QoS guarantees.

1.2 Dissertation Overview

QoS guarantee for network traffic comes at a price – the NSP needs to dedicate a portion of physical network resources (such as link capacity, buffer space and computation resources) for each customer. From an NSP's point of view, the network resources need to be utilized in the most efficient manner possible. Despite considerable research invested on network QoS in the last decade, the actual deployment of QoS enabling technologies is woefully small in present day Internet. One of the principal reasons is the lack of mechanisms that enable NSPs to efficiently virtualize their network resources and maximize their revenue base while at the same time providing QoS guarantees to their customers' network traffic.

Provisioning techniques for network resource virtualization must be designed to satisfy two primary objectives. The first objective is to satisfy the performance requirements for each customer's network traffic. The second objective is to maximize the overall resource utilization efficiency of the network which in turn impacts the total revenue derived

by the NSP. [1] For instance, in a network that carries voice traffic, the network provider might wish to admit as many voice trunks as possible, where traffic in each voice trunk requires bounded end-to-end delay.

This dissertation proposes a comprehensive set of resource provisioning techniques for network resource virtualization that achieve these two (often conflicting) objectives on a physical network infrastructure that supports real-time network traffic with heterogeneous QoS requirements. The algorithms operate in the context of network traffic that require (a) end-to-end delay bounds, (b) long-term bandwidth guarantees, and (c) protection against failure along the flow path.

The delay bound guarantees could be either deterministic or statistical. *Deterministic* delay guarantees promise an absolute end-to-end delay bound for every packet carried by a flow. On the other hand, with *statistical* delay guarantees, the end-to-end delay bound is accompanied with a small probability of delay violation. For applications that can tolerate occasional delay bound violations, statistical guarantees can help to reduce the resource requirement for each flow. For instance, if 99.9% of the packets are observed to experience only 50% of their expected worst-case delay, statistical guarantees can potentially help reduce the per-flow resource requirement by almost half compared to deterministic guarantees. Similarly, long-term bandwidth guarantees could also be deterministic or statistical. For instance, an NSP could choose to over-subscribe the network capacity with the expectation that at any point in time, only a fraction of the customers would actively generate traffic. Fault-tolerance along the flow path could take several forms. Flow path could be protected against single or multiple node/link failures. Fault-protection could be proactive, in which backup network resources are reserved in advance, or reactive in which backup resources are discovered after the failure occurs. Fault-protection could also be local, in which backup resources are discovered in the neighborhood of the fault, or global in which backup resources are discovered over the entire network. This dissertation addresses both deterministic and statistical delay bound guarantees, deterministic long-term bandwidth guarantees, and protection against single node/link failures using a global and proactive approach.

[1] The amount of customer traffic supported over a fixed network infrastructure acts as our measure of resource utilization efficiency.

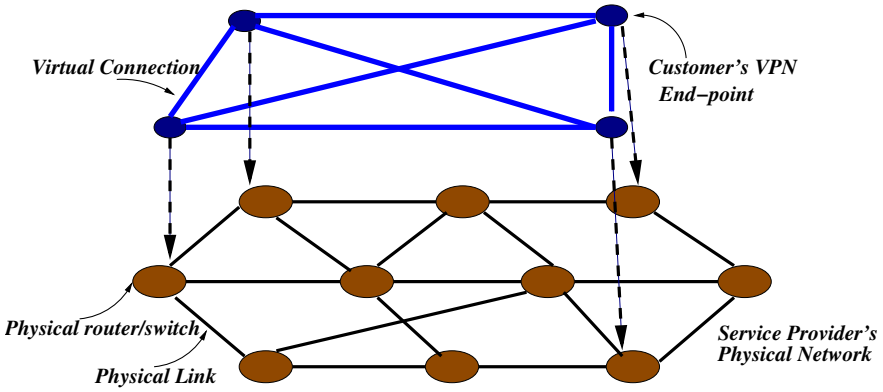

Figure 1.1: Virtual Private Network (VPN) is a collection of virtual connections between endpoints belonging to a customer. Each virtual connection traverses multiple physical links and routers. Each VPN endpoint is a source/sink of traffic on virtual connections.

1.3 Network Resource Provisioning

The process of provisioning network resources for flows with QoS guarantees is a combination of three different components – *admission control*, *resource allocation*, and *runtime enforcement*. *Admission control* determines whether or not sufficient network resources, such as link bandwidth and buffer space, exist to support QoS requirements of new flow requests. *Resource allocation* addresses the question of how much network resources to actually allocate for newly admitted flows so as to satisfy network-wide optimization objectives. *Runtime enforcement* ensures that the promised QoS guarantees are indeed delivered in a consistent manner throughout the lifetime of the flow and that the flow does not violate its traffic specifications. The specific form of provisioning depends upon the resource virtualization context in which it is applied. We illustrate three practical scenarios in which provisioning algorithms proposed in this dissertation play a critical role in obtaining high resource utilization efficiency.

1.3.1 Virtual Private Networks and Virtual Overlay Networks

Emerging trends in network resource virtualization include Virtual Private Networks and Virtual Overlay Networks, the former being the more standardized of the two and the latter still being a subject of research studies.

A Virtual Private Network (VPN) consists of a set of long-term circuit-switched tunnels, or *virtual connections,* between pairs of remote end-points in the network. A virtual connection carries a long-term aggregate traffic flow that is composed of several smaller short-lived traffic streams (such as individual TCP/IP or VoIP connections). Figure 1.1 shows a VPN in which each virtual connection provides an illusion of a dedicated data-path between two remote VPN end-points. In reality each virtual connection traverses multiple physical links and switches that are shared with other traffic streams. VPN end-points do not perform traffic forwarding or routing and act solely as traffic producers and consumers.

Virtual Overlay Networks (VON) provide the next level of resource abstraction where, in addition to the data-path virtualization, control-plane is also virtualized. Figure 1.2 shows a VON in which several nodes in the network are interconnected by means of virtual connections. A key aspect of VONs is the ability to associate any physical network attributes to virtual networks. Thus each VON can have its own virtual topology, control-plane (such as its routing protocols), and data plane (such as traffic processing and forwarding). Owners of VONs have a finer level of control and flexibility in managing virtual network resources instead of merely producing and consuming traffic, as in the case of VPNs.

VPNs and VONs raise a host of network resource provisioning issues. How does one map a virtual topology onto physical network topology? What are the best routes in the physical network for each virtual connection? Such mapping should take into account both the QoS requirements of the virtual connection as well as fault tolerance and survivability requirements of the VON/VPN. For instance, it is desirable that failure of a physical link or node does not result in disconnected virtual topology. Another aspect of special interest in this dissertation is that the mapping from virtual to physical resources must be performed in a manner that maximizes the number of VONs/VPNs that can be supported on top of a given physical network infrastructure. Additionally, runtime mechanisms are needed to ensure isolation in the dimensions of functionality, performance, and security of different VONs/VPNs sharing the physical network.

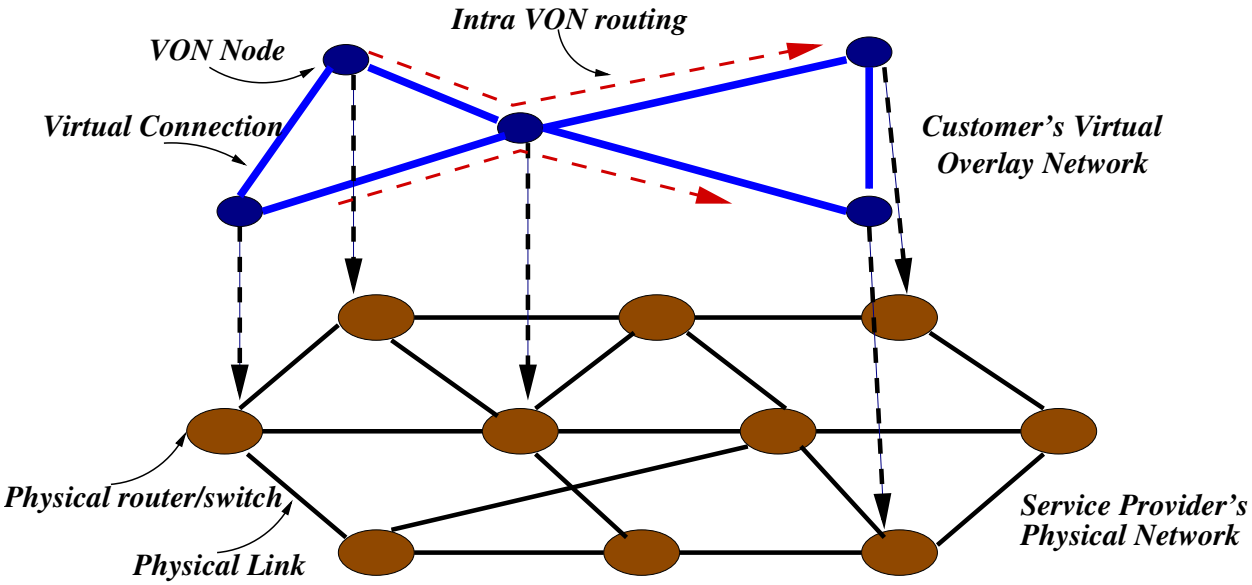

Figure 1.2: Virtual Overlay Network (VON) provides the abstraction of a virtual network overlaid on top of a physical network. Each node in a VON has a VON-specific address and is connected by virtual connections to other VON nodes. Further, every VON node can participate in control-plane activities such as intra-VON routing protocols.

1.3.2 Switched Metropolitan Area Ethernet Networks

Metropolitan Area Networks (MAN) are a rapidly growing class of networks whose geographical span extends to the boundaries of metropolitan cities. Traditionally, MANs have been composed of a set of interconnected LANs that co-operate to provide network services within a metro region. In recent years, switched Ethernet is fast becoming the preferred networking technology for deployment of MANs due to its high bandwidth, scalability, interoperability with legacy LAN technologies and dropping costs.

Large scale deployment of switched Ethernet in the metro-area context beyond the confines of LANs has raised interesting network resource provisioning issues. Specifically, there is a need to virtualize switched MAN resources in order to provide performance and isolation guarantees to customer traffic. However, the traditional switched Ethernet technology poses a significant obstacle in improving resource usage efficiency in the MAN context. Traditional switched Ethernets rely upon a single spanning tree based switching mechanism that utilizes at most N-1 links in a network of N nodes. MANs are typically designed with rich connectivity. Limiting the traffic forwarding path to a single spanning tree limits the network utilization and produces load imbalance near root of the tree. This imbalance is unhealthy from resource utilization perspective given the large amount of traffic carried over such switched networks. Additionally, failure of any active link in a spanning tree disconnects the tree and disrupts the traffic over entire network till a new spanning tree is rebuilt by activating previously blocked links. With high availability being one of the primary requirements for critical network traffic carried by MANs, network disruption of any kind is undesirable.

In order to overcome the shortcoming of the single spanning tree architecture, a multi-spanning-tree architecture has been proposed in [99]. Multiple spanning trees are overlaid on a physical switched network by novel use of the Virtual LAN (VLAN) technology. VLAN is a virtualization technology in switched LAN environments that has traditionally been used in to enforce isolation between different traffic categories. However, it has been shown in [99] that VLANs can also be effectively used to improve resource utilization efficiency in metro-area networks by construction of multiple spanning trees.

One of the critical resource provisioning issues in multi-spanning-tree MANs concerns with constructing the spanning trees in such a manner that traffic carried over the

MAN is evenly distributed over the links of switched network and roots of multiple spanning trees are evenly distributed across its nodes. For load balance, the tree construction process should ensure that different spanning trees minimally overlap with each other so as to maximize the number of active links in the switched network. Fault-tolerance requirements dictate that for traffic between each pair of end-points, there exist at least two disjoint switching paths that are parts of different spanning trees. Additionally, if network traffic comes with QoS requirements such as bandwidth or delay guarantees, then the provisioned switching paths need to satisfy the QoS constraints as well.

1.3.3 Wireless Mesh Networks

Wireless networks are gaining widespread acceptance as viable and inexpensive means of providing last-mile connectivity. Wireless mesh networks [82, 76, 20] constitute a promising technology in the context of metropolitan area networks. Fixed wireless mesh networks enable high-performance access where wired infrastructure is impractical or outdated. Wireless mesh networks eliminate the costs and delays associated with building or upgrading a wired infrastructure in a metropolitan area. These networks can emulate the topology and protocols of wired networks, but are optimized for wireless data transmission. Efficient resource virtualization and provisioning techniques are required to enable rapid deployment of wireless mesh networks on large metro-area scales with performance and traffic isolation guarantees for customer traffic.

Recent proposals [91, 68] have advocated the use of multiple wireless channels for data transmission to alleviate the problems associated with limited wireless channel access bandwidth. Such multi-channel wireless mesh networks give rise to interesting resource provisioning issues. One of the issues concerns with how to map a fixed number of channels to different access links of the wireless mesh network. The goal of the channel assignment is to maximize the cross-sectional bandwidth available between different source-destination nodes in the network by minimizing the channel interference between neighboring mesh nodes. Effective channel assignment needs to utilize measured traffic profile between source-destination pairs such that more bandwidth can be made available in parts of the network that need to shoulder higher load. The resource allocation techniques proposed in this dissertation are a critical component of network resource

management in wireless mesh networks.

1.4 Multiple Levels of Resource Allocation

Assume that an NSP receives a request for setting up a real-time flow between source S and destination D. The flow requires the QoS guarantees of end-to-end delay bound D_i, a delay violation probability P_i, average bandwidth ρ_i^{avg}, and single failure protection. There are three distinct levels at which network resource provisioning needs to be performed – (1) network-level route selection, (2) intra-path QoS constraint partitioning, and (3) link-level reservations.

1.4.1 Network-level Route Selection

The first level of resource allocation selects a distinct primary route and a backup route between source S and destination D. The primary route transports the flow's traffic during normal operations and also has sufficient resources to satisfy the QoS requirement of the flow. The backup route is kept ready to transport the flow's traffic whenever there is a single link/node failure along the primary route. An important consideration is to select the primary and backup routes in such a manner that maximum number of flow requests can be accommodated in the future.

Consider the network topology shown in Figure 1.3. We need to select a route for a flow request F_1 between nodes S_1 and D_1. There are two candidate routes: (S_1, A, B, D_1) and (S_1, C, D, D_1). Which of these two routes is better from the perspective of long-term network usage efficiency? Let's say that we expect future flow requests between S_2 and D_2. Then the better route to select for flow F_1 would be (S_1, C, D, D_1) because it leaves the resources along the link (A, B) free for future flow requests between S_2 and D_2. This simple example illustrates that route selection process should, as far as possible, avoid those physical links that are of critical importance to a large number of source-destination pairs. The challenge here is to make a judicious route selection without the knowledge of future request arrival patterns.

The fundamental requirement for being able to tolerate single-element failures is that the primary and backup routes of a flow must be completely disjoint with respect to all the

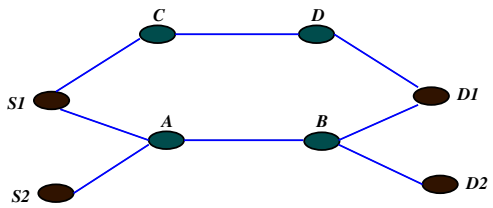

Figure 1.3: A simple problem of route selection. Selecting the route (S_1, C, D, D_1) leaves the link (A, B) free for future flows between S_2 and D_2.

intermediate network elements (except, of course, the source and destination nodes). This is to ensure complete traffic switch-over to backup route if any one element fails along the primary route. If any network element was to be shared between primary and backup routes, and the shared element happened to fail, then both primary and backup routes would be disabled simultaneously. For instance, routes (S_1, A, B, D_1) and (S_1, C, D, D_1) in Figure 1.3 satisfy the primary-backup route pair criteria. On the other hand, there happen to be no valid primary-backup route pairs between nodes S_2 and D_2.

1.4.2 Intra-path QoS Constraint Partitioning

Given a route between source S and destination D, the second level of resource allocation maps the end-to-end QoS requirements of the flow to per-physical-link QoS requirements. Consider the route shown in Figure 1.4 on which we need to set up a real-time flow with a end-to-end delay bound D_i and delay violation probability bound P_i. We need to partition D_i and P_i into per-physical-link requirements $D_{i,l}$ and $P_{i,l}$. The nature of partitioning is important since it determines the extent of load balance (or imbalance) across the links. When the links on the selected network path carry different loads, one way to partition the end-to-end QoS is to share it based on the current loads on the network links. For example, assume that the three links in Figure 1.4 carry a load of 40%, 80% and 20% respectively. In this example, the less loaded links 1 and 3 should be assigned tighter delay and delay-violation requirements than more loaded link 2; otherwise resources at link 2 can be exhausted much earlier than links 1 and 3, preventing any new flows from being admitted along the route. Note that the QoS constraint partitioning problem is relevant only in the context of additive or multiplicative parameters such as end-to-end

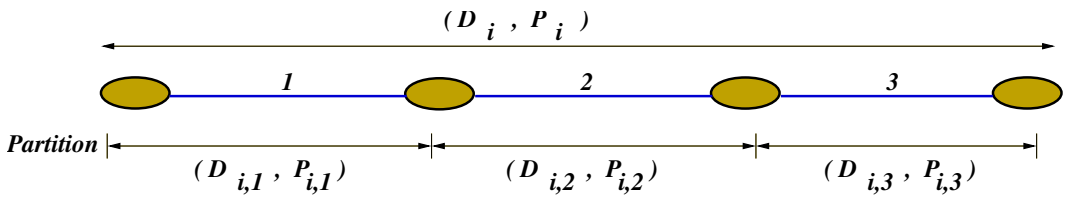

Figure 1.4: Example of partitioning end-to-end delay D_i and delay violation probability P_i over a three-hop path.

delay and delay violation probability. On the other hand, the partitioning is trivial for bottleneck QoS parameters such as long-term average bandwidth requirement and can be achieved by assigning the same minimum QoS ρ_i^{avg} at each link along the flow path.

1.4.3 Link-level Reservations

Given the link-level delay requirements $D_{i,l}$, violation probability $P_{i,l}$, and the average bandwidth requirements ρ_i^{avg}, the next level of resource allocation determines the bandwidth $\rho_{i,l}$ that needs to be reserved at the link l in order to satisfy these requirements. Note that $\rho_{i,l}$ is different from the average bandwidth requirement ρ_i^{avg} of a flow since it is dictated by both the delay requirement $(D_{i,l}, P_{i,l})$ and the average bandwidth requirement ρ_i^{avg}; specifically $\rho_{i,l} \geq \rho_i^{avg}$. While ρ_i^{avg} is average rate at which flow carries traffic, $\rho_{i,l}$ is used by link-level schedulers at run time to bound the delay experienced by packets of a flow as well as to guarantee minimum bandwidth of ρ_i^{avg}. The relationship will be further explained in Chapter 4. The problem of determining $\rho_{i,l}$ is straightforward in the deterministic case, where $D_{i,l}$ and ρ_i^{avg} are the only dimensions that affect the bandwidth reservation and the standard results in deterministic traffic scheduling suffice to derive $\rho_{i,l}$. However, in the case of statistical delay guarantees, the problem is more challenging since the presence of an additional parameter – the delay-violation probability $P_{i,l}$ – implies that we need to account for the current level of statistical multiplexing among flows traversing the link.

1.4.4 Inter-level Dependencies

There are several existing solutions that attempt to solve each level of the resource provisioning problem *independently*. For instance most of the current QoS routing algorithms perform route selection with the assumption that independent mechanisms exist for intra-path QoS constraint partitioning. Similarly, existing algorithms for intra-path QoS constraint partitioning assume that sufficient resources exist at individual physical links to support a chosen partition. At the first glance, the three levels of resource provisioning indeed appear to be three separate problems that could be tackled independently and in a sequential order. However, there are inter-dependencies between resource provisioning at multiple levels that make such independent sequential allocations undesirable. For instance, the QoS constraint partition selected along a route directly impacts the level of resource consumption at different links along the route. This in turn affects the future route selection decisions and hence the amount of traffic supported by the network. Additionally, several of the QoS constraint partitions may be infeasible due to the fact that the underlying physical links do not have sufficient resources to support them. From resource utilization efficiency perspective, one would like to select a route and a corresponding feasible QoS constraint partition that best maintains the load balance across the network links and maximizes the chances of admitting future flow requests.

1.5 Contributions

This dissertation takes an integrated approach to the problem of multi-level resource provisioning for network resource virtualization. At the link level, we propose a *Delay Distribution Measurement* (DDM) based admission control algorithm. The algorithm exploits applications' resilience to delay violations in order to provide distinct per-flow statistical delay guarantees to flows that traverse a common network link.

At route-level, we propose a *Load-based Slack Sharing* (LSS) scheme that utilizes the link-level resource mapping information to partition the end-to-end QoS such that maximum number of flows can be admitted over the route in the long term. The basic approach is to perform QoS constraint partitioning so as to keep the loads on different links along the path as balanced as possible.

At network level, we propose *Link Criticality Based Routing* (LCBR) algorithm that interacts the route-level QoS constraint partitioning algorithm to select primary and backup routes for flows that are best in terms of network resource usage efficiency. LCBR applies a simple notion of link importance to select primary and backup routes that satisfy the end-to-end QoS while balancing the load across the network.

1.6 Outline

The dissertation is organized as follows. Chapter 2 reviews related research in the area of network resource provisioning. Chapter 3 presents the model of network resource provisioning and a brief introduction on scheduling and packet delay bounds for real-time network traffic that will form a basis for subsequent sections. Chapter 4 presents the DDM admission control algorithm that guarantees statistical delay requirements at link-level. Chapter 5 presents the LSS algorithm that partitions end-to-end QoS requirements of a flow into per-hop QoS requirements. Chapter 6 presents the LCBR algorithm for network-level primary and backup route selection. Chapter 7 provides a summary of contributions from this dissertation and future research directions.

Chapter 2

Background

The principal objective of network resource management is to enhance the overall performance of a network infrastructure in terms of both the service quality visible to the users as well as the efficient use of network resources. The scope of network resource management in a wide-area network can range from managing link-level resources, such as link bandwidth and buffer space, to performing intra-path QoS partitioning, and to network-wide operations such as route selection, load-balancing, capacity planning and traffic engineering. This chapter provides the background for the work presented in this dissertation. It discusses the state of the art in network resource management for wide-area networks and places the contributions of this dissertation in the context of earlier approaches.

2.1 Prevalent Service Models

Quality of Service (QoS) refers to the availability of performance guarantees for network traffic from underlying network infrastructure. QoS guarantees could be in terms of dedicated network bandwidth, bounds on end-to-end delay, delay jitter, packet loss rate, or relatively higher priority in packet transmission. The Internet Engineering Task Force (IETF) has proposed several service models. Significant ones among these are Integrated Services (Intserv), Differentiated Services (Diffserv), RSVP, MPLS and Traffic Engineering. While Intserv and Diffserv are two contrasting approaches, other service

models overlap and often compliment each other's features. In this section, we review each of these service models and examine their inter-relationships.

2.1.1 Integrated Services

The Integrated Services (or Intserv) model [101, 57] supports QoS guarantees on a per-stream basis. It represents a fundamental shift from the best-effort Internet architecture and is influenced by the work of Ferrari [36]. The philosophy underlying the Intserv model is that routers *must* reserve per-stream resources, such as bandwidth and buffers, in order to provide QoS guarantees to network streams [16]. The Intserv model requires that routers store and maintain per-stream state, and implement additional mechanisms beyond the best-effort model such as admission control, packet classifiers and link schedulers.

Intserv model defines two types of service classes: Guaranteed Service [101], and Controlled-load Service [115]. Guaranteed service provides firm bounds on packet delays by controlling the queuing delay experienced by packets along the the path of the stream. Controlled-load service provides enhanced best-effort service, i.e. the same level of service at all times as best-effort service in a lightly loaded network. Thus controlled-load service is more suitable for applications that can tolerate some excess delays, but are sensitive to high levels of congestion. A signaling protocol, such as RSVP [17, 53], is required to set up reserved stream-paths.

The advantage of Intserv framework is that end-to-end performance guarantees can be provided on a per-stream basis. However such a fine grained control comes at the price of network scalability. Network devices can easily become overwhelmed with having to store state for millions of streams, additional mechanisms (such as classification and scheduling) can affect router performance, and Guaranteed Service requires support from every router in the network.

2.1.2 Differentiated Services

Differentiated services (or Diffserv) model [13, 56] provides an alternative approach to overcome the scalability problem of Intserv by aggregating individual streams into finite number of traffic classes and providing QoS at the coarser granularity of classes. Thus,

network devices only need to distinguish among finite number of aggregate traffic classes using the DS field in the IP header, also known as Differentiated Services Code Point (DSCP). Packets entering a Diffserv network are classified, marked, policed and shaped at the ingress router. Subsequently, The DSCP is used to determine the forwarding treatment at each intermediate router and may be re-marked when a packet traverse domain boundaries, according to agreement between domains.

IETF has standardized three service levels, also known as Per Hop Behaviors (PHB). These are (a) *best effort*, which does not have any guarantees except that it will not be starved, (b) *expedited forwarding* [58], which provides a low-latency, low-jitter, low-loss service, and (c) *assured forwarding* [52] which provides a low-loss ordered service.

The main advantage of Diffserv is its scalability since amount of state maintenance is proportional to the finite number of traffic classes. However, Diffserv performs traffic management on a per-hop basis rather than on a network-wide basis. In particular, Diffserv does not provide any absolute guarantees on the level of QoS that any traffic will receive, nor does it have support for building services with end-to-end performance guarantees. Additional traffic engineering mechanisms, such as route planning, are required to complement the capabilities of Diffserv.

2.1.3 Resource Reservation Protocol (RSVP)

RSVP is a soft-state signaling protocol that allows applications to dynamically communicate their QoS requirements to the network. RSVP was originally proposed [17, 53] on the lines of ATM signaling in the context of Intserv model, and has later been extended to other models such as Diffserv and MPLS.

Under Intserv/RSVP, the sender transmits a PATH message to the receiver specifying traffic characteristics and the intermediate routers forward the PATH message to the receiver. Upon receiving the PATH message, the receiver transmits a RESV message that back-tracks along the original path of PATH message and reserves router resource along the path. The connection establishment is successful if the request is accepted by all routers along the path, and rejected if any router along the path does not have sufficient resources.

Scalability is a principal concern under the Intserv/RSVP model. Recently, RSVP has

been extended in several ways [12, 8, 7, 53] to alleviate the scalability concern in both Diffserv and MPLS networks.

2.1.4 Multi-Protocol Label Switching

2.1.4.1 Technology

Multi-Protocol Label Switching (MPLS) [95, 19] is a popular technology that allows explicit long-term traffic tunnels to be set up in the physical network to forward packets between remote end-points. MPLS adopts the philosophy of "*route at the edge and switch in the core*". Packets belonging to a flow are marked with unique fixed-length labels at the entry points in the network (ingress routers). These labels identify their Forwarding Equivalence Class (FEC) and are used to forward the packets along pre-configured Label Switched Paths (LSP). Once marked, the packets are forwarded using light-weight label switching through traditional ATM, frame relay, or custom-built MPLS switches. Forwarding table at each switch consists of a Label Information Base (LIB) that is built using a Label Distribution Protocol (LDP). LDP could be *request-driven*, *topology driven*, or *traffic driven*. Request driven LDPs are based on traditional signaling mechanisms such as RSVP [17] or ATM signaling [24, 25]. Topology driven LDPs are based on dynamic routing information such as BGP [94], OSPF [78] or MPLS-LDP [2]. Traffic driven LDPs use monitored network information to construct LIBs [80].

2.1.4.2 Applications of MPLS

MPLS decouples the route determination and forwarding mechanisms in the network. This decoupled architecture enables routes to be determined by complex traffic engineering considerations that are different from current hop-by-hop IP routing based purely on packet destination address. Specifically, traffic engineering mechanisms that use MPLS can explicitly route the network traffic on alternate paths different from the shortest paths (selected by traditional IGP routing) and thus achieve more balanced network utilization.

MPLS can be applied in several different contexts [15, 6]. The most common current application of MPLS is to avoid congested routes or links in the network by explicitly creating LSP tunnels over alternate paths that bypass the point of congestion. MPLS can also

be helpful in making use of multiple parallel links between nodes (such as transoceanic links) by distributing aggregate network traffic across LSPs that are distributed over the parallel links. MPLS supports the concept of affinities that can be used to group together links with common properties for routing considerations. For instance, affinities can be used to restrict continental traffic from traversing transoceanic links. A useful application of MPLS is in provisioning hot-standby LSPs that provide backup routes for primary LSP's traffic in case of network failures.

Another important application of MPLS, that relates to the subject of this dissertation, is as a QoS enabling mechanism in the Internet [116]. MPLS is an interesting and versatile technology that permits the co-existence of both Intserv and Diffserv-like mechanisms on the same network infrastructure. For instance, an LSP that carries aggregate traffic could be set up in the network core with absolute performance guarantees using Intserv/RSVP mechanisms. At the same time, packets from different traffic streams within the same aggregate LSP traffic could be treated differently using Diffserv mechanisms at the edges. Alternatively, different Diffserv traffic classes could use different LSPs, essentially partitioning the physical network into multiple virtual networks, with one virtual network for each traffic class.

2.1.4.3 Circuit Switching Paradigm of MPLS

MPLS introduces a *circuit-switching paradigm* on top of the basic packet-switching framework of the Internet. In fact it has been argued [35] that the circuit-switching paradigm will become increasingly prevalent in the future, with core of the Internet mainly carrying circuit-switched traffic aggregates and IP based packet-switching mainly thriving at the edges to provide best effort services. The fundamental advantage of circuit-switching paradigm of MPLS is that it enables performance isolation between traffic that belongs to different LSPs – something that packet-switching by itself cannot guarantee. Performance-isolation means that one can prevent the performance of one flow from being effected by a traffic belonging to another flow. LSPs can help maintain QoS guarantees for each flow that is independent of other traffic streams sharing the physical network.

2.1.5 Traffic Engineering

Traffic Engineering is broadly concerned with improving performance and resource utilization of networks [5, 6]. Specifically, traffic engineering deals with mapping of traffic flows onto routes along the physical network so as to achieve desired operational objectives [15]. For instance, some operational objectives could be (a) to guarantee QoS to customer traffic, (b) to maximize service provider's revenue by improving resource utilization, (c) to keep the network free from congestion etc. Several of these and other objectives may co-exist simultaneously, in which case Traffic Engineering can be generalized to the problem of Constraint-Based Routing. MPLS, discussed in Section 2.1.4, is one of the technologies that help realize traffic engineering objectives. Other techniques could include manipulation of dynamic routing metrics in IGP or BGP protocols or explicit routing using virtual circuits in ATM or Frame relay networks.

Traffic engineering techniques could be classified in a number of ways as noted in [5]. Three primary methods of classification are (a) online vs. offline, (b) centralized vs. distributed, and (c) local vs. global.

2.1.5.1 Online vs. Offline

In online traffic engineering, routing plans need to be computed and executed in real-time. Online computations need to be simple and fast, though not necessarily close to optimal. For instance, congestion control schemes based on load-balancing via LSP tunnels [31] constitute online traffic engineering. Another feature of online techniques in the context of QoS-based traffic engineering is that path set-up requests are provisioned one at a time, as and when the requests arrive. Online techniques do not assume any knowledge of future requests and typically do not re-provision older paths to accommodate new ones.

On the other hand, in offline traffic engineering, routing plans could be computed offline at longer time-scales and downloaded to network elements periodically or only when current plans become unacceptably non-optimal. Offline traffic engineering is typically applicable where traffic characteristics are stable. In the context of QoS-based traffic engineering, offline techniques periodically optimize the routes for all currently active flows as well as any new route requests. Such knowledge of all requests being provisioned enables offline computations to be more optimal than online techniques. Offline compu-

tations tend to be more computationally expensive, such as based on extensive searches of the solution space. An example of offline traffic engineering is constraint-based routing [6].

2.1.5.2 Centralized vs. Distributed

Centralized traffic engineering is performed by a central entity that collects information, such as network parameters and traffic demands, and constructs route plans on behalf of all network elements. The periodicity of route computation and update is important in determining the effectiveness of centralized traffic engineering. In distributed traffic engineering, routers perform route selection independently based on their respective view of the state of network. The network state can be obtained by the router either by distributed link-state exchange or by means of probing. Most offline traffic engineering techniques fall into the category of centralized approach. Online traffic engineering could also be implemented in a centralized manner, but are generally implemented in a distributed manner for better response times and scalability.

2.1.5.3 Local vs. Global

Local traffic engineering techniques make routing decisions based on state of a small local portion of the network, such as bandwidth or traffic loss rate along specific paths. On the other hand, global traffic engineering techniques make routing decisions using information about entire network. Global techniques typically result in more optimal routing decisions than local techniques. However, the former also incur higher communication overhead in terms of collection and dissemination of network state information. Centralized traffic engineering techniques would normally make use of global network state information, whereas distributed traffic engineering techniques typically rely on local network state information.

Traffic engineering techniques can also be classified in a number of other ways; open-loop versus closed-loop, depending on whether feedback about network state information is utilized; tactical versus strategic, depending on whether routing is performed to deal with short-term problems or with long-term objectives in mind; time-dependent versus

state-dependent versus event-dependent, depending on whether decisions are made periodically or depending on current network state or in response to specific network events.

2.1.5.4 Operational Complexity Vs. Control Granularity

Keeping the administration complexity low is an important consideration in deploying any traffic engineering mechanism. In the context of MPLS, on one hand more number of LSPs allow finer traffic control in the network. On the other hand, large number of LSPs also imply higher management overhead in terms of resource assignment, traffic policing, and service monitoring. A growing consensus in the networking community is to assign LSP paths to aggregate traffic flows rather than to individual streams or connections. This has the twofold advantage of reducing management overhead and having to deal with relatively stable long-term aggregate traffic.

The proposals in this dissertation are applicable in the context of all MPLS-like frameworks where traffic streams are aggregated into long-term flows at the edges and these aggregate flows are circuit switched through the interior links of the network. Thus, the resource provisioning algorithms in this dissertation are applicable in all contexts where MPLS is used to provide QoS guarantees, irrespective of whether an Intserv or Diffserv model is followed. As mentioned in Section 1.4, admission control and resource allocation decisions are made at three distinct levels of the network: *link-level*, *path-level*, and *network-level*. In the following sections, we review the related work at each of the three different levels of resource allocation, with specific focus on end-to-end delay guarantees and place the contributions of this thesis in context of prior works.

2.2 Link-level Mechanisms

QoS enforcement mechanisms at link-level can be classified in a number of ways [118]. Three primary classifications are (a) rate-based vs. non rate-based, (b) deterministic vs. statistical, and (c) work-conserving vs. non work-conserving.

2.2.1 Rate-based Vs. Non-rate-based Approaches

In rate-based QoS enforcement mechanisms, such as [83, 85, 120], the delay guarantees depend directly upon the amount of link bandwidth reserved for network flows. Rate-based approaches allow one to derive direct correlation between the amount of link bandwidth reserved for a flow and the worst-case delay bound experienced by the flow. This permits implementation of simple admission control schemes. Furthermore, a direct correlation enables us to provide distinct statistical delay guarantees on a per-flow basis.

On the other hand, non-rate-based approaches, such as [36, 111], decouple the delay bound for a flow from its service rate. Such approaches are capable of achieving higher link utilization than rate-based schemes. However, it is difficult to derive explicit resource-delay relationship among non-rate-based schemes, which makes the corresponding admission control process more complicated.

2.2.2 Work-Conserving Vs. Non-Work-Conserving Approaches

Various QoS-enforcement mechanisms can also be classified according to whether the packet scheduler remains busy or idle in the presence of work [118]. In work-conserving service disciplines, the scheduler does not remain idle whenever there is a packet to send. Thus work-conserving approaches ensure that link remains completely utilized in the presence of backlogged traffic. However, this may also result in increasing the delay-jitter experienced by packets traversing across multiple network nodes. Examples of work-conserving service disciplines include Delay-EDD [36], Virtual Clock [120], WFQ [83], SCFQ [44], and WF^2Q [11].

Non-work-conserving service disciplines attempt to bound the delay-jitter by assigning an eligibility time at which each packet can receive service. Thus, if all packets waiting for service are ineligible, the non-work-conserving scheduler will remain idle till one of the packets becomes eligible. Such an approach helps control the traffic distortions experienced by traffic inside the network at the expense of increasing the average delay and decreasing the average throughput. Examples of non-work-conserving service disciplines include Jitter-EDD [111], Stop-and-Go [43], Hierarchical Round Robin [62], and Rate-controlled Static Priority [119].

2.2.3 Deterministic vs. Statistical Approaches

Deterministic link-level approaches provide each flow with an absolute guarantee on the worst-case packet delay or minimum bandwidth. This is achieved by allocating sufficient resources at the link to ensure that the guarantees for each flow are never violated. Much of early research in packet scheduling and admission control [83, 85, 120, 36, 111, 11, 44] has focused on deterministic guarantees on bandwidth and packet delays. Zhang [118] provides a detailed overview of various deterministic service disciplines. Deterministic approaches provide absolute guarantee on performance, albeit at the cost of resource usage efficiency. For instance, under normal non-congested operating conditions, delays experienced by most packets traversing a link would rarely approach their worst-case delays. However, since flow resources are reserved to cover worst-case scenarios, the link resources remain underutilized most of the time.

Statistical approaches attempt to overcome the problem of resource underutilization encountered with deterministic approaches. Instead of providing absolute guarantees on performance, statistical approaches allow for small level of violations in guarantees in order to significantly reduce the amount of resources reserved for a flow. For instance, statistical admission control can exploit small tolerance to delay violations in order to significantly reduce the amount of resources reserved for each flow. Knightly and Shroff [64] provide a good overview of admission control approaches for link-level statistical QoS. In one of the early statistical approaches, Kurose[67] derived probabilistic bounds on delay and buffer occupancy of flows for nodes that use FIFO scheduling, which is commonly used in most network elements. Since FIFO schedulers inherently cannot enforce any form of QoS for flows traversing the link, subsequent approaches have focused on providing explicit statistical QoS guarantees to real-time flows. For instance, statistical approaches using rate-based scheduling under fluid traffic models have been proposed in [93, 32]. However, corresponding generalization to packetized traffic models are difficult to implement. On the other hand, statistical approaches using non-rate-based Earliest Deadline First (EDF) scheduling under packetized traffic models have been proposed in [3, 103]. EDF scheduling itself is simple to implement. However, due to strong interactions between flows sharing a link that is served by an EDF scheduler, the corresponding admission control process becomes complicated and it becomes difficult (if not impossi-

ble) to provide flow-specific statistical QoS guarantees.

An additional category, called measurement-based approach, attempts to provide statistical performance guarantees by dynamically measuring traffic characteristics of existing flows. Breslau *et al.* [18] provided a comparative study of several MBAC algorithms [88, 60, 38, 42, 27] under FIFO service discipline and concluded that none of them accurately achieve loss rate targets. These algorithms principally differ from each other in terms of their traffic measurement techniques and how these measurements are translated into statistical QoS guarantees.

2.2.4 Discussion

In contrast to the proposals in [67, 93, 32, 3, 103], our proposal in Chapter 4 to provide statistical delay guarantees is based on *packetized* rate-based schedulers that provide explicit resource-delay correlation. Additionally, our approach permits delay violation probability bound specification for each flow independently, and does not assume any specific traffic or loss rate models. While our proposal is also a measurement-based approach, an important difference of our algorithm with the existing MBAC schemes [88, 60, 38, 42, 27] is that the latter mainly focus on providing bandwidth or loss-rate guarantees but do not address real-time flows that require statistical delay bounds on a per-flow basis.

2.3 Path-Level End-to-end QoS

The next level of resource allocation is to map the end-to-end QoS requirements of the flow into per-physical-link QoS requirements along the path of a flow. In general, the QoS constraint partitioning problem is a special case of the general resource allocation problem that has been shown to be intractable [55, 92]. Thus all the solutions seek to find an approximate QoS constraint partition that satisfies certain optimization objectives. Previous solutions to this problem can be classified under two categories: fixed partitioning and *cost-based* partitioning.

2.3.1 Fixed Partitioning

The fixed partitioning solutions, as the name suggests, apportion the end-to-end QoS among constituent links of a path using some form of fixed division criteria. Simplest form of fixed partitioning is the Equal Allocation (EA) approach [79], that divides the end-to-end loss rate requirement equally among constituent links of a flow path. The focus in [79] is on optimizing the minimal link utility by equally partitioning the end-to-end loss rate requirements over the links. The performance of this scheme was shown to be reasonable for short paths with tight loss rate requirements but deteriorated for longer paths or higher loss rate tolerance. However, since EA does not take into account any information about link loads, it ends up treating a congested and a non-congested link with the same importance. Hence EA can potentially end up consuming excessive resources along bottleneck links.

The Proportional Allocation (PA) approach, proposed in [37], is a simple extension to the EA approach to QoS partitioning problem. In PA, the end-to-end QoS is apportioned among constituent links of a path in proportion to the current utilization of each link. The work in [37] applied PA for partitioning end-to-end QoS over links of a multicast tree. The performance of PA was shown to be better than EA since it accounts for different link loads.

In general, fixed partitioning approaches are also applicable in handling multiple simultaneous QoS requirements that are additive or multiplicative in nature. For instance, a real-time channel abstraction with deterministic and statistical delay bounds, based on a modified earliest deadline first (EDF) scheduling policy, has been proposed in [36]. This proposal uses the EA scheme to assign per-link delay and packet loss probability bounds.

2.3.2 Cost Function Based Solutions

Fixed partitioning schemes, although simple, suffer from the drawback that QoS division among links is not performed with any specific optimization objective in mind. Cost-function based solutions attempt to address this aspect by searching for a QoS-partition that minimizes or maximizes a global cost-function which is defined as the sum of local link costs. The cost-function can be designed as a measure of some network optimization objective, such as load-balance across the links of a path. Cost-function based QoS parti-

tioning algorithms over unicast or multicast paths have been proposed in [70, 34, 65]. The cost functions are assumed to be weakly convex in [70] and increase with the severity of QoS requirement at the link, whereas [34] addresses general cost functions. On the other hand, [92] addresses the problem in the context of discrete link costs in which each link offers only a discrete number of QoS guarantees and costs, such as in the framework of class-based discrete QoS in Diffserv. While the above approaches tackle the QoS partitioning problem in isolation, the combined problem of partitioning as well as QoS routing using cost-function based approach has been addressed in [46, 71] where the goal is to maximize the probability of meeting the QoS requirements of a flow.

2.3.3 Discussion

Our proposal in Chapter 5, called Load-based Slack Sharing (LSS), differs from the prior approaches in the following aspects. First, our proposal directly balances the loads on different links instead of indirectly addressing the objective via equal/proportional/cost function based partitioning. Secondly, the above proposals directly partition the entire end-to-end QoS over the links of the path. If a particular partition results in tighter delay requirement than the minimum possible at some link, then the proposal in [37] assigns the minimum delay at that link and then performs equal/proportional allocation over remaining links. With such an approach, the minimum delay assignment converts the corresponding link into a bottleneck, disabling all the paths that contain this link. In contrast, our LSS algorithm first identifies the *slack* in end-to-end QoS which quantifies the available flexibility in partitioning the end-to-end QoS. Instead of directly partitioning the end-to-end QoS, LSS proceeds to partition the slack which helps to prevent the formation of bottleneck links as long as non-zero slack is available. The principal conclusion in [79] is similar to ours, that is the key to maximize resource utilization is to reduce the load imbalance across network links.

For the cost-based algorithms in [70, 34, 65, 92] to be effective, one needs to carefully devise a per-link cost function that accurately captures the global optimization objective – in our case that of maximizing number of flows admitted by balancing the loads among multiple links of the path. As we will demonstrate in Section 5.6, the best cost function that we could devise to capture the load balancing optimization criteria, when used with

algorithms in [70], does not yield as high resource usage efficiency as the explicit load balancing approach proposed in this chapter. Instead of indirectly addressing the load balancing objective via a cost function, our LSS algorithm explores only those partitions that maintain explicit load balance among links. Furthermore, the cost-function based algorithms consider only single QoS dimension at a time. In contrast, our algorithm can handle multiple simultaneous QoS dimensions, such as both delay and delay violation probability requirements.

2.4 Network Level Route Selection

Internet originally evolved as a best-effort network in which routing was performed in a distributed and dynamic manner without any form of central control. In recent times, more sophisticated QoS routing and traffic engineering based mechanisms for route selection have evolved due to the need for greater control in tuning various aspects of network performance. In this section, we review the prior and current techniques in improving the operational performance of networks by means of route selection.

2.4.1 Dynamic Best-Effort Routing

The concept of dynamic best-effort routing was incorporated into the design of Internet as it evolved from the Arpanet in early days [75]. The basic idea behind dynamic routing is to make routing decisions based on the current link-state information. The most commonly used dynamic routing algorithms are based on the Shortest Path (SP) algorithm that selects the route having the smallest distance to the destination. Distance could be measured in terms of sum of link delays or the number of links along the path. Reachability information, such as delay or hop-count to various nodes, is periodically transmitted by a node to its neghbors. Adaptations to the SP algorithm, such as RIP [51] and OSPF [78], tend to use more general link metrics for calculating shortest paths. Link metrics could either be statically assigned or based on dynamic link-state information such as delay or packet loss rate. Such distributed routing protocols are highly scalable, reliable and have been the key factor in rapid growth of the Internet infrastructure.

2.4.1.1 Drawbacks of Dynamic Best-Effort Routing

On the flip side, simple best-effort routing algorithms do not provide sufficient control for tuning various aspects of network performance. Since this routing is best-effort in nature, dynamic best-effort algorithms inherently do not support QoS guarantees for traffic flows. Moreover, SP-based algorithms ignore the fact that links along the selected route might already be congested while some non-congested alternatives may never be considered because they do not lie along the shortest route. Limited traffic engineering, such as load balancing, can be performed by tuning link-state metrics. However frequently changing link-state metrics can result in oscillation of network congestion from one location to another leading to network instability. The drawbacks of legacy shortest-path-based dynamic routing protocols have been long recognized. Some variations of SP algorithm described below attempt to address these drawback to a limited extent.

2.4.1.2 Type-of-Service Routing

To address the need for service quality, a Type-of-Service (TOS) field was introduced in IP packet header that enables traffic to be classified into multiple traffic classes [81], such as low delay or high throughput. Each traffic class is associated with a different link-metric that is used to perform shortest-path routing for the corresponding traffic class. While ToS routing performs service differentiation between traffic classes to a limited extent, the basic drawbacks of shortest-path based algorithm remain the same within each traffic class. TOS-based routing has now been superceded by Differv based service model described earlier in Section 2.1.2.

2.4.1.3 Equal Cost Multi-Path

Equal Cost Multi-Path (ECMP) [78] algorithm attempts to overcome the load-imbalance problem encountered in shortest-path algorithm. If multiple path shortest paths exist to a destination, ECMP distributes the traffic equally among all available shortest paths. This is in contrast to traditional shortest-path algorithm that always selects only one among the multiple shortest paths. Traffic distribution could be at the granularity of either packet level or at the granularity of flow-level. Packet-level traffic splitting does not require

per-stream state maintenance, but can result in out-of-order packets. Stream-level traffic splitting maintains packet ordering, but can result in unmanageable state maintenance at routers. Furthermore, load sharing is limited in ECMP since it cannot be performed with routes that have costs greater than that of the shortest path.

2.4.2 QoS Routing

QoS routing techniques attempt to address the drawbacks of dynamic best-effort routing in providing performance guarantees to individual traffic streams. The goal of QoS routing is to find routes for individual traffic streams that satisfy the QoS requirements of each traffic stream. QoS metric can be of two kinds: bottleneck and end-to-end. Bottleneck QoS requirements are those in which path availability is determined by the bottleneck QoS available at any link along the path. For instance, bandwidth requirement of a stream can be considered as bottleneck metric because a route can be selected only if the smallest available bandwidth among all the links in the path is sufficient to satisfy the bandwidth required by the stream. On the other hand, with end-to-end QoS metrics, path availability is determined by combined QoS of all links in the path. For instance, delay is an example of end-to-end QoS metric since the total packet delay is determined by the sum of individual link delays. Similarly, delay violation probability and packet-loss rate are other forms of end-to-end QoS metrics. QoS routing schemes can be classified based on whether single or multiple QoS metrics are considered.

2.4.2.1 Single Metric Schemes

In general bottleneck QoS merics, such as bandwidth, can be handled by topology filtering approach [7] in which only those links in the network with sufficient residual capacity to satisfy the bandwidth requirements of the flow are used in shortest route computation. The Widest Shortest Path (WSP) algorithm [47] improves upon basic topology filtering by selecting the feasible shortest route that has maximum residual capacity at the bottleneck link. While marginally better than the SP algorithm, WSP still restricts its choice of routes to those with fewest number of hops. In order to account for varying link loads, Plotkin [87] proposed an improvement to the SP algorithm by assigning link weights that dynamically increase with loads on the link. Thus one can avoid more loaded links in the

network while performing route selection.

The delay constrained routing problem does not admit to simple topology filtering solution, but requires some form of search for paths that satisfy the end-to-end delay requirements. Shin and Chou [102] proposed routing scheme that discovers a route satisfying end-to-end delay by flooding route messages from source to destination. Intermediate routers only forward those messages that hold better accumulated delay than earlier messages and the first message received by the destination determines the delay-constrained route. Selective probing [22] can be used to reduce the routing overhead of Shin-Chou algorithm. In the context of ATM networks, Hou [54] proposed another flooding based routing algorithm to discover delay-constrained routes.

Variations of the two single-metric problems of bandwidth-constrained and delay-constained route selection have been studied by Guerin and Orda [47] in which information about network state is imprecise and the goal is to find a path that has highest probability of satisfying a new stream's bandwidth or delay requirement. The bandwidth-constrained problem is addressed using the shortest path algorithm where link-weights depend on probability of satisfying the bandwidth requirement. The delay-constrained problem is addressed by a heuristic that transforms the end-to-end delay constraints into per-link constriants such that every link has an equal probability of satisfying local delay bound. The heuristic then finds the path with best multiplicative probability of satisfying the end-to-end delay bound.

2.4.2.2 Multiple Metric Schemes

Several hybrid routing problems can be derived by combination of multiple QoS routing constraints. For instance, the bandwidth-delay constrained route selection problem is addressed by Wang and Crowcroft [113]. Links with smaller available bandwidth than the required bandwidth are first eliminated by means of topology filtering and the smallest delay route is selected from the residual graph. Chen and Nahrstedt [21] have proposed a heuristic solution for the problem of finding a feasible route that requires a bound on both end-to-end delay and cost of the path. While the original problem is NP-complete, the heuristic solves the problem in polynomial time by mapping the original cost and delay of every link, which are unbounded real numbers, to bounded integers. The new prob-

lem can be solved by an extended SP or Bellman-Ford algorithm. The bandwidth-delay constrained and delay-cost constrained route selection problems are specific instances of the more general k-constrained route selection problem, which is known to be NP-hard [59, 114]. Yuan and Liu [117] have proposed two heuristic solutions to the general k-constrained route selection problem that is based on an extended Bellman-Ford algorithm.

The Restricted Shortest Path (RSP) problem deals with finding a minimum cost path through the network that satisfies multiple end-to-end constraints, where each link has a constant cost. This problem is also known as the Multi-Constrained Optimal Path Selection (MCOP). RSP has been shown to be NP-hard [40]. Hassin [50] gave a pseudo-polynomial approximation algorithm that is based on the technique of rounding and scaling [96]. The solutions in [86, 92] are all based on the approximation to the RSP problem given in [50]. The Delay Constrained Unicast Routing (DCUR) [97] routing addresses the problem of finding a delay-constrained path that has minimum cost. Routing is accomplished by utilizing both least-cost and least-delay vectors maintained at each node by means of distance vector protocol. Routing loops can occur in DCUR and [106] proposes a loop-free version of the algorithm. Another approach to solving RSP is to find the k-shortest paths, with the expectation that one of the paths is feasible and near-optimal [33, 48]. This approach provides a simple, albeit somewhat computationally expensive, solution to the RSP problem.

In the above solutions, the link delay and cost are assumed to be fixed values. On the other hand, [34, 74], as well as the work in this dissertation, consider the case where the link costs vary with the amount of resources allocated at the link.

2.4.3 Traffic Engineering Based Approaches

The inability of dynamic best-effort routing schemes to provide QoS guarantees to network flows and to control network performance led to development of QoS routing schemes. QoS routing schemes are designed to satisfy the performance requirements of individual traffic streams and act on short time-scales. This prevents long-term planning constraints, such as traffic load-balancing criteria, to be explicitly considered in route-selection decisions. Additionally, these schemes typically require per-stream state to be stored on inter-

mediate routers/switches - an impractical requirement in network infrastructure backbone that deals with millions of traffic streams.

To address these issues, Traffic Engineering (TE) based solutions go one step further towards addressing long-range network management goals while satisfying customer's traffic perforamnce requirements. Specifically, TE-based schemes provide the ability to select routes that are different from the shortest route to achieve a better long-term balance in network load. Unlike QoS routing schemes which operate at the granularity of short-term individual traffic streams, TE-based solutions operate at the granularity of long-term aggregate traffic flows. MPLS, discussed earlier in Section 2.1.4, is one of the enabling technologies in this direction. In this section, we review some of the traffic engineering based schemes that enable route selection with QoS guarantees and network performance control.

2.4.3.1 Enhanced Link-State Routing Protocols

One approach to TE-based route selection is to enhance standard interior gateway protocols (IGPs) to accomodate traffic engineering considerations [116, 15]. For instance, an IGP such as IS-IS [1] can be enhanced [100, 104] to propogate TE-specific link attributes, such as reservable link bandwidth, in addition to normal link state attributes. Furthermore, other policy specific information, such as link affinities (grouping), can be propogated as well. A traditional IGP constructs router's forwarding tables using just network topology information and normal link-state attributes. On the other hand, an TE-enhanced IGP would additionally take into account resource requirements of LSPs that need to be provisioned, such as bandwidth and latency required, permissible links etc. Such an enhanced IGP achieves network load-balance by utilizing the information about maximum reservable link resources and QoS requirements for each LSP being provisioned. However, specific mechanism for LSP route computation depends upon the network optimization objectives of the service provider, which is captured by means of enhanced IGP link-state attributes.

2.4.3.2 Minimum Interference Routing Algorithm

The Minimum Interference Routing Algorithm (MIRA) [66] is an online technique for routing bandwidth guaranteed flows that improves upon earlier techniques by minimizing the interference of routes with critical links of the network. MIRA was the first algorithm that used the knowledge of all ingress-egress pairs in the network to make intelligent routing decisions. Similar to our proposal, MIRA uses the notion of link criticality to determine which links to avoid in selecting a route. However the criticality of a link in MIRA depends upon the number of mincuts between the different ingress-egress pairs that the link belongs to. While this notion proves useful in avoiding links that impact the maxflow of a large number of ingress-egress pairs, it proves to be insufficient in several situations such as the Parking Lot topology in Chapter 6. MIRA forms the heart of a software system called Routing and Traffic Engineering Server (RATES) [4] that obtains link state information from OSPF topology database and uses MIRA for online selection of bandwidth guaranteed routes.

2.4.3.3 Profile Based Routing

Profile Based Routing (PBR) [107] Selects routes for bandwidth guaranteed flows by using measured profile of past traffic as a rough predictor for future traffic distribution. The offline phase of PBR uses the traffic profile to solve a multi-commodity network flow problem and derives a reduced graph that consists of advanced reservations for each traffic class on every link. The reduced graph represents the expected bandwidth demand from each traffic class on every link of the network and is used to bound the medium-term bandwidth allocation for each class. The online phase of PBR uses SP algorithm on the reduced graph to admit new bandwidth guaranteed route in that class. PBR ensures long-term resource usage efficiency by rejecting flows that do not fall within the measured profile of past traffic, thus overcoming the basic shortcomings of MIRA algorithm. However, it comes at the cost of having to pre-allocate resources for each traffic class. Specifically if pre-allocated reservation for one class is exhausted then new reservation requests in that class would be rejected even if unused pre-allocated bandwidth exists in another class.

2.4.4 Discussion

An important difference between our proposal in Chapter 6, called Link Criticality Based Routing (LCBR), and the shortest path and traffic engineering approaches is that the latter mainly address flows that require average bandwidth guarantees and assume that the other QoS parameters such as delay bounds and delay violation bounds can be expressed in terms of average bandwidth requirements. In contrast, our LCBR algorithm addresses flows that have separate requirements for delay bounds, delay violation probability and average bandwidth. Specifically we make an explicit distinction between the bandwidth requirement dictated by delay bounds and the long-term average bandwidth requirements of a flow. The former is used by link-level schedulers at run time to bound the delay experienced by packets of a flow while the latter is used by traffic regulators at the ingress of the network to enforce long-term bandwidth usage limits. In comparison to [66], our definition of link criticality accounts for the impact of the link on different source destination pairs, irrespective of whether the link belongs to any mincuts. Another distinguishing characteristic of our work is that we account for the inter-dependencies between route selection decisions, intra-route resource allocation decisions and link-level resource availability. For instance, the way one partitions the end-to-end delay of a flow along the links of a route directly impacts the load on the links of the route. Thus our route selection algorithm takes into account the effect of end-to-end delay partitioning employed along a route in deciding routes.

The approach taken in LCBR is similar to the k-shortest paths approach taken in [33, 48]. This is mainly due to the ease with which multiple QoS constraints, such as delay and delay violation probability, as well as the simultaneous primary-backup route selection problems can be handled in the k-shortest paths framework. However, our specific focus in LCBR is to combine the k-shortest path approach with traffic engineering techniques in order to improve resource utilization efficiency of the network. In contrast, the approaches taken in [34, 74] require integer cost functions and delays and are not designed to handle multiple QoS requirements or simultaneous primary-backup route selection problems.

2.5 Fault Tolerant QoS Routing

Fault tolerant QoS routing deals with ensuring that QoS guarantees provided to network flows continue to hold even in the presence of network faults, such as link failures or node failures. There is a wide body of work in the area of network fault-management and we review some that are relevant to our proposal.

2.5.1 Reactive vs. Proactive Techniques

Reactive techniques [10] react to a network-fault only when a fault occurs. For instance, when a network link/node fails, unreserved network resources are re-allocated among the affected flows. The advantage of reactive schemes is that additional resources do not need to be set aside in advance to deal with failure scenarios, leading to higher network resource utilization efficiency. The disadvantage is that some flows affected by a failure may need to be rejected due to lack of adequate unreserved resources in the network. Also, the time to recover from a failure could be long especially under heavy loads. *Proactive* techniques [9, 30] pre-allocate network resources in order to provide guaranteed and fast recovery from network faults, albeit at the expense of resource usage efficiency.

2.5.2 Local vs. Global Techniques

Local schemes recover from failure by (re)allocating resources in the neighborhood of the fault. As a result they tend to be simpler, but do not make efficient use of network resources. For instance, Single Failure Immune (SFI) [122] method reserves additional resources in the vicinity of each real-time channel and these resources are used as detours in the event of a failure. *Global* schemes [30], on the other hand, perform more efficient resource (re)allocation in an end-to-end manner to cover fault scenarios.

2.5.3 Resource Aggregation Techniques

Resource Aggregation for Fault Tolerance (RAFT) [30] is a proactive and global fault-recovery scheme in which bandwidth is reserved along a secondary route in case the primary route fails. Given a secondary route, backup reservations are aggregated along

the route for flows that do not interfere with each other, thus reducing the overhead of providing fault-tolerance. However, the route selected for backup resource aggregation is entirely up to the routing protocol. Dependable Real-Time Protocol (DRTP) [49] provides a slightly different form of resource aggregation by overbooking backup reservations for flows along backup routes. Multiple backup routes can be set up to achieve different levels of reliability. However, due to overbooking, there is a probability that some affected flows might be rejected after a failure. Kim et. al [63] built upon DRTP and presented methods for selecting primary and backup routes using expanded link-state database and bounded flooding.

2.5.4 Optical Network Restoration

A large body of work in the area of optical network restoration deals with the problem of handling network faults in optical networks. Most works fall under the category of reactive schemes and focus on restoration of all lightpaths in the event of a physical link failure. Local schemes [90, 89, 39] recover from a link failure by directly restoring the failed physical link. Global schemes [90, 89, 29] restore each lightpaths independently by selecting alternative end-to-end paths. Some of the proactive fault-management techniques are as follows. Modiano and Narula [77] have addressed a slightly different problem of survivable routing of logical links (lightpaths) on a physical network topology such that the logical topology remains connected in the event of any single physical link failure. They proved the necessary and sufficient conditions for a routing to be survivable and used this condition to formulate the problem as an integer linear program. Crochat and Boudec [26] have explored routing techniques with the goal of minimizing the number of source destination pairs that would get disconnected in the event of physical link failure.

2.5.5 Discussion

The LCBR algorithm for backup route selection, that we propose in Chapter 6, falls under the category of proactive and global fault management. In LCBR, we select primary and backup routes such that resource usage efficiency of the network is maximized and then apply backup resource aggregation that is similar to RAFT [30], along the selected

routes. In contrast to [49, 63], our proposal provides guaranteed recovery from single failures. Our criteria for selecting primary and backup routes includes the aggregation information, among other factors, to determine relative importance of different routes.

2.6 Summary

In this chapter, we presented the background of research related to this dissertation. We discussed different frameworks for providing QoS in the present day Internet and saw how MPLS is emerging as an enabling technology for different QoS frameworks. In addition, we reviewed the past research in network resource allocation for time-sensitive network traffic at the link level, path level and network level, and placed our contributions in their context.

Undeniably, the research area of network resource management is large. With the continuing growth of network applications that generate time-sensitive network traffic, there is also a growing need for mechanisms that support real-time traffic and for efficiently managing network resources in this environment. In the subsequent chapters, this dissertation attempts to address some of the issues that arise in efficient network resource management in the presence of real-time network traffic.

Chapter 3

Network Model and Traffic Scheduling

In this chapter, we first formulate the model of the network and justify the assumptions underlying our proposals. Next we review some of the basic concepts in network traffic scheduling relevant to this dissertation. Finally we look at the typical components of end-to-end delay in voice packet networks.

3.1 Model of the Network

We consider a network of routers and links that are under the administrative control of an NSP. A subset of the routers are known to be ingress and egress points for network traffic and these are typically placed at the NSP's point-of-presence locations, i.e. places where NSP's network interfaces with customer sites.

3.1.1 Flow Specifications

A *flow* is a reserved path that is set up through the underlying network to carry a real-time traffic stream from an ingress router to an egress router. By definition, flows are unidirectional. Each flow is identified by a unique label i. Every request for setting up a flow F_i is specified by $(s_i, d_i, \rho_i^{avg}, D_i, P_i, \sigma_i)$ where s_i and d_i are ingress and egress routers for the flow, ρ_i^{avg} is its average bandwidth requirement, D_i is its end-to-end delay requirement, P_i is its end-to-end delay violation probability requirement and σ_i is the maximum burst size. For instance, if $\rho_i^{avg} = 256$kbps, $D_i = 200ms$, $P_i = 10^{-3}$, and

$\sigma_i = 2000$ bytes, it means that the flow needs to carry network traffic with an average data rate of 256kbps, no more than a fraction 10^{-3} of packets belonging to the flow can experience a delay greater than $200ms$, and the maximum data burst at any instant would be limited to 2000 bytes.

Flows are long-lived connections possibly lasting several months at a stretch. Hence a flow is set up to carry customer's aggregated traffic rather than a single small individual flow. We assume that all the small flows in the aggregate have similar delay and delay violation requirements and that the bandwidth of flow is sufficient to accommodate the cumulative requirements of individual flows. For instance a flow could be a 256kbps VoIP trunk that multiplexes traffic from several individual VoIP connections, each of which requires an end-to-end delay of 150ms and 16kbps bandwidth.

3.1.2 Centralized Resource Management

As shown in Figure 3.1, network resources are allocated by a *central resource manager* (CRM) that has knowledge of both static and dynamic information about the entire network. Static information consists of the network topology such as connections and routers (assuming that the topology does not change over very short time-scales). Dynamic information includes link-state parameters such as residual capacity, buffer space, and delay distribution of packets at each link in the network. We assume that a distributed information collection mechanism, such as the one proposed by Lin et. al. [69], is in place to collect and deliver the dynamic link-state information to CRM. Typically, NSPs have mechanisms to centrally monitor the health of physical links and routers in their network; hence this is a reasonable assumption to make.

In addition to flows being long-lived connections, we assume relatively large inter-arrival times between successive flow requests at the CRM. Thus the dynamic link-state information collected by the CRM has sufficient time to stabilize between resource allocations for successive flow.

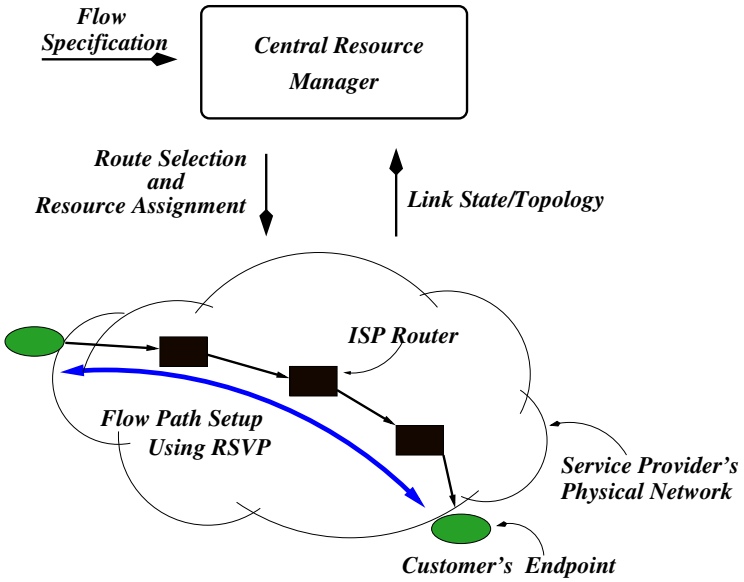

Figure 3.1: Centralized resource management. A central server collects dynamic link-state and static topology information from the physical network. It uses this information to select route and assign physical resources for newly arriving flow requests. The assigned resources are reserved along the selected route using a signaling protocol such as RSVP.

3.1.3 On-line Resource Provisioning

In our model, the flow requests are processed by the CRM one at a time as they arrive and the resource allocation algorithm has no a priori knowledge of future request arrivals. This process is known as *on-line provisioning*. In contrast, *off-line provisioning* assumes that all possible flow requests are known in advance at the time of resource allocation, which is not a practical assumption in real networks.

3.1.4 Explicit Route Setup Using Signaling

We assume that a signaling protocol, such as the Resource Reservation Protocol (RSVP) [121], is used to explicitly reserve resources for a flow along a uni-directional route from the source to destination. RSVP supports resource reservation for unicast as well as multicast traffic. A flow spec specifies the QoS to be provided to a new flow and a filter spec specifies the traffic to which QoS should be applied. RSVP does not provide any routing support and relies on routing table information at intermediate routers for path selection. Awduche et. al. [7] have proposed extension to RSVP protocol in order to support explicit routing in RSVP path messages.

3.2 Network Traffic Scheduling

In this section, we briefly discuss the basic concepts in network traffic shaping, traffic scheduling and bounds on packet delay.

3.2.1 Traffic Regulator

Before the traffic belonging to a flow F_i enters the network at an ingress node, it traverses a traffic regulator that shapes the connection traffic according to shaping parameters agreed upon in advance between the customer and the NSP. We consider a popular traffic regulator called the *token bucket*. Let $A_i(\tau)$ be the amount of flow F_i traffic that traverses through a (σ, ρ) token bucket in time interval τ, where σ is the burst size and ρ is the average data rate permitted by the traffic regulator. The following relationship holds:

$$A_i(\tau) \leq \sigma + \rho \times \tau \tag{3.1}$$

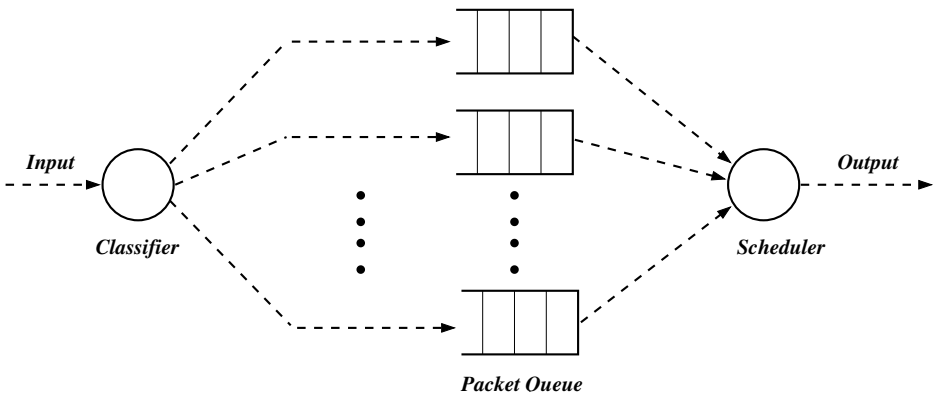

Figure 3.2: Different stages of packet processing at a link. Each packet is first examined by a classifier, then queued at a flow specific queue, from where it is picked up by a link scheduler and transmitted over the output link.

In other words, if the flow F_i was greedily trying to use bandwidth, then the token bucket would regulate F_i's traffic by allowing a one-time burst of size σ bits followed by a steady rate output of ρ bits per second.

3.2.2 Rate-based Link Schedulers

Figure 3.2 shows the different components of a link which handle flow traffic. Each packet is first examined by a classifier, then queued at a flow specific queue, from where it is picked up by a link scheduler and transmitted over the output link.

The job of a link scheduler is to dynamically prioritize the transmission of packets belonging to different flows over a common link. We assume that packets at all links in the network are serviced by *rate-based link schedulers*. Rate-based schedulers explicitly use bandwidth reservations of flows in determining inter-packet priority. We will consider the Generalized Processor Sharing (GPS) [85] schedulers, in which network traffic is modeled as a fluid flow (as opposed to discrete packets). Each flow F_i that shares a link l with other flows is assigned a weight $\phi_{i,l}$. Let $W_{i,l}(\tau)$ be the amount of F_i traffic serviced by the scheduler in time τ. Then the following relation holds between any two flows F_i and F_j sharing the link l.

$$\frac{W_{i,l}(\tau)}{W_{j,l}(\tau)} = \frac{\phi_{i,l}}{\phi_{j,l}} \tag{3.2}$$

In other words, link capacity is divided among flows in proportion to their assigned weights. GPS treats traffic as fluid flows, whereas real-world traffic consists of discrete packets. A popular packetized approximation of GPS is known as Weighted Fair Queuing (WFQ) [28, 85] scheduler. Conceptually, the WFQ scheduler maintains a virtual time and one queue per flow. During admission control each flow F_i is assigned a bandwidth reservation $\phi_{i,l} = \rho_{i,l}$. A *busy period* is an interval of time when the scheduler always has some packets to service. Let time $t_1 = 0$ be the time instant when the first packet arrives in a busy period and t_j be the time when j^{th} event occurs (arrival or departure of a packet) in a busy period. Further, let B_j represent the set of backlogged flows that have packets waiting in their queue for transmission at the time of j^{th} event. Then the *virtual time* $\mathcal{V}_l(t)$ at link l progresses as follows.

$$\begin{aligned} \mathcal{V}_l(0) &= 0 \\ \mathcal{V}_l(t_{j-1} + \tau) &= \mathcal{V}_l(t_{j-1}) + \frac{\tau}{\sum_{i \in B_j} \rho_{i,l} / \sum_{all\ m} \rho_{m,l}} \end{aligned} \tag{3.3}$$

Thus the virtual time increases at the marginal rate at which backlogged sessions receive service. At time $a_{i,k}$, the k^{th} packet of flow F_i, having size $L_{i,k}$, arrives. This packet is stamped with a virtual finish time $F_{i,k}$ given by the following expression.

$$F_{i,k} = max\{F_{i,k-1}, \mathcal{V}_l(a_{i,k})\} + L_{i,k}/\rho_{i,l} \tag{3.4}$$

Packets are served by the scheduler according to the increasing order of virtual finish times $F_{i,k}$. It can be shown [85] that the worst-case queuing delay $D_{i,l}^{wc}$ experienced at link l by any packet belonging to flow F_i under WFQ service discipline is given by the following expression.

$$D_{i,l}^{wc} = \frac{\sigma_{i,l}}{\rho_{i,l}} + \frac{L_{max}}{\rho_{i,l}} + \frac{L_{max}}{C_l} \tag{3.5}$$

where $\sigma_{i,l}$ is F_i's burst size at link l, L_{max} is the maximum packet size, $\rho_{i,l}$ is the reservation for F_i at link l, and C_l is the total capacity of link l. The first component of the delay is fluid fair-queuing delay, the second component is the packetization delay and the third component is scheduler's non-preemption delay. The delay bound of Equation 3.5 also holds in the case of Virtual Clock [120] and WF^{2}Q [11] schedulers.

We are interested in rate-based schedulers since, in their case, the relationship between delay bound and the amount of bandwidth reserved for a flow can be explicitly

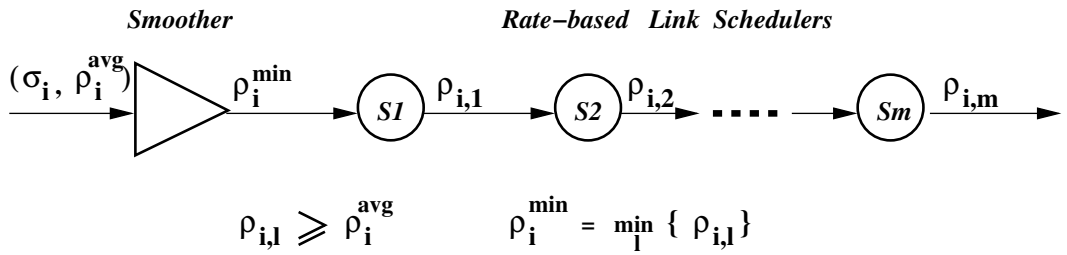

Figure 3.3: Network components along the path of a multi-hop flow. Flow F_i that has (σ_i, ρ_i^{avg}) traffic characterization passes through a smoother followed by a set of rate-based link schedulers. F_i's service rate at each link l is $\rho_{i,l} \geq \rho_i^{avg}$ and the bursts are smoothed before the ingress at a rate of $\rho_i^{min} = \min_l\{\rho_{i,l}\}$.

specified. Furthermore, as we will see in Chapter 4, rate-based schedulers enable us to differentiate among flows in terms of their statistical delay violation probability requirements. In contrast, for non rate-based schedulers, such as Earliest Deadline First (EDF), the resource-delay relationship for each flow is difficult to determine, which in turn makes the admission control process more complicated. Also, due to strong interactions among competing streams in non-rate-based schedulers, it is difficult to differentiate among the streams in terms of individual delay violation probabilities. Hence, even though non rate-based schedulers can potentially provide higher link utilization, it is difficult to provide statistical guarantees (such as bounds on loss rate or delay-violation probability) on a *per-flow basis*.

3.3 End-to-end Delays Bounds

We consider the framework of smoothing at the network ingress and bufferless multiplexing in the network interior as advocated in [93, 41]. Specifically, as shown in Figure 3.3, a regulated flow F_i first traverses a traffic smoother followed by a set of rate-based link schedulers at each of the intermediate links along its multi-hop path. The first component smoothes the burstiness in F_i's traffic before the ingress node. Each rate-based scheduler at intermediate link l services the flow at an assigned rate $\rho_{i,l} \geq \rho_i^{avg}$. The manner in which rates $\rho_{i,l}$ are assigned will be described later in Chapter 5. The smoother regulates

flow F_i's traffic at a rate $\rho_i^{min} = \min_l\{\rho_{i,l}\}$ i.e., at the smallest of per-link assigned rates. Since F_i's service rate at each link scheduler is greater or equal to smoothing rate at the ingress, F_i's traffic does not become bursty at any of the intermediate links. Completely smoothing F_i's traffic before the ingress node has the advantage that it allows the interior link multiplexers to employ small buffers for packetized traffic. Additionally, as shown below, it permits decomposition of flow's end-to-end delay requirement D_i into delay requirements at each network component along the path.

We now proceed to identify different components of end-to-end delay experienced by a F_i. The first component is the *smoothing delay*. Since the input traffic from flow F_i has a maximum burst size of σ_i, the output rate of the smoother is $\rho_i^{min} = \min_l\{\rho_{i,l}\}$, and the output burst size of the smoother is 0, the worst-case delay experienced by a packet at the smoother can be shown to be as follows [41].

$$D_{i,s} = \sigma_i / \rho_i^{min} \tag{3.6}$$

The second component of end-to-end delay is the accumulated *queuing delay* at intermediate links. Since our network model employs bufferless multiplexing at interior links, the input burst $\sigma_{i,l} = 0$ at each link l and $D_{i,l}$ in Equation 3.5 reduces as follows.

$$D_{i,l} = \frac{L_{max}}{\rho_{i,l}} + \frac{L_{max}}{C_l} \tag{3.7}$$

In Figure 3.3, the end-to-end delay bound D_i for flow F_i over a h-hop path, where each link is served by a WFQ scheduler, is given by the following expression [83, 84].

$$D_i = \frac{\sigma_i}{\rho_i^{min}} + \sum_{l=1}^{h} \left(\frac{L_{max}}{\rho_{i,l}} + \frac{L_{max}}{C_l} \right) \tag{3.8}$$

where $\rho_{i,l} \geq \rho_i^{avg}$ and $\rho_i^{min} = \min_l\{\rho_{i,l}\}$. In other words, end-to-end delay is the sum of traffic smoothing delay and per-link queuing (packetization and non pre-emption) delays.

3.4 Typical End-to-end Delays in Packet Voice Networks

In this work, we assume without loss of generality that a real-time flows carry packet voice traffic (VoIP) that is aggregated from several smaller VoIP connections. As per International Telecommunications Union (ITU) recommendations, acceptable end-to-end delays

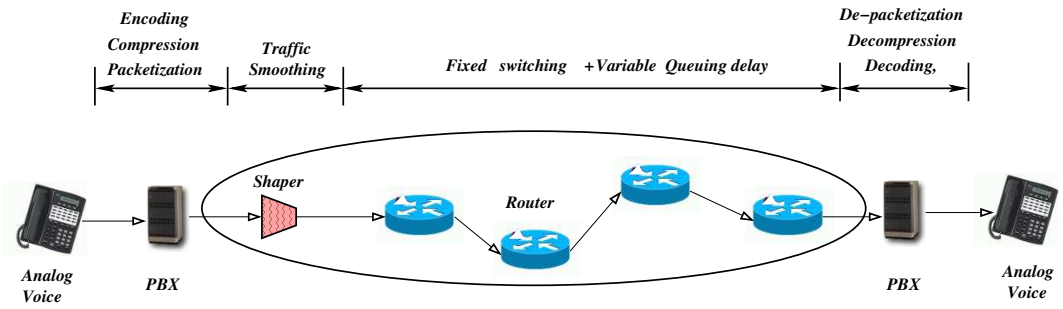

Figure 3.4: Components of end-to-end delay in packet voice networks.

for voice traffic should be less than 150ms. Figure 3.4 shows the fixed and variable components of this end-to-end delay in packet voice networks. As documented in [110], the delay in encoding, compression, decompression, and decoding of voice signal between analog telephone and digital PCM formats is typically around 20ms. The delay in packetizing encoded/compressed speech is around 30ms. Major NSPs (such as Level-3, AT&T, Sprint) report the typical round trip propagation times (RTT) of 60 to 80ms between east coast and west coast of United States. Thus we can assume that typical one-way end-to-end fixed switching delays across the network would be half i.e, around 30 to 40 ms. This gives us a total end-to-end fixed network delay of around 80 to 90ms. Thus, out of the 150ms end-to-end delay budget for voice traffic, we are left with 60 to 70ms of variable delays (smoothing and queuing) that can be partitioned across the underlying links.

3.5 Summary

In this chapter, we outlined a model of network under the administrative control of an NSP where resources are allocated for flows by a central resource manager. Flows are long-lived connection paths that carry aggregate real-time traffic. Explicit signaling protocol, such as RSVP, is used for setting up flows between pairs of ingress-egress nodes. Flow traffic is characterized at the ingress by its burst-size and average rate and scheduled in the network interior using rate-based link schedulers. Finally, we reviewed standard results for delay bounds in single link and multi-hop network settings and looked at the typical end-to-end delays values in voice networks.

Chapter 4

Link-level Statistical Admission Control

4.1 Problem Statement

In this chapter, we start with the problem of reserving resources at the level of a network link for flows that carry real-time network traffic. Real-time traffic requires a tight bound on delay experienced by each packet. However, typical network applications, that generate real-time traffic, can tolerate occasional packets that exceed their delay bounds or get dropped by the network in transit. For instance, VoIP streams can tolerate up to 10^{-3} fraction of their packets experiencing excess delays or losses without perceptually affecting audio quality.

A simple resource allocation approach is the *deterministic* approach that allocates sufficient resources to each flow in order to cover for the worst case such that the packets never encounter any excess delay in the lifetime of the flow. However, the downside of deterministic admission control is that it comes at the cost of severely under-utilizing a link's resources. Two statistical effects in aggregated real-time traffic could be exploited to reduce the bandwidth requirement of flows and increase the link utilization.

- **Tolerance to delay violations:** Many multimedia applications, such as voice over IP (VoIP), video conferencing, streaming media and content distribution, can tolerate certain levels of excess delays or packet losses in their real-time traffic [112]. If 99.9% of the packets are observed to experience only 50% of their expected worst-case delay, a statistical resource allocation approach can potentially reduce the flow

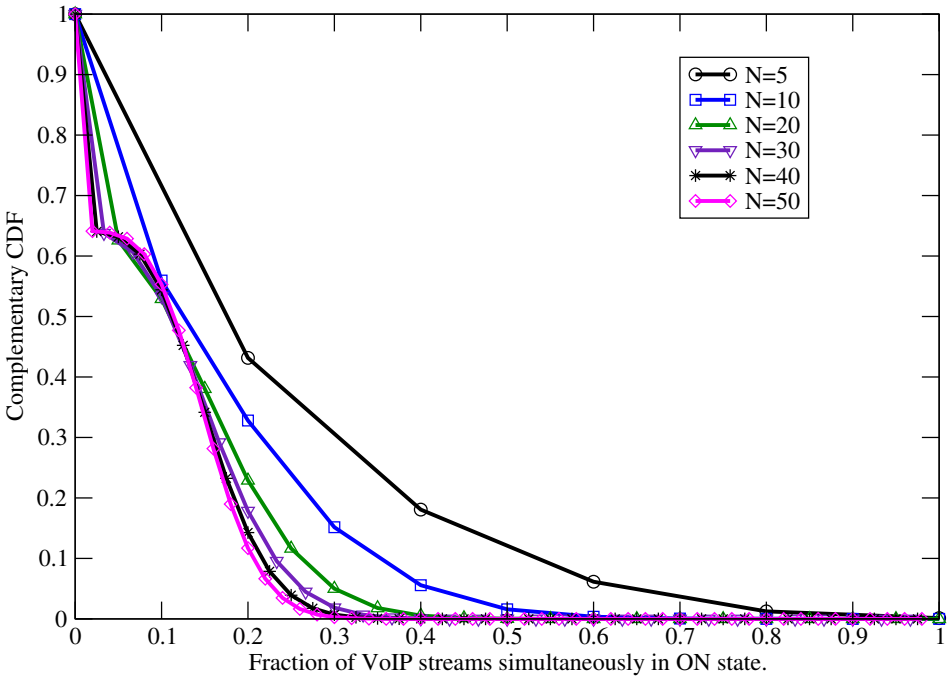

Figure 4.1: Complementary CDF of the fraction of VoIP streams in ON state simultaneously. N is the number of VoIP streams in aggregate flow. For $N \geq 20$, rarely more than 40% of the streams are in their ON state simultaneously.

reservation by almost half compared to deterministic approach. Tolerance to delay violations gives admission control algorithms the flexibility to reduce resource reservation for real-time flows.

- **Statistical multiplexing:** Due to statistical multiplexing, not all the real-time streams on a link carry traffic at their peak load simultaneously. The consequence of this multiplexing is that packet delays rarely approach worst-case delays bounds that are based on all streams transmitting at their peak rate simultaneously. To illustrate this multiplexing effect, we aggregated packet traces for different number of recorded VoIP streams (described later in Section 4.4). Each VoIP stream is an ON-OFF packet source with an average data rate of about 13kbps and a peak data rate of 34kbps during ON periods. Figure 4.1 shows the complementary CDF of the fraction of VoIP streams in each aggregate that are simultaneously in their ON state. We observe that when 20 or more streams are aggregated, half the time less than

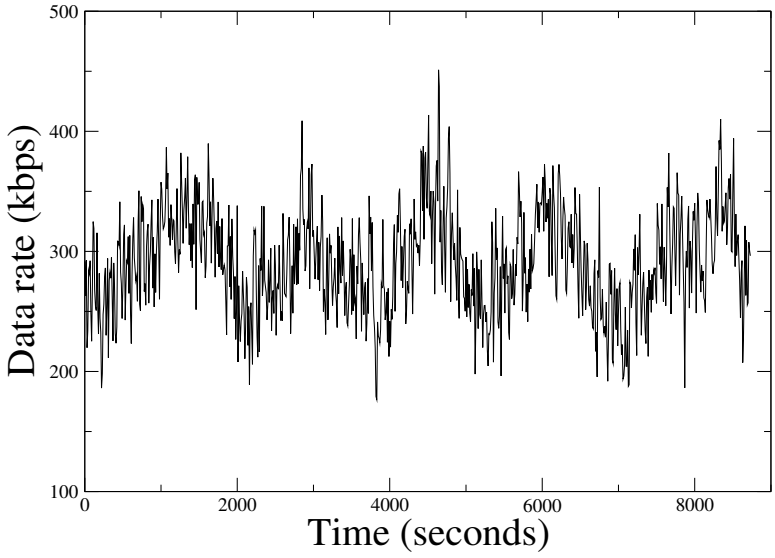

Figure 4.2: Data rate in each 10 second slot during the lifetime of an aggregate of 20 VoIP streams. The data rate never approaches the possible peak of 680 kbps (20×34 kbps).

12% of the VoIP streams are in their ON state simultaneously and its almost never the case that more than 40% of the streams are simultaneously active. Figure 4.2 shows that the data rate in each 10 second slot for the aggregated trace of 20 VoIP streams never approaches the peak rate of 680 kbps. Instead, the observed data rate has only occasional highs above 400 kbps. This shows that the total bandwidth allocated to an aggregated set of streams need not be equal to the summation of their peak bandwidth requirements.

Most of the prior research on statistical delay guarantees [67, 93, 32, 3, 103] have focused on service models based on fluid flows and do not allow flows to specify individual delay violation probability requirements. Additionally, statistical guarantees in these approaches are based on specific traffic and loss rate models. On the other hand prior measurement-based approaches, such as [88, 60, 38, 42, 27, 14] mainly focus on providing statistical bandwidth or loss-rate guarantees but do not address real-time flows that require statistical delay bounds on a per-flow basis.

In this chapter, we propose a novel link-level approach, called *Delay Distribution Measurement* (DDM) based admission control, that exploits statistical multiplexing among

real-time flows in order to maximize the number of flows admitted with statistical delay guarantees. Unlike prior proposals, DDM provides a distinct per-flow statistical delay guarantee to each real-time flow sharing the link. We develop a novel quantitative framework to exploit the well known fact that the actual delay experienced by most packets of a real-time flow is usually far smaller than its worst-case delay bound requirement. Flows that can tolerate more delay bound violations can reserve less resources than those that tolerate less, even though they have the same delay bound requirement. DDM is the first approach that applies the concept of measurement-based admission control to provide statistical delay guarantees on a per-flow basis.

The main challenge in providing statistical delay guarantees is to determine the mapping between delay bound, delay violation probability and resource requirements. DDM dynamically measures the service delay of each packet, computes the ratio between the actual service delay and the worst-case delay that the packet could experience, and derives a delay ratio distribution. Given the delay bound and delay violation probability for a new flow, DDM uses this dynamically measured delay ratio distribution to derive the bandwidth reservation required to achieve a given delay bound and delay violation probability bound. Once the DDM algorithm reserves certain bandwidth for a flow, a rate-based packet-by-packet scheduler (such as WFQ[28, 85] or Virtual Clock[120]) guarantees the assigned bandwidth share at run-time.

The rest of this chapter is organized as follows. In Section 4.2, we derive the resource correlation function that maps the delay and delay violation probability bound requirements of a flow to its resource requirement. In Section 4.3, we present the DDM algorithm that utilizes the delay distribution information to make admission control and resource allocation decisions for real-time flows. In Section 4.4, we evaluate the performance of DDM against deterministic admission control approach and in Section 4.5, we present a summary of contributions from this chapter.

4.2 Delay to Resource Mapping

In this section, we derive the correlation function that maps the real-time flow F_i's specification $(\rho_i^{avg}, \sigma_{i,l}, D_{i,l}, P_{i,l})$ to its bandwidth reservation $\rho_{i,l}$. This is a two-step pro-

cess. First we dynamically measure the delay distribution information for traffic currently traversing the link. Next this information is leveraged to derive the delay-to-resource mapping for new flows.

We consider the context of a single link l with capacity C_l. Traffic belonging to every real-time flow F_i that traverses link l requires each packet to be serviced by the scheduler within a delay bound $D_{i,l}$ and with a delay violation probability no greater than $P_{i,l}$. For instance, if $D_{i,l} = 10ms$ and $P_{i,l} = 10^{-5}$, it means that no more than a fraction 10^{-5} of packets belonging to the real-time flow can experience a delay greater than $10ms$. Following the model outlined in Section 3.2.2, we assume that each flow's incoming traffic at link l is characterized by burst size $\sigma_{i,l}$ and average rate ρ_i^{avg}. The amount of flow F_i traffic arriving at the scheduler in any time interval of length τ is bounded by $(\sigma_{i,l} + \rho_i^{avg}\tau)$. In the next chapter, we will consider the multi-hop setting where traffic is completely smoothed before the ingress node and $\sigma_{i,l} = 0$.

Statistical delay guarantees assist in reducing the bandwidth reservation for each real-time flow by exploiting their tolerance to certain level of delay violations. The statistical multiplexing effect ensures that bursts of size $\sigma_{i.l}$ from different flows F_i tend to be temporally spread out and rarely occur at the same time. As a result, worst-case delay is rarely experienced by packets traversing a link. Assume that the request for a real-time flow F_i specifies its average rate ρ_i^{avg}, burst size $\sigma_{i.l}$, required delay bound $D_{i,l}$ and delay violation probability $P_{i,l}$ at link l. Each flow F_i traversing the link is assigned a bandwidth reservation $\rho_{i,l} \geq \rho_i^{avg}$, which satisfies both the delay requirement $(D_{i,l}, P_{i,l})$ as well as the average rate requirement ρ_i^{avg}. Note that ρ_i^{avg} is the long-term average rate of F_i whereas the bandwidth reservation $\rho_{i,l}$ is used by the scheduler to determine run-time preference for F_i's traffic over other real-time flows.

4.2.1 CDF Construction

This section describes how a meaningful delay distribution history is constructed for traffic traversing a link. Recall from Equation 3.5 that the worst-case queuing delay $D_{i,l}^{wc}$ experienced at link l by any packet belonging to flow F_i under WFQ service discipline is

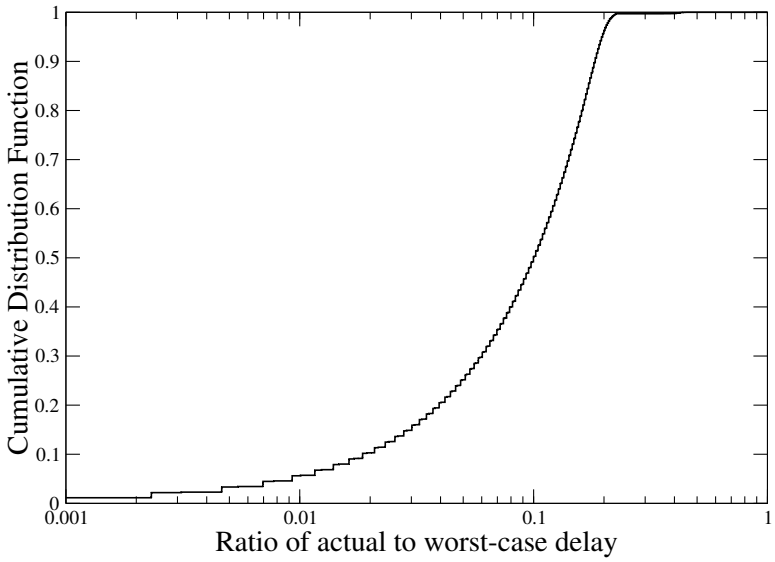

Figure 4.3: CDF records the distribution of the ratio of actual delay to worst-case delay experienced by packets. X-axis is in log scale. 39 VoIP flows traverse a 10Mbps link, with each flow having 256kbps average rate, 10ms delay bound, and 10^{-5} delay violation probability requirement.

given by the following expression.

$$D_{i,l}^{wc} = \frac{\sigma_{i,l}}{\rho_{i,l}} + \frac{L_{max}}{\rho_{i,l}} + \frac{L_{max}}{C_l} \tag{4.1}$$

where $\sigma_{i.l}$ is F_i's burst size at link l, L_{max} is the maximum packet size, $\rho_{i,l}$ is the reservation for F_i at link l, and C_l is the total capacity of link l. The first component of the delay is fluid fair queuing delay, the second component is the packetization delay and the third component is scheduler's non-preemption delay.

Assume that for each packet k, the system tracks the run-time measurement history of the ratio r_k of the actual packet delay experienced $D_{i,l}^k$ to the worst-case delay $D_{i,l}^{wc}$, i.e., $r_k = D_{i,l}^k / D_{i,l}^{wc}$ where r_k ranges between 0 and 1. We can use these measured samples of ratio r_k to construct a cumulative distribution function (CDF) $Prob(r)$. For instance, we can partition the ratio range from 0 to 1 into a million sub-ranges (or with some other appropriate granularity). Then for each sub-range, we keep updating the count of packets transmitted whose ratio r_k falls within the sub-range. The CDF can be constructed by computing the accumulated count of packets from the smallest sub-range to each sub-

range. Figure 4.3 shows an example of a CDF constructed in this manner by simulating 39 VoIP flows traversing a 10Mbps link, with each flow having 256kbps average rate, 10ms delay bound, and 10^{-5} delay violation probability requirement. (Simulation details follow in Section 4.4). We can see from the figure that most of the packets experience less than 1/4th of their worst-case delay.

4.2.2 Resource Mapping

The distribution $Prob(r)$ gives the probability that the ratio between the actual delay encountered by a packet and its worst-case delay is smaller than or equal to r. Conversely, $Prob^{-1}(p)$ gives the maximum ratio of actual delay to worst-case delay that can be guaranteed with a probability of p. Given the measured estimate of functions $Prob(r)$ and $Prob^{-1}(p)$, we can use the following heuristic to determine the delay-derived bandwidth reservation $\rho_{i,l}^{delay}$ required to satisfy real-time flow F_i's statistical delay requirement $(D_{i,l}, P_{i,l})$.

$$D_{i,l} = \left(\frac{\sigma_{i,l} + L_{max}}{\rho_{i,l}^{delay}} + \frac{L_{max}}{C_l} \right) \times Prob^{-1}(1 - P_{i,l}) \tag{4.2}$$

Equation 4.2 states that in order to obtain a delay bound of $D_{i,l}$ with a delay violation probability bound of $P_{i,l}$, we need to reserve a minimum bandwidth of $\rho_{i,l}^{delay}$ which can guarantee a worst-case delay of $D_{i,l}^{wc} = D_{i,l}/Prob^{-1}(1 - P_{i,l})$. Conversely, the delay-derived bandwidth requirement $\rho_{i,l}^{delay}$ of a flow F_i at link l is

$$\rho_{i,l}^{delay} = \mathcal{B}_l(D_{i,l}, P_{i,l}, \sigma_{i,l}) = \frac{\sigma_{i,l} + L_{max}}{\frac{D_{i,l}}{Prob^{-1}(1-P_{i,l})} - \frac{L_{max}}{C_l}} \tag{4.3}$$

The actual reservation required to satisfy flow F_i's QoS requirement $(\rho_{i,l}^{avg}, \sigma_{i,l}, D_{i,l}, P_{i,l})$ is $\rho_{i,l} = max\{\rho_{i,l}^{avg}, \rho_{i,l}^{delay}\}$. In other words, the actual bandwidth reservation for a real-time flow is dictated by the tighter of two QoS requirements – one imposed by its average bandwidth requirement $\rho_{i,l}^{avg}$ and the other imposed by its statistical delay requirement $(D_{i,l}, P_{i,l})$.

4.3 Admission Control Using Delay Distribution Measurement

In this section, we describe the DDM admission control algorithm for admitting a new flow F_N that arrives at a link l on which $N-1$ flows have already been admitted. The DDM algorithm consists of two steps. First, we need to estimate the delay distribution assuming flow F_N is admitted. The second step performs the actual admission control by applying the delay-to-resource mapping function from Section 4.2 on the estimated CDF to account for the future resource requirements of the new flow as well as existing flows.

4.3.1 Factors Influencing CDF Evolution

If the new flow F_N is admitted, the link with a finite capacity C_l has to shoulder the additional traffic load from F_N. As a result packets for all flows traversing the link will experience larger delays on the average. More specifically, the additional load from F_N could impact the CDF curve shown in Figure 4.3 by moving it to the right. In other words, for the same delay violation probability p, if $r_1 = Prob_{old}^{-1}(1-p)$ before admitting F_N and $r_2 = Prob_{new}^{-1}(1-p)$ after admitting F_N, then $r_2 \geq r_1$. Because a larger value of $Prob_{new}^{-1}(1-p)$ translates into higher bandwidth requirement, CDF_{new} is said to be more *conservative* than CD_{old} since CDF_{new} can admit fewer flows than CD_{old}. Figure 4.4 provides an example of CDF_{old} and right-shifted CDF_{new} for one simulation scenario in the Y-axis range from 0.99 to 1.0 (since this range happens to be of most interest).

If we simply use CDF_{old} to derive the bandwidth reservation for F_N, and the actual CDF_{new} turns out to be significantly more conservative than CDF_{old}, F_N may be assigned a much smaller bandwidth than what it actually needs to meet its statistical delay requirement. The key research challenge of delay distribution measurement-based admission control thus lies in how to predict the impact of the new flow F_N on the delay distribution of $(N-1)$ existing flows *without assuming a priori traffic models*.

The impact of new flow F_N on CDF_{old} depends upon several factors. In general tight QoS requirements – such as a small delay requirement $D_{N,l}$, a low tolerance to delay violation $P_{N,l}$, a large average rate ρ_N^{avg} or a big burst size σ_N – all lead to larger ratio of actual to worst-case delay for packets traversing the link. Furthermore, the increment

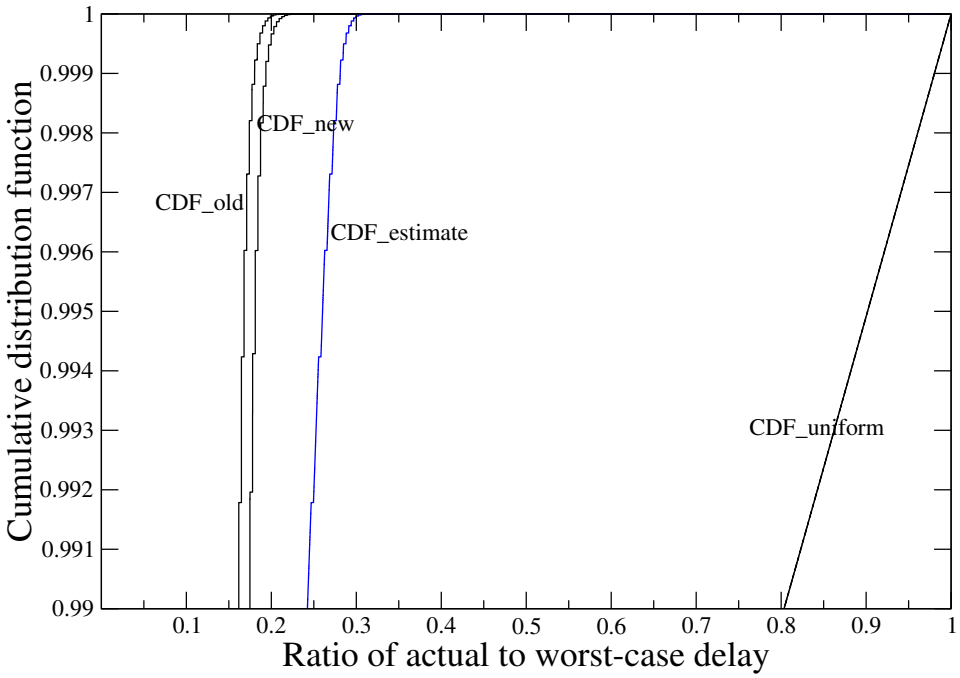

Figure 4.4: Example of different CDF curves for a simulation scenario.

from $Prob_{old}^{-1}(1-p)$ to $Prob_{new}^{-1}(1-p)$ could be different for different values of violation probability p.

Another factor that introduces non-determinism is the level of statistical multiplexing of flow F_N's traffic bursts with those of existing $N-1$ flows. For instance, Figure 4.5 shows the increment in value of $Prob^{-1}(1-10^{-5})$ for different loads imposed by a new flow on a 10Mbps link. Each existing flow requires 256kbps average rate and 10ms delay bound, at 10^{-5} violation probability. We consider the impact of the last new flow that is admitted. That is, the new flow's additional load saturates the 10Mbps link. As the load of the last admitted flow increases, the value of $Prob^{-1}(1-10^{-5})$ also increases, suggesting that the magnitude of the last flow admitted indeed affects the difference between the CDFs before and after the new flow is admitted.

4.3.2 Constructing Estimated CDF

Given the multitude of factors that influence the evolution of CDF, it is difficult, if not impossible, to exactly predict CDF_{new} using CDF_{old} and flow F_N's QoS requirements.

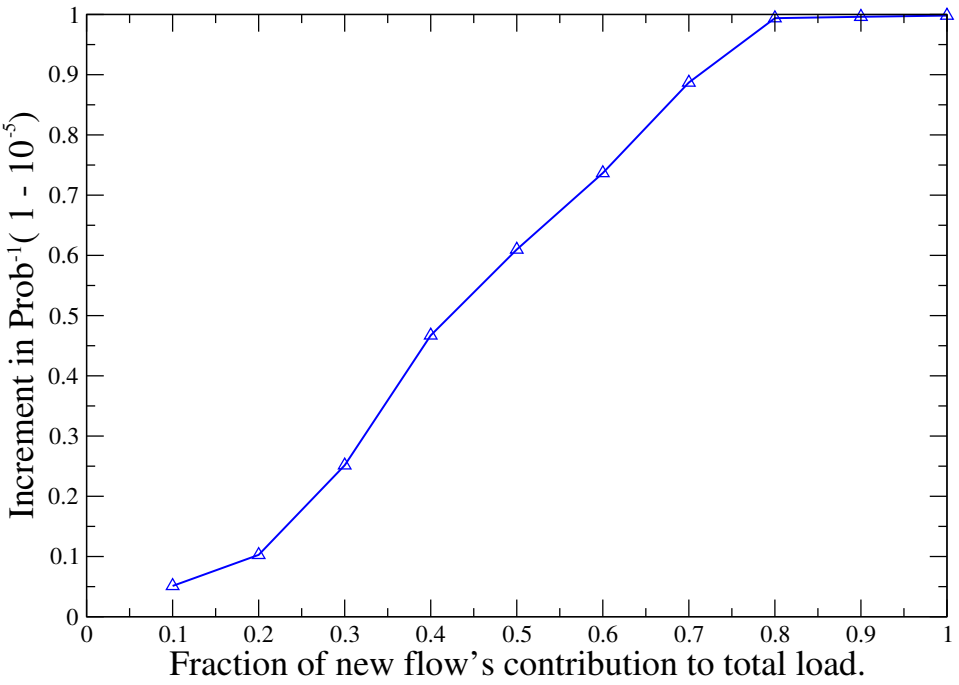

Figure 4.5: The plot shows the effect of the relative load imposed by a new flow on the ratio r in the CDF that corresponds to delay violation probability of 10^{-5} for one simulation scenario.

The DDM algorithm uses a heuristic approach to approximate CDF_{new}. Let τ be the length of a moving time window over which the delay distribution CDF_{old} of existing $N-1$ flows is measured. Let m be the number of packets generated by $N-1$ flows that traverse the link in duration τ. In a time interval τ, the flow F_N can potentially transmit a maximum of $n = \rho_N^{avg} * \tau / L_{min}$ number of packets, where L_{min} is the minimum packet size. Assume that these n additional packets experience a uniform distribution of actual to worst-case delay ratio. A uniform distribution is a conservative assumption since it implies that packet delays for the new flow F_N are expected to be uniformly distributed over the range of ratios from 0 to 1.

We first combine the uniform delay ratio distribution for F_N with a weight of $\frac{n}{n+m}$ and the delay ratio distribution CDF_{old} with a weight of $\frac{m}{n+m}$ to obtain a distribution called $CDF_{uniform}$, which represents an estimate of the cumulative distribution that would result if F_N was fully loaded and the delay ratio of the packets from F_N were distributed uniformly between 0 and 1. $CDF_{uniform}$ can be constructed using the technique described in Section 4.2, but with the difference that before computing the accumulated sum for each ratio sub-range, we add n/R to the count of ratio samples in each sub-range, where R is the number of sub-ranges between 0 and 1. In other words, n delay ratios are assumed to be uniformly distributed over all ratio sub-ranges.

Empirically $CDF_{uniform}$ is still a very conservative estimate of the distribution CDF_{new}, because both the uniform delay ratio distribution assumption and the full load assumption are too pessimistic. As a result, CDF_{new} lies somewhere between CDF_{old} and $CDF_{uniform}$ constructed above. We further approximate CDF_{new} by constructing CDF_{est}, which in turn is a weighted combination of CDF_{old} and $CDF_{uniform}$. Specifically,

$$Prob_{est}^{-1}(1-p) = \alpha Prob_{uniform}^{-1}(1-p) + (1-\alpha) Prob_{old}^{-1}(1-p) \qquad (4.4)$$

The factor α is the *impact factor* that determines how far the distribution curve CDF_{est} is from $CDF_{uniform}$ and CDF_{old}. For a new flow that imposes a relatively large load on the link with respect to existing load, CDF_{est} should be close to $CDF_{uniform}$ since the latter is more conservative in admitting flows. On the other hand, for a new flow that imposes a relatively small load with respect to existing load, CDF_{est} should be closer to CDF_{old} since in this case the new flow has a relatively smaller impact on CDF_{old}. With this consideration in mind, we define the impact factor as the fraction of new flow F_N's

load on the total expected load on the link.

$$\alpha = \frac{\rho_{N,l}}{\sum_{i=1}^{N} \rho_{i,l}} \tag{4.5}$$

Here $\rho_{i,l}$ is computed using the distribution $CDF_{uniform}$ since it is the only estimate of future delay distribution we have at the time of admitting F_N. Since we are practically interested in only the delay violation probabilities $P_{i,l}$ for existing and new flows, we only need to compute that portion of CDF_{est} which covers these delay violation probabilities of our interest; typically the violation probabilities lie in the range 10^{-2} to 10^{-6} which corresponds to a small upper portion of the Y-axis in Figure 4.3. An example of different CDF curves is illustrated in Figure 4.4 within the Y-axis range of 0.99 to 1 for one simulation scenario. We see CDF_{est} is the closest approximation to CDF_{new}, although a bit more conservative. $CDF_{uniform}$ is the most conservative of all the curves.

Note that constructing CDF_{est} involves two levels of weighted combinations – first in constructing $CDF_{uniform}$ from CDF_{old} and a uniform distribution of new flow's packets, and second in constructing CDF_{est} from CDF_{old} and $CDF_{uniform}$. The difference is that the $CDF_{uniform}$ provides a first-cut conservative estimate of CDF_{new} whereas this estimate is further refined by constructing CDF_{est}.

4.3.3 Admission Control

With the delay-probability-bandwidth correlation function in place, we now present the DDM algorithm in Figure 4.6. Assume that $N-1$ real-time flows are currently being served by the scheduler and real-time flow F_N arrives for admission. The algorithm first calculates $CDF_{uniform}$ using the measured delay distribution CDF_{old} and flow F_N's average rate requirement ρ_N^{avg}. For each of the N real-time flows (including the new one) the algorithm next computes the delay-derived bandwidth requirement $\rho_{i,l}^{delay}$ using Equations 4.3 and 4.4. Note that we do not need to compute the entire CDF_{est} explicitly. Rather, only the required values in CDF_{est} are computed on-demand using Equation 4.4. The actual bandwidth reservation $\rho_{i,l}$ is the larger of the delay-derived requirement $\rho_{i,l}^{delay}$ and average requirement $\rho_{i,l}^{avg}$. The new real-time flow F_N is admitted only if following condition is satisfied.

$$\sum_{i=1}^{N} \rho_{i,l} \leq C_l \tag{4.6}$$

Input : (a) $(D_{i,l}, P_{i,l}, \rho_i^{avg}, \sigma_i)$ for each real-time flow F_i, $1 \leq i \leq N$.
(b) The measured delay ratio distributions.

Compute CDF_{old} and $CDF_{uniform}$ from delay ratio distributions.

for $i = 1$ to N
Compute $\rho_{i,l}^{delay} = \mathcal{B}_l(D_{i,l}, P_{i,l}, \sigma_i)$ using Equations 4.3 and 4.4.
$\rho_{i,l} = max\{\rho_i^{avg}, \rho_{i,l}^{delay}\}$

/*Perform admission checks*/
if $(\sum_{i=1}^{N} \rho_{i,l} > C_l)$ then
Reject real-time flow F_N and exit.

/*Flow F_N can be admitted*/
for $i = 1$ to N
Reserve bandwidth $\rho_{i,l}$ for F_i.

Figure 4.6: The DDM algorithm to determine whether a new real-time flow F_N can be admitted such that each flow F_i, $1 \leq i \leq N$, can be guaranteed a delay bound $D_{i,l}$, delay violation probability $P_{i,l}$, and average rate ρ_i^{avg}. Note that we do not need to compute the entire CDF_{est} explicitly. Rather, only the required values in CDF_{est} are computed on-demand using Equation 4.4.

Equation 4.6 states that the sum of bandwidth reservations of all real-time flows should be smaller than C_l. The flow F_N is rejected if this condition cannot be satisfied. If the new real-time flow is accepted, the algorithm sets the bandwidth reservation for each real-time flow to $\rho_{i,l}$ as computed above.

4.3.4 Complexity

The step for computing CDF_{old} and $CDF_{uniform}$ has $O(R)$ time complexity where R is the number of sub-ranges in the delay ratio interval from 0 to 1. The subsequent steps in the algorithm have $O(N)$ time complexity where N is the number of real-time flows

being considered. Thus the complexity of the DDM algorithm is $O(N + R)$. In practice, the first step of computing CDF_{old} and $CDF_{uniform}$ is the more dominant of the two components due to the large number of sub-ranges R. The algorithm itself is invoked infrequently when new flow requests arrive for admission at the link. The run-time overhead of maintaining CDFs is minimal since we only need few arithmetic operations to record the ratio for each packet transmitted by the link.

4.4 Performance of DDM

In this section, we study the performance of the DDM algorithm in comparison to deterministic admission control.

4.4.1 Simulation Setup

Using the *ns-2* simulator, we configured a single link at 10 Mbps and packets arriving at the link were served by a WFQ scheduler. Each real-time flow traffic consisted of aggregated traffic traces of recorded VoIP conversations used in [61], in which spurt-gap distributions were obtained using G.729 voice activity detector. Each VoIP stream had an average data rate of around 13 kbps, peak data rate of 34 kbps, and packet size of $L_{max} = 128$ bytes. We temporally interleaved the 20 VoIP streams to generate aggregate traffic trace for each real-time flow with an aggregate data rate of $\rho_i^{avg} = 256$kbps. Each aggregated flow trace was 8073 seconds long. Every real-time flow sent traffic for the entire lifetime of the simulation with the aggregate traffic trace being repeated over its lifetime. Traffic from each admitted real-time flow passed a token bucket with bucket depth of 1280 bytes (10 packets) and token rate of 256 kbps.

Each new real-time flow required a guarantee on delay bound and a delay violation probability. The admission control algorithm decided whether to admit or reject the real-time flow and how much bandwidth to reserve according to algorithm in Figure 4.6. A connection generator initiated each real-time flow with a periodic inter-arrival time of $10,000$ seconds. The reason we selected periodic instead of exponential inter-arrival times (as in other works) is that our real-time flows are long lived and are expected to arrive fairly infrequently, so that the measured CDF can stabilize before being used to admit

another real-time flow. Hence the request arrival pattern does not significantly impact the admission control decisions. Each simulation run lasted for $1000,000$ seconds.

The CDF was measured over a time interval of $10,000$ seconds between flow arrivals. For simulations, we recorded the ratio of actual to worst-case delay of every packet traversing the link within the current sliding window (although in a realistic scenario an intelligent sampling mechanism would be more desirable). The observed ratios are accumulated into a histogram. The actual CDF is computed from the histogram only when an admission decision needs to be made or the bandwidth reservations of existing real-time flows need to be re-calculated.

4.4.2 Delay Bound Variation

First we compare the performance of DDM algorithm against deterministic admission control as delay bound requirement varies. With DDM algorithm, the delay violation probability for each real-time flow is 10^{-5} whereas deterministic admission control considers a zero delay violation probability. Figure 4.7 plots the number of real-time flows admitted as the delay-bound requirement is varied from 3 to 50ms. Figure 4.8 plots the factor improvement in number of flows admitted with DDM algorithm compared to the deterministic approach. The maximum number of real-time flows that can be admitted on the 10Mbps link is limited to 39 real-time flows by the average rate requirement of 256kbps for each real-time flow. Figure 4.8 shows that for small delay bound requirements, DDM algorithm admits around 3.0 times more number of real-time flows than deterministic admission control when delay violation probability as small as 10^{-5} is allowed. As the delay bound requirement becomes less stringent, DDM algorithm still admits more real-time flows and achieves better link utilization than deterministic algorithm, but with smaller improvements. Beyond 45ms delay requirement, both algorithms are limited to admitting 39 real-time flows due to the average rate requirement of 256kbps for each real-time flow. The gain for DDM algorithm comes from the fact that large majority of packets experience just 1% to 3% of the worst-case delay dictated by their reserved bandwidth. This statistic gets reflected in the CDF which in turn helps to reduce the resource requirement for each real-time flow.

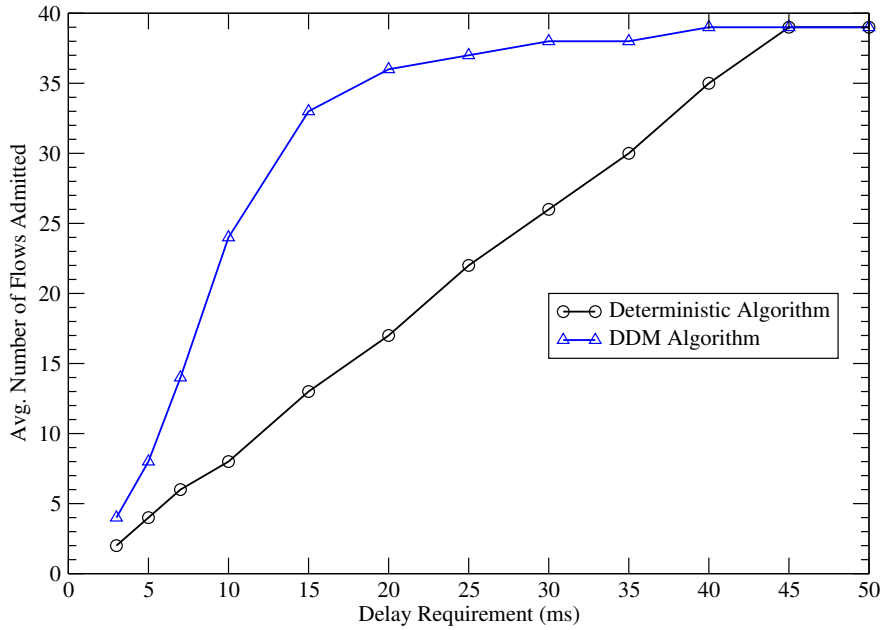

Figure 4.7: Number of admitted real-time flows vs. delay bound. Delay violation probability = 10^{-5}. Burst Size=10 packets. Link capacity = 10Mbps.

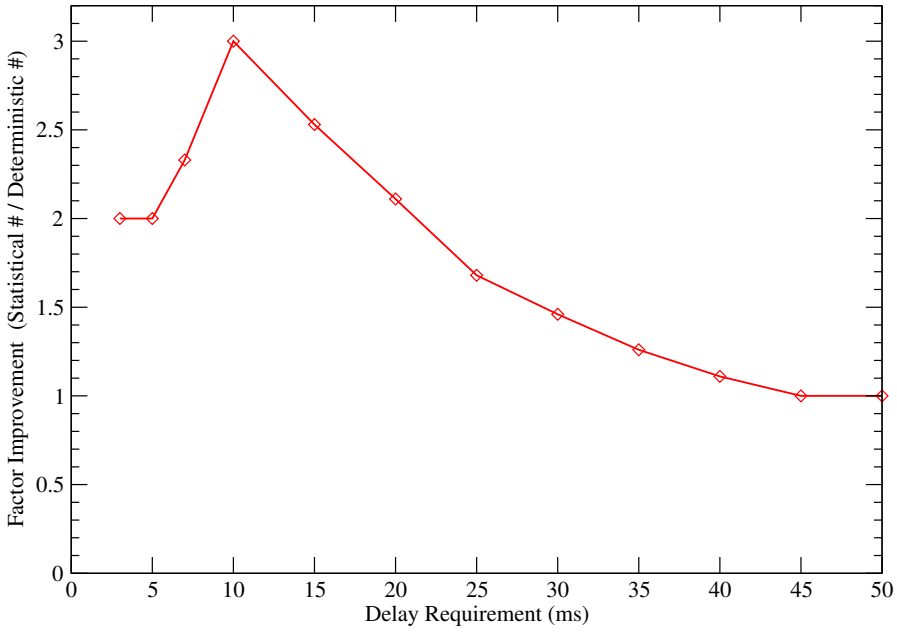

Figure 4.8: Factor improvement in number of admitted flows vs. delay bound. Delay violation probability = 10^{-5}. Burst Size=10 packets. Link capacity = 10Mbps.

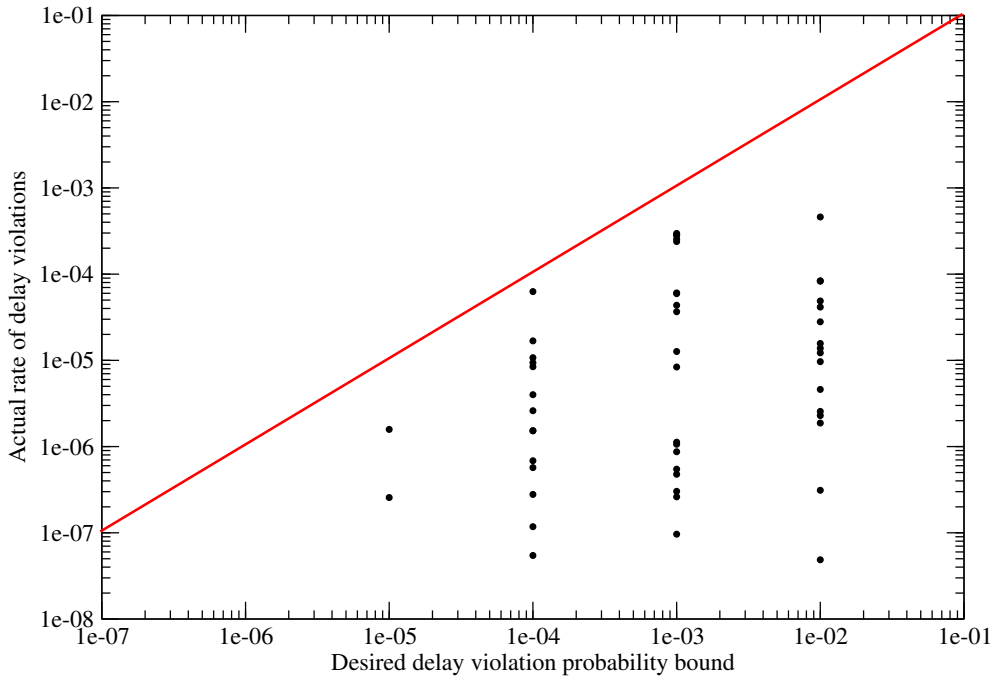

Figure 4.9: The DDM algorithm satisfies distinct per-flow delay violation guarantees. Each data point corresponds to one real-time flow experiencing delay violation. Delay bound=20ms. Burst Size=10 packets. Link capacity = 10Mbps. The plot includes data points from 5 simulation runs with different random seeds.

4.4.3 Heterogeneous Delay Violation Probabilities

Next we show that the DDM algorithm indeed provides distinct guarantees on heterogeneous delay violation probabilities for a mix of different traffic types. In this experiment we consider a traffic mix in which all real-time flows request the same delay bound of 20ms, same average rate of 256kbps, and same burst size of 10 packets, but require a different guarantees on delay violation probability; the requirement being uniformly distributed among the four values 10^{-2}, 10^{-3}, 10^{-4} and 10^{-5}. Figure 4.9 plots the actual fraction of packets exceeding their delay bound against the desired violation probability for each real-time flow that experiences any excess delay. Each data point represents the rate of delay violation experienced by one real-time flow. The line through the graph marks the limit above which the actual rate of delay violations would exceed the desired delay violation probability. The fact that all data points are below the line indicates that

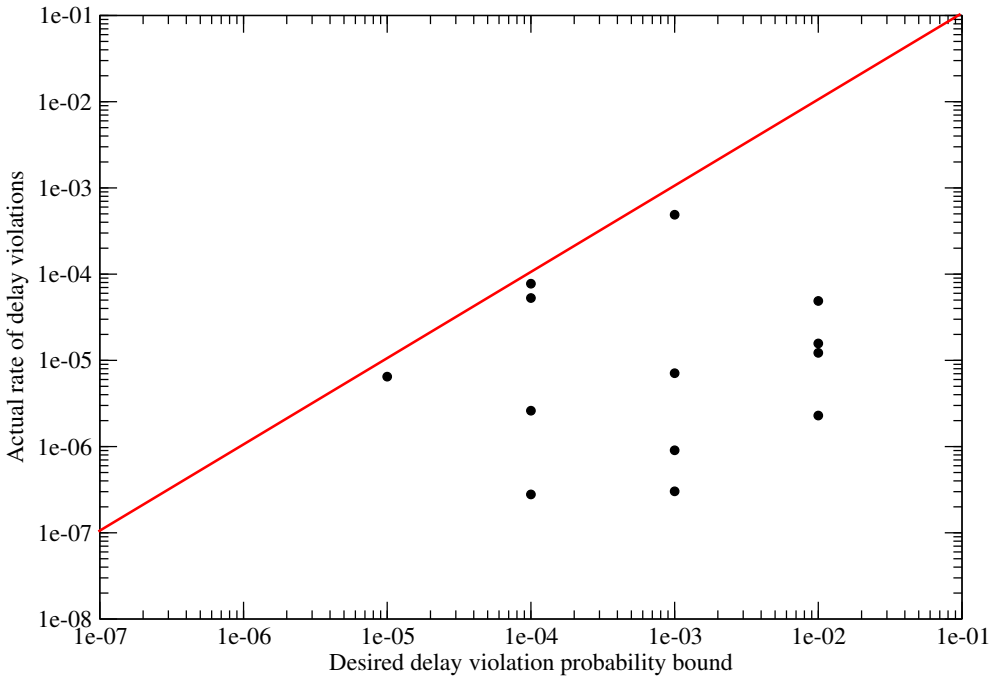

Figure 4.10: The DDM algorithm satisfies distinct per-flow delay violation guarantees even when constituent flows have different heterogeneous delay bound, data rate requirements, and burstiness requirements. Each data point corresponds to one real-time flow experiencing delay violation. The plot includes data points from 5 simulation runs with different random seeds.

the actual delay violation rate is smaller than the maximum permissible for each real-time flow. Furthermore the figure shows that real-time flows that have higher tolerance to delay violations are more likely to experience a higher rate of violation than real-time flows with lower tolerance. The DDM algorithm is able to distinguish among real-time flows in terms of delay violation rates because it assigns service bandwidth $\rho_{i,l}$ to real-time flows in the inverse proportion of their tolerance to delay violations. This translates to higher dynamic preference for packets belonging to real-time flows with low delay tolerance and vice-versa. Figure 4.10 shows that DDM also achieves the desired delay violation targets when the constituent flows have heterogeneous QoS requirements. Delay bound is randomly chosen from 10ms, 20ms, or 30ms; data rate from 256kbps, 512kbps, 1Mbps, or 2Mbps; burst size from 10, 20, 30, or 40 packets.

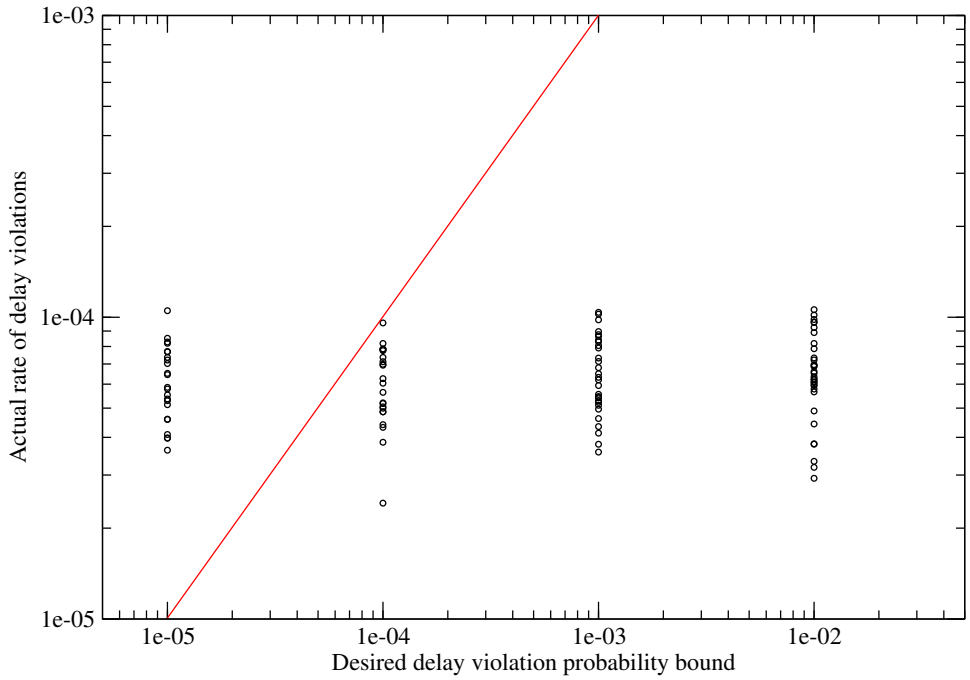

Figure 4.11: The pure over-subscription based algorithm cannot satisfy distinct per-flow delay violation guarantees. Each data point corresponds to one real-time flow experiencing delay violation. Delay bound=20ms. Burst size=10 packets. Link capacity=10Mbps and is over-subscribed by a factor of 2.0. The plot includes data points from 5 simulation runs with different random seeds.

In the next experiment, we show that pure over-subscription of link capacity cannot provide distinct guarantees on heterogeneous delay violation probabilities. We use the same parameters as the previous experiment, except that instead of using the DDM algorithm, we use deterministic admission control and over-subscribe the link capacity by a factor of 2.0 so as to admit the same number of real-time flows as the DDM algorithm (i.e. 35 flows) with no over-subscription. Figure 4.11 shows that irrespective of desired delay violation bounds, all real-time flows experience similar rates of actual delay violations. In fact, real-time flows with low tolerance (10^{-5}) to delay violations can experience an order of magnitude higher delay violations than their actual tolerance. This is because pure over-subscription based algorithm does not correlate delay violation bound requirements for a real-time flow with its bandwidth reservation. Hence we need, more than just band-

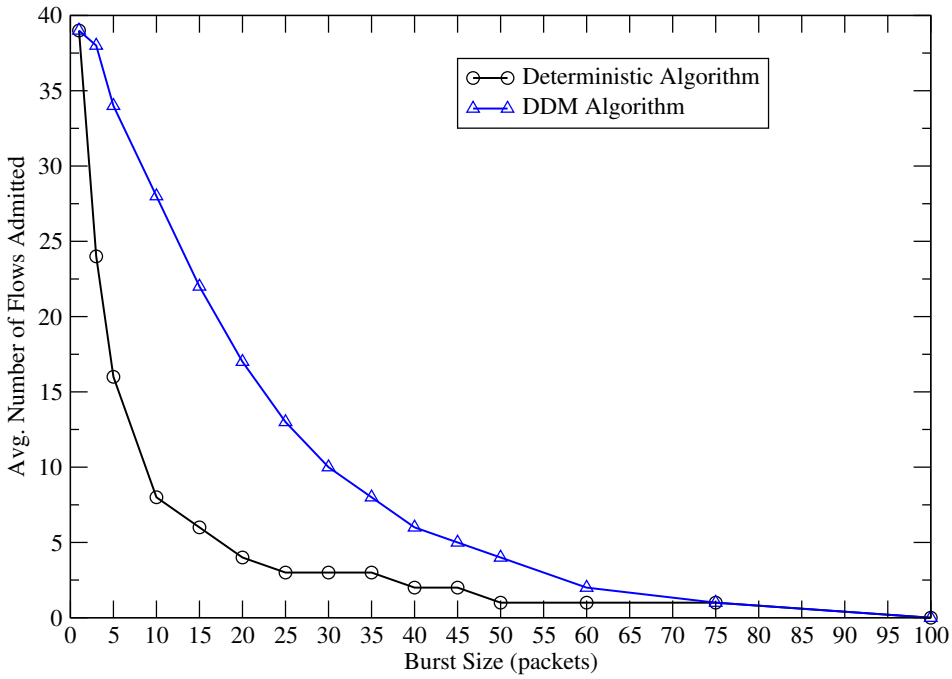

Figure 4.12: Number of admitted real-time flows vs. burst size. Delay bound=10ms. Violation Probability=10^{-5}. Link Capacity = 10Mbps.

width over-subscription – specifically a delay-probability-bandwidth correlation such as the one in Equation 4.2 – to guarantee distinct per-flow statistical guarantees.

4.4.4 Burst Size Variation

Figure 4.12 compares the DDM algorithm against deterministic admission control as the burst size σ_i for each flow is increased from 1 to 100 packets. Up to burst sizes of 40 packets, DDM algorithm admits significantly larger number of flows than deterministic algorithm since it can successfully exploit the statistical multiplexing among bursts from different flows. For larger burst sizes, the delay-derived bandwidth requirement is too high to be adequately compensated by statistical multiplexing.

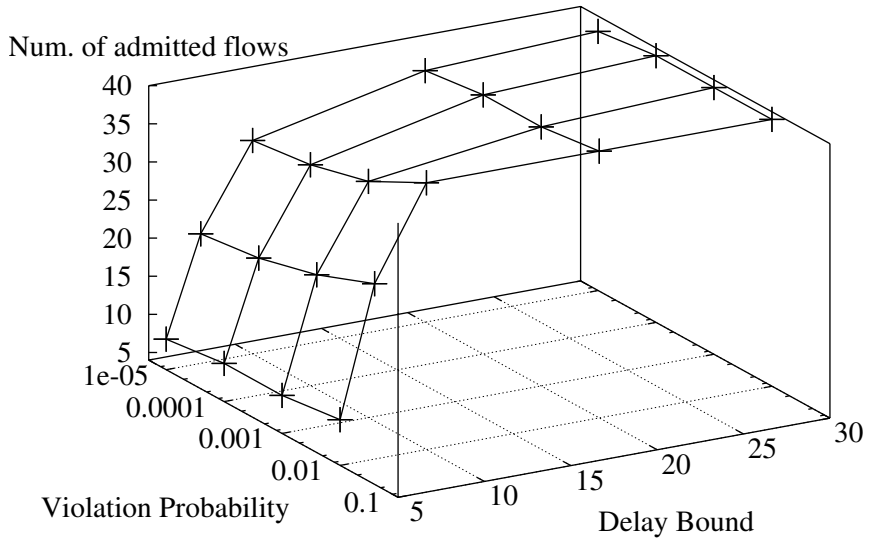

Figure 4.13: Admission region for various combinations of delay and delay violation probability. More flows are admitted as delay requirements become less stringent; the maximum number of flows admitted being 39. Link Capacity = 10Mbps. Burst size=10 packets. Average rate=256kbps.

4.4.5 Admission Region

Figure 4.13 shows the admission region for various combinations of delay and delay violation probability. As the delay bound and delay violation probability requirements become less stringent, the number of admitted flows increases. Note that even with a low violation probability of 10^{-5} at 10ms delay, the DDM algorithm can admit up to 24 flows which is 3 times more than deterministic case of 8 flows.

4.4.6 Effect of CDF Measurement Window

Another factor influencing the performance of the DDM algorithm is CDF measurement window. Figure 4.14 shows that a large measurement window leads to a more conservative admission process i.e., a large measurement window admits fewer real-time flow

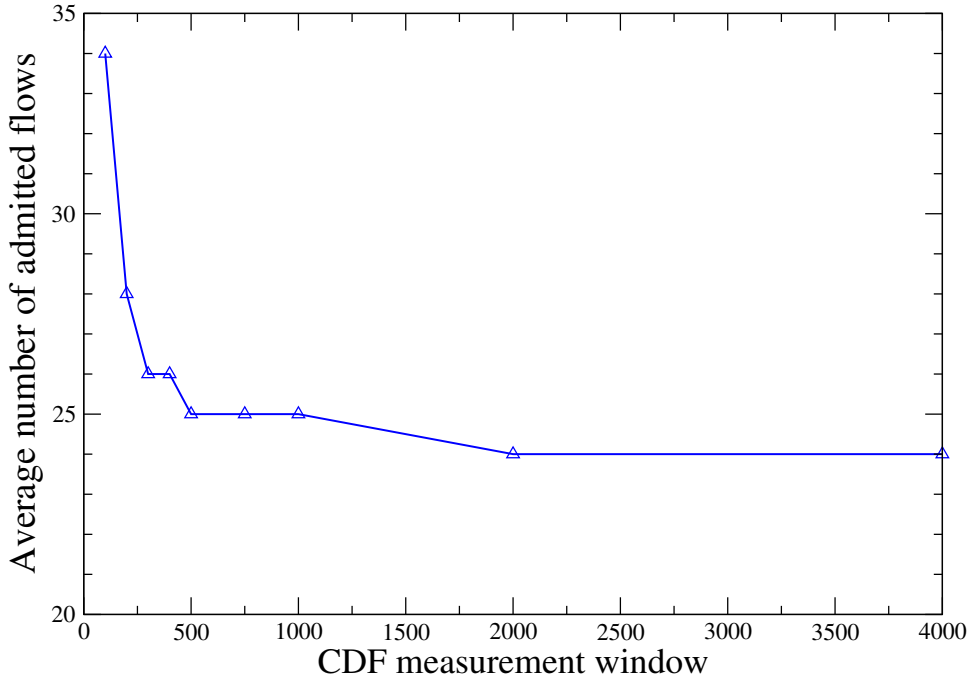

Figure 4.14: Number of admitted real-time flows with different CDF measurement windows. Admission control becomes more conservative with larger measurement windows. Delay bound=$10ms$. Violation probability = 10^{-5}. Burst size=10 packets.

requests than a small window over the same interval of time. The reason for this behavior can be traced back to Figure 4.1. Typically bursts from different flows tend to be temporally spread out and it is relatively rare for several flows to burst simultaneously. However, such events do occur and small window sizes are more likely to miss out such rare simultaneous traffic bursts whereas large window sizes are more likely to capture these. Consequently, larger measurement windows produce more representative CDF curves than small measurement windows. Admission decisions based on small measurement windows could thus be over-optimistic leading to more number of real-time flows being admitted quickly. With large window sizes, the DDM algorithm is slower in reacting to changes in traffic patterns and thus admits fewer flows as traffic load increases. While a very small window size can result in over-optimistic admissions, an extremely large window size would also lead to inaccurate admission decisions since it might include history that could be too old for consideration. Thus one needs to strike a right

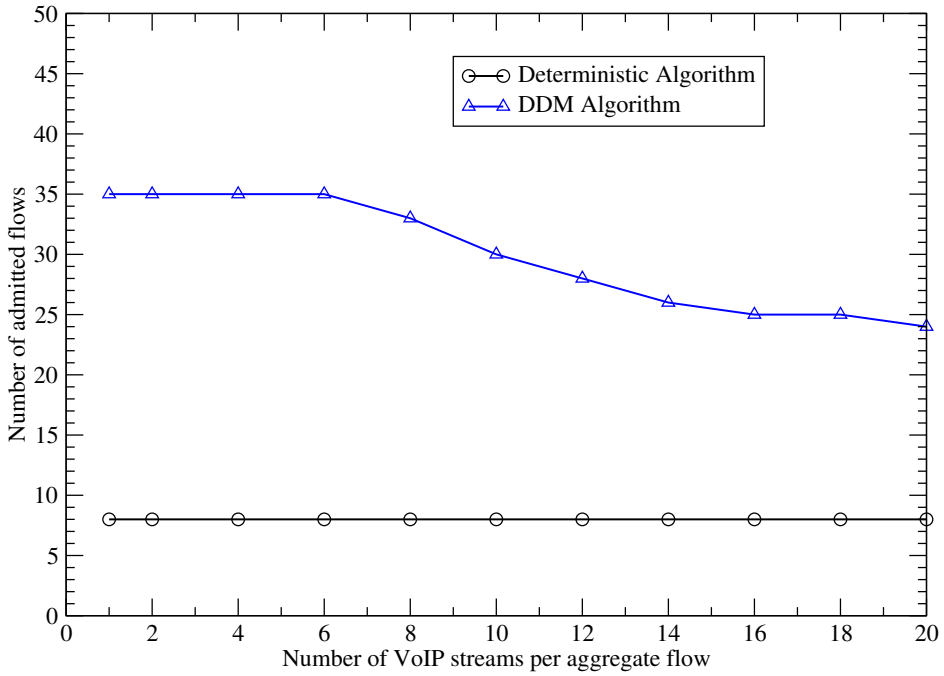

Figure 4.15: Number of admitted aggregate flows vs. number of VoIP streams peraggregate flow. Delay bound=$10ms$. Violation probability = 10^{-5}. Burst size=10 packets.

balance in selecting a measurement window size that yields optimal performance. A possible choice for the CDF measurement window could be the duration between successive flow arrivals since the traffic during this period can be expected to be largely stable and indicative of true load imposed by currently active flows.

4.4.7 Gain From Per-flow Under-utilization

Finally, we vary the number of streams in each flow to determine the extent of gain we obtain by under-utilizing the aggregate flow's reserved capacity. At full capacity, each aggregate flow can carry 20 VoIP streams. Figure 4.15 shows that the number of admitted flows decreases from 35 to 24 as the level of aggregation in each flow increases from 2 to 20 VoIP streams. Thus, DDM algorithm can successfully exploit additional statistical multiplexing due to smaller level of aggregation in each flow in order to admit more flows. In this case, the maximum gain is limited by the average rate requirement of 256 kbps for each flow and link capacity of 10Mbps. The multiplexing gains could be higher if the

DDM algorithm can exploit the statistical multiplexing effects along delay dimension as well as the bandwidth dimension.

4.5 Summary of Contributions

In this chapter, we presented the Delay Distribution Measurement (DDM) based admission control algorithm, that provides distinct per-flow statistical delay guarantees (i.e., both delay bound and delay violation probability bound) to each real-time flow sharing a link. We developed a novel quantitative framework to exploit the well known fact that the actual delay experienced by most packets of a real-time flow is usually far smaller than its worst-case delay bound requirement. Flows that can tolerate more delay bound violations can reserve less resources than those that tolerate less, even though they have the same delay bound requirement. DDM effectively exploits statistical multiplexing effects by dynamically measuring the distribution of the ratios between actual packet delay and worst-case delay bound. With this delay ratio distribution, DDM is able to gauge the actual load of a system, estimate the actual resource requirement of new flows with distinct statistical delay requirements, and eventually determine whether to admit them. DDM is the first approach that applies the concept of measurement-based admission control to provide per-flow statistical delay guarantees. A comprehensive simulation study using Voice over IP traces shows that, when compared to deterministic admission control algorithms, the DDM algorithm can potentially increase the number of admitted flows (and link utilization) by up to a factor of 3.0 when the delay violation probability is as small as 10^{-5}.

Chapter 5

Partitioning of End-to-end Delay Constraints

5.1 Problem Description

Traffic engineering techniques in Multi-Protocol Label Switched (MPLS) networks select explicit routes for Label Switched Paths (LSP) between a given source and destination. Each LSP acts as a traffic trunk carrying an aggregate traffic flow that requires QoS guarantees such as bandwidth and delay bounds. The key approach underlying traffic engineering algorithms, such as [66, 107], is to select network paths so as to balance the loads on the network links and routers. Without load balancing, it is possible that resources at one link might be exhausted much earlier than others, thus rendering the entire network paths unusable.

For real-time network flows that require end-to-end delay guarantees, there is an additional optimization dimension for balancing loads on the network links, that is the partitioning of end-to-end delay requirements along the links of a selected network path. Specifically we are interested in the variable components of end-to-end delay such as queuing delay at intermediate links and smoothing delay before the ingress, rather than the fixed delay components such as propagation and switching delays. We use the term "delay" to refer to variable components of end-to-end delay.

Consider the path shown in Figure 5.1 in which each link is serviced by a packetized

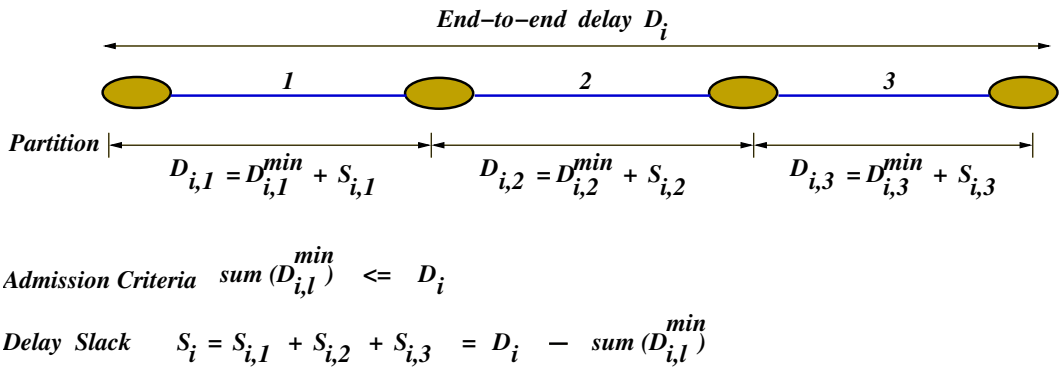

Figure 5.1: Example of partitioning end-to-end delay budget D_i over a three-hop path. The slack S_i indicates the amount of flexibility available in partitioning the delay budget D_i.

rate-based scheduler such as WFQ [85]. Given a request for setting up a real-time flow F_i along this path with a deterministic end-to-end delay requirement D_i, we need to assign delay budgets $D_{i,l}$ to F_i at each individual link l of the path. Intuitively, low delay typically requires high bandwidth reservation. In other words, since flow F_i's packet delay bound at each link is inversely proportional to its bandwidth reservation, the amount of delay budget allocated to F_i at a link determines the amount of bandwidth reservation that F_i requires at that link. How do we partition the end-to-end delay requirement D_i into per-link delay budgets such that (a) the end-to-end delay requirement D_i is satisfied and (b) the amount of traffic admitted along the multi-hop path can be maximized in the long-term? This is the *Delay Budget Partitioning* problem for delay constrained network flows.

As we saw in Chapter 4, a class of real-time traffic, such as VoIP, can tolerate their packets experiencing delays in excess of D_i within a small delay violation probability P_i. Statistical delay requirements of the form (D_i, P_i) can assist in reducing bandwidth reservation for real-time flows by exploiting their tolerance levels to delay violations. When we consider partition of the end-to-end delay violation probability requirement P_i as well as the delay requirement D_i, the delay budget allocation problem is generalized to statistical delay requirements for network flows.

Another important context in which the delay budget partitioning problem arises is that of co-ordinated multi-resource allocation in real-time operating systems [45]. Here

each system component, such as CPU, disk, or network interface, constitutes a resource instance, analogous to the earlier example of multiple network links along a network flow path. Real-time applications need end-to-end delay guarantees across the usage of multiple system resources. The goal of resource allocation is to assign delay budgets to tasks using different system resources such that the number of real-time applications admitted in the long term can be maximized.

One of the important considerations in maximizing network resource utilization efficiency is to maintain load balance across different links. The prior approaches to solving the delay budget allocation problem were based on heuristics such as equal partitioning [79], proportional partitioning [37] or cost function based partitioning [70, 34, 65, 92] that did not directly consider the load balancing criteria. Additionally, the prior proposals handle partition of only a single QoS requirement and directly partition the entire end-to-end QoS over the links of the path which leads to the link disabling problem described in Section 2.3.3.

In this chapter we propose a new algorithm, called Load-based Slack Sharing (LSS), to solve the delay budget allocation problem, and describe its application to unicast and multicast flows with deterministic and statistical delay guarantees. LSS explicitly takes into account the load balancing criteria in order to maximize the network resource utilization efficiency. Additionally, LSS can handle multiple simultaneous flow QoS requirements such as end-to-end delay bounds and delay violation probability bounds. A central notion in LSS is that of *slack* in end-to-end QoS that helps avoid the link disabling problem encountered with prior approaches.

We model our work on the lines of Guaranteed Service architecture [101] where the network links are serviced by packetized rate-based schedulers [85, 120, 41, 118] and there exists an inverse relationship between the amount of service rate of a flow at a link and the corresponding delay bound that the flow's packets experience. A rate-based model with GPS schedulers was also adopted in [37]. Note that we address the problem of partitioning *additive* end-to-end delay requirement and the *multiplicative* delay violation probability requirement. Specifically, the problem we address is *not* the same as partitioning *bottleneck* QoS parameters. Indeed, for packetized rate-based schedulers such as WFQ, the end-to-end delay bound for a flow contains both additive (per-link) and

bottleneck delay components. In Chapter 3, we showed how these two components can be separated and how the QoS constraint partitioning problem is relevant in the context of rate-based schedulers.

The rest of the chapter is organized as follows. Section 5.2 introduces the notion of *slack sharing* which is central to our work. Section 2.3 places our work in the context of related research in delay budget allocation. In Section 5.3, 5.4 and 5.5 we describe a series of delay budget allocation algorithms for unicast and multicast network paths having deterministic and statistical end-to-end delay requirements. In Section 5.6 we analyze the performance of the proposed algorithms and Section 5.7 summarizes the main research results.

5.2 Slack and Slack Sharing

The extent of flexibility available in balancing the loads across links of a multi-hop path is quantified by the *slack* in end-to-end delay budget. For each link of the multi-hop path in Figure 5.1, we can compute the minimum local delay budget $D_{i,l}^{min}$ guaranteed to a new flow F_i at the link l provided that all residual (unreserved) bandwidth at l is assigned for servicing packets from the new flow. The difference between the end-to-end delay requirement D_i and the sum of minimum delay requirements $D_{i,l}^{min}$, i.e. $\Delta D_i = D_i - \sum_{l=1}^{3} D_{i,l}^{min}$, represents an excess *slack* that can be shared among the links to reduce their respective bandwidth loads.

The manner in which slack is shared among links of a flow path determines the extent of load balance (or imbalance) across the links. When the links on the selected network path carry different loads, one way to partition the slack is to share it based on the current loads on the network links. For example, assume that the three links in Figure 5.1 carry a load of 40%, 80% and 20% respectively. Given a total slack in delay budget ΔD_i, one could partition the slack proportionally as $\frac{2}{7}\Delta D_i$, $\frac{4}{7}\Delta D_i$, and $\frac{1}{7}\Delta D_i$, respectively, rather than assigning each $\frac{1}{3}\Delta D_i$. The reason that the former assignment is better from the viewpoint of load balancing is that a more loaded link should be assigned a larger delay budget in order to impose a lower bandwidth demand on it. The latter scheme, on the other hand, would lead to second link getting bottlenecked much earlier than the

other two, preventing any new flows from being admitted along the path. In fact, as we will show later, we can do even better than proportional allocation described above if we explicitly balance the loads across different links.

5.3 Unicast Flow with Deterministic Delay Guarantee

We now propose a series of algorithms for delay budget allocation that we call Load-Based Slack Sharing (LSS) algorithms. The first algorithm, presented in this section, addresses deterministic delay guarantees for unicast flows. The second algorithm addresses statistical delay guarantees for unicast flows and the third algorithm extends LSS for multicast flows; these are presented in subsequent sections. The deterministic and statistical algorithms for unicast flows are named D-LSS and S-LSS respectively and those for multicast flows are named D-MLSS and S-MLSS.

Let us start with the case where a flow F_N is requested on a unicast path and requires an end-to-end deterministic delay guarantee of D_N i.e, none of the packets carried by F_N can exceed the delay bound of D_N. Assume that $N-1$ flows have already been admitted on the unicast path. The total capacity and current bandwidth load on the l^{th} link are represented by C_l and L_l respectively. In the following discussion, we overload the notation F_N to represent the set of links along the path of the unicast flow.

The goal of delay budget allocation is to partition flow F_N's end-to-end delay budget D_N into a set of delay budgets $D_{N,l}$ on the links of F_N and the smoothing delay $D_{N,s}$ at the smoother, such that the following partition constraint is satisfied

$$D_{N,s} + \sum_{l \in F_N} D_{N,l} \leq D_N \tag{5.1}$$

and *the number of flows that can be admitted over the unicast path in the long-term is maximized.*

We saw in Chapter 3 that, in general for rate-based schedulers like WFQ, there exists a function of the form $\mathcal{D}_l(\rho_{i,l})$ that correlates a flow F_i's bandwidth reservation $\rho_{i,l}$ on a link l to its packet delay bound $D_{i,l}$. The specific form of relation $\mathcal{D}_l(\rho_{i,l})$ is dependent on the packet scheduling discipline employed at the links of the network. For example, if a link is managed by a WFQ scheduler, then the relation is given by Equation 3.5.

5.3.1 Admission Control (D-ADM)

Before computing $D_{N,l}$, one needs to determine whether the flow F_N can be admitted into the system in the first place. Towards this end, first we calculate the minimum delay budget that can be guaranteed to F_N at each link if all the residual bandwidth on the link is assigned to F_N. Thus the minimum delay budget at link l is given by $\mathcal{D}_l(C_l - L_l)$, where C_l is the total capacity and L_l is the currently reserved capacity. From Equation 3.6, the corresponding minimum smoothing delay is $\sigma_N / \min_{l \in F_N}\{C_l - L_l\}$. The flow F_N can be admitted if the sum of minimum smoothing delay and per-link minimum delay budgets is smaller than D_N; otherwise F_N is rejected. More formally,

$$\frac{\sigma_N}{\min_{l \in F_N}\{C_l - L_l\}} + \sum_{l \in F_N} \mathcal{D}_l(C_l - L_l) \leq D_N \tag{5.2}$$

5.3.2 Load-based Slack Sharing (D-LSS)

Once the flow F_N is admitted, next step is to determine its actual delay assignment at each link along the unicast path. We define the *slack* in delay as

$$\Delta D_N = D_N - D_{N,s} - \sum_{l \in F_N} D_{N,l} \tag{5.3}$$

If the flow F_N can be admitted, it means that after assigning minimum delay budgets to the flow at each link, the slack ΔD_N is positive. The purpose of slack sharing algorithm (D-LSS) is to reduce the bandwidth requirement of a new flow F_N at each of its links in such a manner that the number of flows admitted in future can be maximized. The D-LSS algorithm for unicast flows with deterministic delay guarantee is given in Figure 5.2. Let the bandwidth assigned to flow F_N on the l^{th} link after delay budget assignment be of the form $\beta_l \times \min\{b, \frac{1}{\beta_l}\} \times (C_l - L_l)$. The algorithm essentially tunes the value of b in subsequent iterations until the resulting slack falls below a predefined *DThreshold*. Smaller the value of *DThreshold*, the closer LSS can get to the optimization objective.

Since $\beta_l \times \min\{b, \frac{1}{\beta_l}\} \times (C_l - L_l)$ represents the bandwidth assigned to F_N on the l^{th} link, β_l can be set differently depending on the optimization objective. If the optimization objective is to ensure that bandwidth assigned on different links is proportional to the remaining link capacity, then β_l should be set to 1. This indeed turns out to be the best

Algorithm D-LSS(F_N,D_N)

/* F_N = Unicast flow being admitted. */

/* D_N = End-to-end delay bound. */

$\delta = \max_{l \in F_N} 1/\beta_l$;

$b = \delta$;

while(1) {

for each link $l \in F_N$ {

$\rho_{N,l} = \max\{\ \beta_l \times \min\{b, \frac{1}{\beta_l}\} \times (C_l - L_l)\ ,\ \rho_N^{avg}\}$

$D_{N,l} = \mathcal{D}_l(\rho_{N,l})$; /* Delay at link l */

}

$D_{N,s} = \sigma_N / \min_{l \in F_N}\{\rho_{N,l}\}$; /* Smoother delay */

$slack = D_N - \sum_{l \in F_N} D_{N,l} - D_{N,s}$;

if ($slack \geq 0$ and $slack \leq DThreshold$)

return $\hat{D_N}$;

$\delta = \delta/2$;

if ($slack > 0$)

$b = b - \delta$;

else

$b = b + \delta$;

}

Figure 5.2: D-LSS: Load-based Slack Sharing algorithm for a unicast flow with deterministic delay guarantee. The algorithm returns the delay vector $\hat{D_N} =< D_{N,1}, D_{N,2}, \cdots, D_{N,m} >$.

strategy when we are allocating resources on a unicast path in isolation, since it leads to perfect load balancing among the links. On the other hand, this strategy may not be the best approach for a general network topology where links happen to be shared among intersecting paths. Thus the value of β_l needs to be carefully chosen depending upon the *network-wide* optimization objective rather than one local to the unicast path. For the objective of maximizing the amount of admitted traffic in the network, we need to ensure that resources at links, that are expected to shoulder more traffic load than other links, should last longer. One effective strategy in a general network topology is to select $\beta_l = 1/u_l$, where u_l is the long-term *expected utilization* for link l obtained by means of traffic profiling. The expected utilization is computed as $u_l = \phi_l/C_l$, where ϕ_l is the profiled expected load on link l and C_l is the link capacity. The process of computing the expected load ϕ_l will be detailed in Chapter 6. Profile based traffic engineering has been shown to be effective in earlier research [107] as also will be shown during evaluations in this chapter and Chapter 6.

5.4 Unicast Flow With Statistical Delay Guarantee

Now we consider the case where a new flow F_N requires statistical delay guarantees (D_N, P_N) over unicast path, i.e, the end-to-end delay of its packets needs to be smaller than D_N with a probability greater than $(1 - P_N)$. Since the delay bound does not need to be strictly enforced at all times, the network resource demand can be presumably smaller for a fixed D_N. The S-LSS algorithm needs to distribute D_N and P_N to constituent links of F_N. In other words, it needs to assign values $D_{N,l}$ and $P_{N,l}$, such that the following partition constraints are satisfied

$$D_{N,s} + \sum_{l \in F_N} D_{N,l} \leq D_N \tag{5.4}$$

$$\prod_{l \in F_N} (1 - P_{N,l}) \geq (1 - P_N) \tag{5.5}$$

and *the number of flow requests that can be admitted into the system in the long-term is maximized.* Here we assume there exist correlation functions $\mathcal{D}_l(\rho_{i,l}, P_{i,l})$ and $\mathcal{P}_l(\rho_{i,l}, D_{i,l})$

that can correlate the bandwidth reservation $\rho_{i,l}$ to a statistical delay bound $(D_{i,l}, P_{i,l})$.

$$D_{i,l} = \mathcal{D}_l(\rho_{i,l}, P_{i,l}) \tag{5.6}$$

$$P_{i,l} = \mathcal{P}_l(\rho_{i,l}, D_{i,l}) \tag{5.7}$$

In Chapter 4, we gave a concrete example of such a correlation function in the context of statistical delay guarantees where links are serviced by a WFQ scheduler.

Note that the above condition on partitioning the end-to-end delay violation probability is more conservative than necessary. In particular, we assume that a packet can satisfy its end-to-end delay bound only if it satisfies its per-hop delay bounds. For instance a packet could exceed its delay bound at one link, be serviced early at another link along the path and in the process still meet its end-to-end delay bound. However, modeling such a general scenario is difficult and depends on the level of congestion at different links and their inter-dependence. Hence we make the most conservative assumption that a packet which misses its local delay bound at a link is dropped immediately and not allowed to reach its destination. This helps us partition the end-to-end delay violation probability into per-hop delay violation probabilities as mentioned above.

5.4.1 Admission Control (S-ADM)

The admission control algorithm for the case of statistical delay guarantees is more complicated than the deterministic case because it needs to check whether there exists at least one set of $\{< D_{N,l}, P_{N,l} >\}$ that satisfy the partitioning constraints 5.4 and 5.5. The detailed admission control algorithm is shown in Figure 5.3. It starts with an initial assignment of minimum delay value $\mathcal{D}_l(C_l - L_l, 0)$ to $D_{N,l}$ assuming that all the remaining capacity of the l^{th} link is dedicated to F_N and $P_{N,l} = 0$. Since the initial assignment might violate end-to-end delay constraint, i.e. $D_{N,s} + \sum_{l \in F_N} D_{N,l} > D_N$, the algorithm attempts to determine if there exists a looser $P_{N,l}$ such that delay partition constraint can be satisfied. Thus the algorithm iteratively increases the value of some $P_{N,k}$ by a certain amount δ and recomputes its associated $D_{N,k}$, until either $D_{N,s} + \sum_{l \in F_N} D_{N,l}$ becomes smaller than D_N, or $\prod_{l \in F_N} (1 - P_{N,l})$ becomes smaller than $(1 - P_N)$. In the first case, F_N is admitted since a constraint satisfying partition exists; in the latter case even assigning all the available resources along the F_N's path is insufficient to support the QoS

Algorithm S-ADM(F_N,D_N,P_N)
/* F_N = Unicast flow being admitted. */
/* D_N = End-to-end delay bound. */
/* P_N = End-to-end delay violation probability. */

for each link $l \in F_N$ do $\{$
$P_{N,l} = 0$;
$D_{N,l} = \mathcal{D}_l(C_l - L_l, 0)$;
$\}$
while $\left(D_{N,s} + \sum_{l \in F_N} D_{N,l} > D_N \right)$ and $\left(\prod_{l \in F_N} (1 - P_{N,1}) \geq (1 - P_N) \right)$ do
$\{$
k = index of link such that reduction in
delay $D_{N,k} - \mathcal{D}_k(C_k - L_k, P_{N,k} + \delta)$ is
maximum among all links;

$P_{N,k} = P_{N,k} + \delta$;
$D_{N,k} = \mathcal{D}_k(C_k - L_k, P_{N,k})$;
$\}$
if ($D_{N,s} + \sum_{l \in F_N} D_{N,l} > D_N$)
Reject flow request F_N;
else
Accept flow request F_N;

Figure 5.3: S-ADM: The admission control algorithm for unicast flow F_N with statistical delay requirements (D_N, P_N).

requested and F_N is rejected. In each iteration, that link k is chosen whose delay budget reduces the most when its violation probability bound is increased by a fixed amount, δ, i.e, the one that maximizes $D_{N,k} - \mathcal{D}_k(C_k - L_k, P_{N,k} + \delta)$.

5.4.2 Load-based Slack Sharing (S-LSS)

In the context of statistical delay guarantees, the goal of slack sharing algorithm is to apportion both the slack in delay ΔD_N and the slack in assigned probability ΔP_N over the network links traversed by the flow F_N. ΔD_N is calculated using Equation 5.3 and ΔP_N is calculated as follows.

$$\Delta P_N = \frac{\prod_{l \in F_N} (1 - P_{N,l})}{(1 - P_N)} \tag{5.8}$$

Let the delay vector $< D_{N,1}, D_{N,2}, \cdots, D_{N,m} >$ be represented by $\hat{D_N}$ and the probability vector $< P_{N,1}, P_{N,2}, \cdots, P_{N,m} >$ be represented by $\hat{P_N}$, where m is the number of links in flow F_N's path. The S-LSS algorithm for unicast flows with statistical delay guarantees is given in Figure 5.4. The algorithm starts with a feasible assignment of minimum delay vector $\hat{D_N}$ and probability vector $\hat{P_N}$ obtained during admission control. In every iteration, the algorithm first relaxes the delay vector $\hat{D_N}$ assuming fixed probability vector $\hat{P_N}$, and then relaxes the probability vector while fixing the delay vector. This process repeats itself till the distance between the values of $\hat{D_N}$ or between $\hat{P_N}$ from two consecutive iterations falls below a predefined threshold. Since the $\hat{P_N}$ values affect the $\hat{D_N}$ value relaxation step and vice-versa, multiple rounds of alternating relaxation steps are typically required to arrive at the final allocation. The unicast_relax_delay() procedure is similar to the deterministic D-LSS algorithm in Figure 5.2 except that the resource correlation function $\mathcal{D}_l(\rho_{N,l})$ is replaced by $\mathcal{D}_l(\rho_{N,l}, P_{N,l})$. Figure 5.5 gives the unicast_relax_prob() procedure which is similar to unicast_relax_delay(), except that the correlation function is $\mathcal{P}_l(\rho_{N,l}, D_{N,l})$ and the slack in probability is defined as in Equation 5.8. In evaluations described in Section 5.6, we empirically observe that this two-step iterative relaxation algorithm typically converges to a solution within 2 to 5 iterations.

Algorithm S-LSS(F_N,D_N,P_N)
/* F_N = Unicast flow being admitted. */
/* D_N = End-to-end delay bound. */
/* P_N = End-to-end delay violation probability. */

Initialize $\hat{D_N}$ and $\hat{P_N}$ with the final $D_{N,l}$ and $P_{N,l}$ values
computed from the admission control algorithm S-ADM;

do {
$\hat{D'_N} = \hat{D_N}$; $\hat{P'_N} = \hat{P_N}$;
$\hat{D_N} =$ unicast_relax_delay(F_N, $\hat{P_N}$, D_N);
$\hat{P_N} =$ unicast_relax_prob(F_N, $\hat{D_N}$, P_N);
} while(($\mid \hat{D_N} - \hat{D'_N} \mid > thresh_D$) or ($\mid \hat{P_N} - \hat{P'_N} \mid > thresh_P$));

Figure 5.4: S-LSS: Load-based Slack Sharing algorithm for unicast flow F_N with statistical delay guarantee (D_N, P_N).

Algorithm unicast_relax_prob(F_N,$\hat{D_N}$,P_N)

/* F_N = Unicast flow being admitted. */

/* $\hat{D_N}$ = Current delay partition vector. */

/* P_N = End-to-end delay violation probability. */

$\delta = \max_{l \in F_N} 1/\beta_l$;

$b = \delta$;

while(1) {

for $l \in F_N$ do {

$\rho_{N,l} = \max\{\ \beta_l \times \min\{b, \frac{1}{\beta_l}\} \times (C_l - L_l)\ ,\ \rho_N^{avg}\}$

$P_{N,l} = \mathcal{P}_l(\rho_{N,l}, D_{N,l})$; /* Violation probability at link l */

}

$slack = \frac{\prod_{l \in F_N}(1-P_{N,l})}{(1-P_N)}$;

if($slack \geq 1$ and $slack \leq PThreshold$)

return $\hat{P_N}$;

$\delta = \delta/2$;

if ($slack > 0$)

$b = b - \delta$;

else

$b = b + \delta$;

}

Figure 5.5: unicast_relax_prob() algorithm to relax probability assignment vector $\hat{P_N}$, given a delay assignment vector $\hat{D_N}$.

5.5 Delay Partitioning for Multicast Flows

It is relatively straightforward to generalize the algorithms in Sections 5.3 and 5.4 to multicast flows. In this section, we present the deterministic and statistical versions of the LSS algorithm for multicast flows, namely D-MLSS and S-MLSS.

A real-time multicast flow M_N consists of a sender at the root and K receivers at the leaves. We assume that the flow requirement is the same value (D_N, P_N) for each of the K leaves, although the number of hops in each of the K paths may be different. A multicast flow M_N with K receivers can be logically thought of as K unicast flows $F_{N,1} \cdots F_{N,K}$, one flow to each leaf. Although logically treated as separate, the K unicast flows share a single common reservation at each common link along their flow paths. In the following discussion, we overload the notation M_N to represent the set of links in the multicast tree and $F_{N,i}$ to represent the set of links along the i^{th} unicast flow path of the multicast tree.

5.5.1 Admission Control (D-MADM and S-MADM)

A K-leaf multicast flow M_N with an end-to-end delay requirement can be admitted if and only if all of the constituent K unicast flows $F_{N,1} \cdots F_{N,K}$ can be admitted. The admission control algorithm for multicast flow thus consists of applying variants of the unicast admission control algorithms in Sections 5.3 and 5.4 to each of the K unicast paths. The variations account for the fact that the K unicast paths are not completely independent.

Figures 5.6 and 5.7 give the admission control algorithms for multicast flows with deterministic and statistical delay guarantees respectively. There are two specific variations from the unicast version that deserve mention. First, the order of verifying admissibility among the K unicast paths makes a difference in the case of statistical delay guarantees. Intuitively, more "loaded" paths should be processed earlier since they provide least partitioning flexibility. Hence, we use the sum of the minimum delay budgets $\mathcal{D}_l(C_l - L_l, 0)$ to determine the processing order of the K unicast paths: paths with higher $\sum_{l \in F_{N,i}} \mathcal{D}_l(C_l - L_l, 0)$ are processed earlier. Secondly, since the K unicast paths are part of the same tree, several of these paths share common links. Before verifying the ad-

Algorithm D-MADM(M_N**,** D_N**)**
/* M_N = Multicast flow being admitted. */
/* D_N = End-to-end delay bound. */

for each link $l \in M_N$ do
$D_{N,l} = \mathcal{D}_l(C_l - L_l)$;

For $i = 1$ to K do $\{$

if ($\sum_{l \in F_{N,i}} D_{N,l} > D_N$) then $\{$
Reject multicast session M_N;
exit;
$\}$
$\}$
Accept multicast session M_N;

Figure 5.6: D-MADM: Admission control algorithm for multicast flow M_N with deterministic delay guarantee D_N. The algorithm determines the minimum delay that can be assigned to a multicast flow at each link of a multicast tree.

Algorithm S-MADM(M_N**,** D_N**,** P_N**)**
/* M_N = Multicast flow being admitted. */
/* D_N = End-to-end delay bound. */
/* P_N = End-to-end delay violation probability bound. */

for each link $l \in M_N$ do $\{$
$P_{N,l} = 0$;
$D_{N,l} = \mathcal{D}_l(C_l - L_l, 0)$;
$\}$

For each unicast flow $F_{N,i}$ in decreasing order of $\sum_{l \in F_{N,i}} \mathcal{D}_l(C_l - L_l, 0)$ do $\{$

while $\left(\sum_{l \in F_{N,i}} D_{N,l} > D_N \right)$ and $\left(\prod_{l \in F_{N,i}} (1 - P_{N,l}) \geq (1 - P_N) \right)$ do $\{$
j = index of the link in $F_{N,i}$ such that reduction in
delay $D_{N,j} - \mathcal{D}_j(C_j - L_j, P_{N,j} + \delta)$ is maximum
among all links in the path $F_{N,i}$;

$P_{N,j} = P_{N,j} + \delta$;

$D_{N,j} = \mathcal{D}_j(C_j - L_j, P_{N,j})$;
$\}$

if ($\sum_{l \in F_{N,i}} D_{N,l} > D_N$) then $\{$
Reject multicast session M_N;
exit;
$\}$
$\}$
Accept multicast session M_N;

Figure 5.7: S-MADM: Admission control algorithm for multicast flow F_N with statistical delay guarantee (D_N, P_N). The algorithm determines the minimum delay and violation probability that can be assigned to a multicast session at each link of a multicast tree.

missibility of $F_{N,i}$ unicast path, the $D_{N,l}$ values on some of its links may have already been computed while processing unicast paths $F_{N,1}$ to $F_{N,i-1}$. Hence, we carry over the previous $D_{N,l}$ assignments for links that are shared with already processed paths.

5.5.2 Load-based Slack Sharing (D-MLSS and S-MLSS)

To compute the final delay allocation for the links of a K-leaf multicast path, we apply a variants of the D-LSS and S-LSS algorithms in Figures 5.2 and 5.4.

Let's start with the deterministic version D-MLSS given in Figure 5.8. D-MLSS essentially applies the unicast delay relaxation (D-LSS) algorithm to the K unicast paths one after another. There are two salient aspects to the algorithm. First, the delay relaxation is applied to the unicast paths $i = 1$ to K in the increasing order of their current slack in delay budget $(D_N - D_{N,s} - \sum_{l \in F_{N,i}} D_{N,l})$, where $D_{N,s}$ is the smoothing delay defined in Equation 3.6. This processing order ensures that slack allocation along one path of the tree does not violate end-to-end delay along other paths that may have smaller slack. Secondly, when processing unicast path $F_{N,i}$, the delay relaxation only applies to those links which are not shared with paths $F_{N,1}$ to $F_{N,i-1}$, i.e. those links in $F_{N,i}$ whose $D_{N,l}$ values have not yet been determined.

The S-MLSS algorithm for multicast flows with statistical delay guarantees, given in Figure 5.9, is similar in structure to the unicast version S-LSS. Specifically, the algorithm starts with the minimum delay and probability values obtained in the admission control phase and successively relaxes the delay value assuming fixed probability and the probability values assuming fixed delays. The difference with unicast version is that the delay and probability relaxation steps apply to the entire multicast tree rather than one constituent unicast flow at a time. The multicast_relax_delay() and multicast_relax_prob() procedures are given in Figures 5.10 and 5.11 and are similar in structure to the deterministic D-MLSS algorithm.

5.6 Performance Evaluation

In this section, we evaluate the performance of the LSS and M-LSS algorithms for unicast and multicast flows against three earlier approaches.

Algorithm D-MLSS(M_N, D_N)
/* M_N = Multicast flow being admitted. */
/* D_N = End-to-end delay guarantee. */

for $l \in M_N$ do
$D_{N,l} = \mathcal{D}_l(C_l - L_l)$;

$R = \emptyset$; /*set of links on which slack has been relaxed*/

for each unicast flow $F_{N,i}$ in the order of decreasing $(\sum_{l \in F_{N,i}} D_{N,l})$ do {

/*extract the next unicast sub-path F to relax*/
$F = F_{N,i} - R$;

/*calculate end-to-end delay requirement D_F over unicast sub-path F*/
$D_F = D_N - \sum_{l \in \{F_{N,i} \cap R\}} D_{N,l}$;

/*relax the delay vector $\hat{D_F}$ over unicast sub-path F*/
$\hat{D_F} =$ D-LSS(F,D_F)

$R = R \bigcup F$;
}

Figure 5.8: D-MLSS: Load-based Slack Sharing algorithm for a multicast flow M_N with deterministic delay guarantee D_N.

Algorithm S-MLSS(M_N**,**D_N**,**P_N**)**
/* M_N = Unicast flow being admitted. */
/* D_N = End-to-end delay bound. */
/* P_N = End-to-end delay violation probability. */

Initialize $\hat{D_N}$ and $\hat{P_N}$ with the final $D_{N,l}$ and $P_{N,l}$ values
computed from the admission control algorithm S-MADM;

do {
$\hat{D_N'} = \hat{D_N}$; $\hat{P_N'} = \hat{P_N}$;
$\hat{D_N} =$ multicast_relax_delay(F_N, $\hat{P_N}$, D_N);
$\hat{P_N} =$ multicast_relax_prob(F_N, $\hat{D_N}$, P_N);
} while(($\mid \hat{D_N} - \hat{D_N'} \mid > thresh_D$) or ($\mid \hat{P_N} - \hat{P_N'} \mid > thresh_P$));

Figure 5.9: S-MLSS: Load-based Slack Sharing algorithm for multicast flow M_N with statistical delay guarantee (D_N, P_N).

Algorithm multicast_relax_delay(M_N**,** $\hat{P_N}$**,** D_N**)**
/* M_N = Multicast flow being admitted. */
/* $\hat{P_N}$ = Current probability partition vector. */
/* D_N = End-to-end delay. */

$R = \emptyset$; /*set of links on which slack has been relaxed*/

for each unicast flow $F_{N,i}$ in the order of decreasing $(\sum_{l \in F_{N,i}} D_{N,l})$ do $\{$

/*extract the next unicast sub-path F to relax*/
$F = F_{N,i} - R$;

/*calculate end-to-end delay requirement D_F over unicast sub-path F*/
$D_F = D_N - \sum_{l \in \{F_{N,i} \cap R\}} D_{N,l}$;

/*relax the delay vector $\hat{D_F}$ over unicast sub-path F*/
$\hat{D_F}$ = unicast_relax_delay(F,$\hat{P_N}$,D_F)

$R = R \bigcup F$;
$\}$

Figure 5.10: The multicast_relax_delay() algorithm for a multicast flow M_N with statistical delay guarantee (D_N, P_N).

Algorithm multicast_relax_prob(M_N**,** $\hat{D_N}$**,** P_N**)**
/* M_N = Multicast flow being admitted. */
/* $\hat{D_N}$ = Current delay partition vector. */
/* P_N = End-to-end delay violation probability. */

$R = \emptyset$; /*set of links on which slack has been relaxed*/

for each unicast flow $F_{N,i}$ in the order of increasing $(\prod_{l \in F_{N,i}} (1 - P_{N,l}))$ do $\{$

/*extract the next unicast sub-path F to relax*/
$F = F_{N,i} - R$;

/*calculate end-to-end delay violation probability requirement P_F over unicast sub-path F*/
$P_F = 1 - \frac{(1-P_N)}{\prod_{l \in \{F_{N,i} \cap R\}} (1-P_{N,l})}$;

/*relax the probability vector $\hat{P_F}$ over unicast sub-path F*/
$\hat{P_F}$ = unicast_relax_prob(F, $\hat{D_N}$, P_F)

$R = R \bigcup F$;
$\}$

Figure 5.11: The multicast_relax_prob() algorithm for a multicast flow M_N with statistical delay guarantee (D_N, P_N).

5.6.1 Earlier Approaches for Comparison

The first scheme, named Equal Slack Sharing (ESS), is based on the Equal Allocation (EA) scheme proposed in [79]. EA equally partitions the end-to-end delay among the constituent links in the path of a unicast flow. As discussed in Section 2.3, ESS is an improvement over EA since it partitions the slack in end-to-end QoS (delay and/or delay violation probability) equally among the constituent links. MESS is a multicast version of ESS in which the variations proposed in Section 5.5 are applied. The second scheme, named Proportional Slack Sharing (PSS), is based on the Proportional Allocation (PA) scheme proposed in [37]. PA directly partitions the end-to-end QoS in proportion to loads on constituent links of unicast/multicast path. As with ESS scheme, PSS is a variant of PA that partitions the slack in end-to-end QoS in proportion to the loads on constituent links and MPSS is a multicast variant of PSS. The third scheme is the Binary-OPQ (or OPQ for short) proposed in [72] for unicast paths, which requires that the global optimization objective is expressed as the sum of per-link cost functions. We use squared sum of per-links loads, i.e. $\sum_{l=1}^{m}(L_l/C_l)^2$, as the cost to be minimized for global optimization since it captures overall loads as well as variation in loads across different links. Thus we defined the per-link cost in OPQ as $(L_l/C_l)^2$. Again, we partition the slack in end-to-end QoS rather than the end-to-end QoS directly. MOPQ is the multicast version of OPQ proposed in [72].

Since OPQ and MOPQ operate with only a single end-to-end QoS requirement, we compare them only against the deterministic D-LSS and D-MLSS versions. On the other hand, it is straightforward to extend ESS, PSS and their multicast versions to handle the two simultaneous QoS requirements of end-to-end delay and delay violation probability. Hence we compare these schemes for both deterministic and statistical cases. As a note on terminology, D-ESS and S-ESS refer to deterministic and statistical versions of ESS scheme for unicast flows, D-MESS and S-MESS refer to the same for multicast flows and so on for PSS.

The admission control algorithm that we use along with ESS, PSS, OPQ, and their multicast versions is exactly the same as what we propose in this chapter for LSS and MLSS. In other words, before slack sharing is performed using any of the schemes, the decision on whether to admit a new flow is made by comparing the accumulated end-to-

end minimum QoS against the required end-to-end QoS. Thus the differences shown in performance result solely from different techniques for slack sharing.

5.6.2 Evaluation Setup

We evaluate the performance of different slack sharing algorithms using unicast paths, multicast trees, and general network topologies. The first topology for unicast paths (Unicast-1) has 45 Mbps capacity at the last link and 90 Mbps capacity at all other links. The second topology for unicast paths (Unicast-2) has a random mix of link capacities between 45 Mbps to 200 Mbps. Similarly, the Multicast-1 topology consists of a tree in which destinations are connected to 45 Mbps links whereas interior links have 90 Mbps capacity. The Multicast-2 topology consists of a random mix of link capacities between 45 Mbps to 200 Mbps. Unicast algorithms are also compared using 5×5 and 6×6 Grid topologies and Sprint's North American IP backbone topology in Figure 5.12. The Sprint topology consists of the OC-192 bidirectional links; however, for manageable scale of simulations, we restrict the link capacities in both Sprint and Grid topologies to a random mix between 45 Mbps and 200 Mbps. In each simulation instance, sources and destinations are selected uniformly among all nodes that are separated by a fixed shortest path distance. In other words, all source-destination pairs are seperated by a path-length of h, where h varies from 6 to 10 depending upon the specific experiment. Flows are established along the shortest paths connecting the selected sources and destinations.

As described in Section 5.3.2, evaluations of LSS algorithms on unicast and multicast paths use $\beta_l = 1$ in the slack sharing phase since it leads to perfect load balancing. On the other hand, for the Grid and Sprint network topologies, we select $\beta_l = 1/u_l$, where u_l is the *expected utilization* for link l obtained by means of traffic profiling. The traffic profile is generated by executing a dry-run in which flow requests are admitted on the network based on their long-term average rates till the network is saturated to capacity. During the actual execution of LSS, flow requests are generated in the same order as the dry-run, but the admission decisions are now based on both delay constraints as well as average rate constraints.

Algorithms for deterministic end-to-end delay guarantees (D-* schemes) are evaluated using C++ implementations whereas those for statistical delay guarantees (S-* schemes)

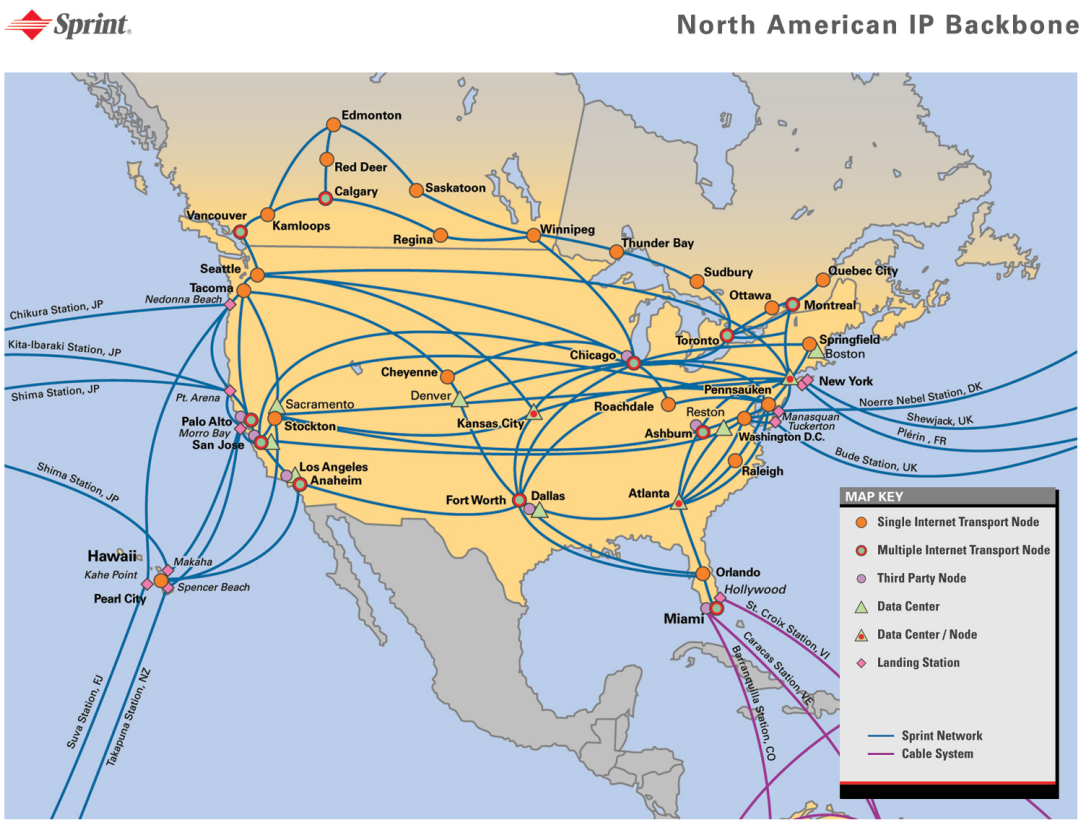

Figure 5.12: Sprint's North American IP backbone topology.

are evaluated using dynamic trace driven simulations with the ns-2 network simulator. Each real-time flow traffic in trace driven simulations consists of aggregated traffic traces of recorded VoIP conversations used in [61], in which spurt-gap distributions are obtained using G.729 voice activity detector. Each VoIP stream has an average data rate of around 13 kbps, peak data rate of 34 kbps, and packet size of $L_{max} = 128$ bytes. We temporally interleave different VoIP streams to generate 5 different aggregate traffic traces, each with a data rate of $\rho_i^{avg} = 100$kbps. Each aggregated flow trace is 8073 seconds long. Every real-time flow sends traffic for the entire lifetime of the simulation with the aggregate traffic trace being repeated over its lifetime. The results shown in statistical experiments are average values over 10 test runs with different random number seed used to select VoIP traces and initiate new flows. Flow requests arrive with a random inter-arrival time between 1000 to 5000 seconds. We perform evaluations mainly for the 'static' case in which flow reservations that are provisioned once stay in the network forever. CDF used in admission control decisions at each link is measured over the time-interval between flow arrivals. Each flow has a reservation state stored at every link that it traverses. The WFQ [120] service discipline is employed for packet scheduling at each link in order to guarantee the bandwidth shares of flows sharing the same link. In the rest of the section, we present performance results for unicast topology with deterministic and statistical delay guarantees, and for multicast trees and general network topologies with deterministic delay guarantees. For multicast trees and general network topologies, the memory requirements in the case of statistical trace driven simulations do not scale in our current system.

5.6.3 Effectiveness of LSS Algorithm

We first take a snapshot view of the performance of LSS algorithm in comparison to ESS, PSS and OPQ algorithms and later examine the impact of different parameters in detail. Table 5.1 shows the number of flows admitted over 7-hop paths with deterministic and statistical delay requirements. Figure 5.13 plots the number of flows admitted with deterministic delay guarantees under Unicast-2, Multicast-2, 5x5 Grid and Sprint topologies for different mixes of link bandwidths. The table and figures demonstrate that in all scenarios, LSS consistently admits more number of flows than all other algorithms. This is

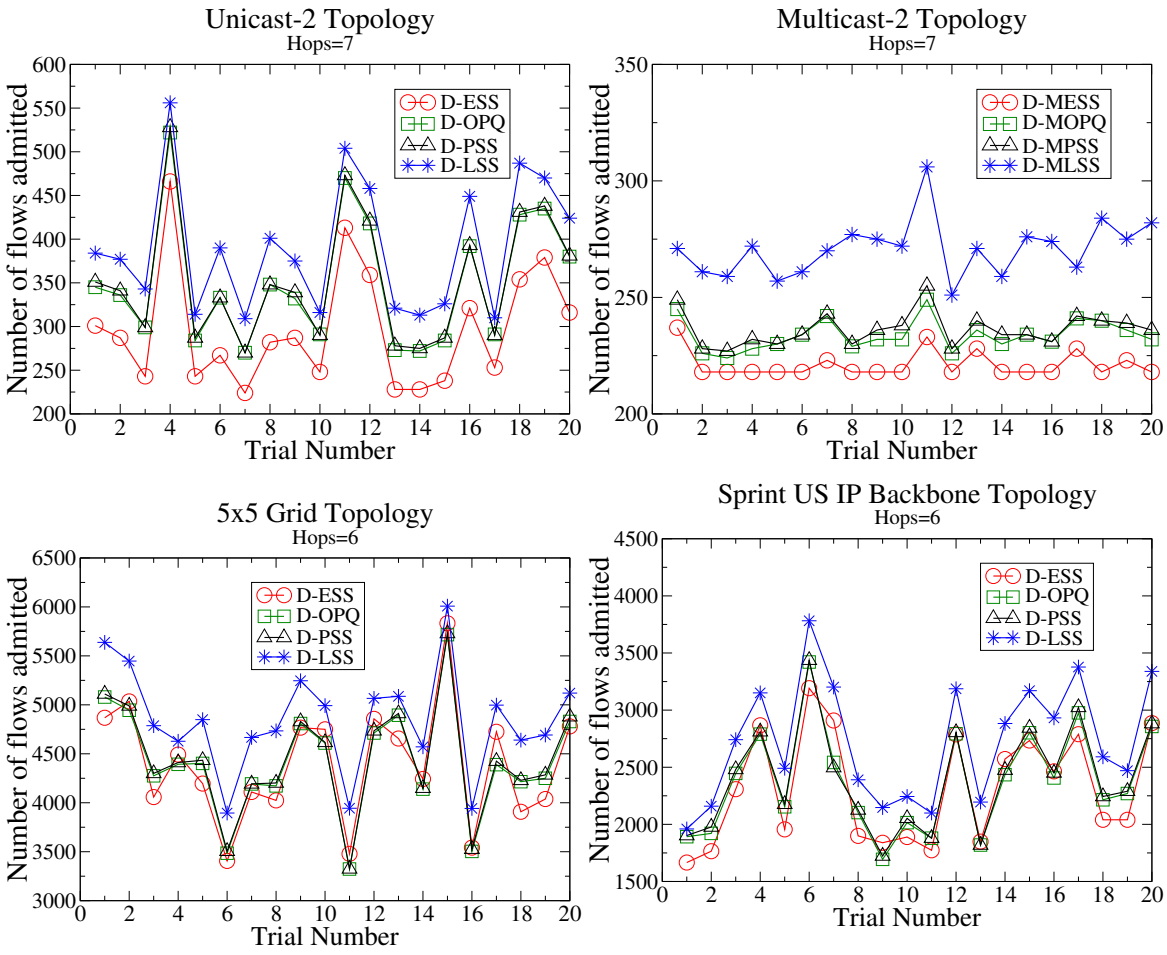

Figure 5.13: Number of flows admitted with different topologies for deterministic delay guarantees. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$.

Scenario	Topology	**ESS**	**PSS**	**OPQ**	**LSS**
Unicast/Deterministic	Unicast-1	219	263	271	295
$D_i = 60$ms	Unicast-2	225	275	284	310
Unicast/Statistical	Unicast-1	81	121	N/A	319
$D_i = 45$ms $P_i = 10^{-5}$	Unicast-2	81	117	N/A	292
Multicast/Deterministic	Multicast-1	221	268	265	292
$D_i = 60$ms	Multicast-2	219	249	244	267
Grid/Deterministic	5x5	2906	3237	3179	3495
$D_i = 60$ms	6x6	6280	6362	6386	6983
Sprint/Deterministic	US IP	1320	1585	1561	1614
$D_i = 60$ms	Backbone				

Table 5.1: Flows admitted with ESS, PSS, OPQ, and LSS algorithms. Hops=7, $\rho_i^{avg} = 100Kbps$ and $\sigma_i = 5\ Kbits$.

because LSS explicitly attempts to balance the loads across different links. In contrast, ESS algorithm does not optimize any specific metric and PSS algorithm does not explicitly balance the loads among the links. Similarly we see that the performance obtained with OPQ algorithm is worse than that with LSS. The main problem lies not within the OPQ algorithm itself, but in coming up with a cost metric that accurately captures the load balancing criteria. In this case, OPQ algorithm performs its work of coming up with a solution that is close to optimal in minimizing the specific cost metric; however, the best cost-metric we can construct turns out to be only an approximation of the final load balancing optimization objective. Instead of implicitly capturing the load balancing criteria by means of a cost function, the LSS algorithm approaches the problem in a reverse fashion by exploring only those slack partitions that maintain the required load balance among the links.

5.6.4 Capacity Evolution

In order to understand why the LSS algorithm admits a higher number of flow requests than other algorithms, we compare their resource usage patterns. Figures 5.14 and 5.15 plot the evolution of available link bandwidth on the constituent links of the Unicast-1

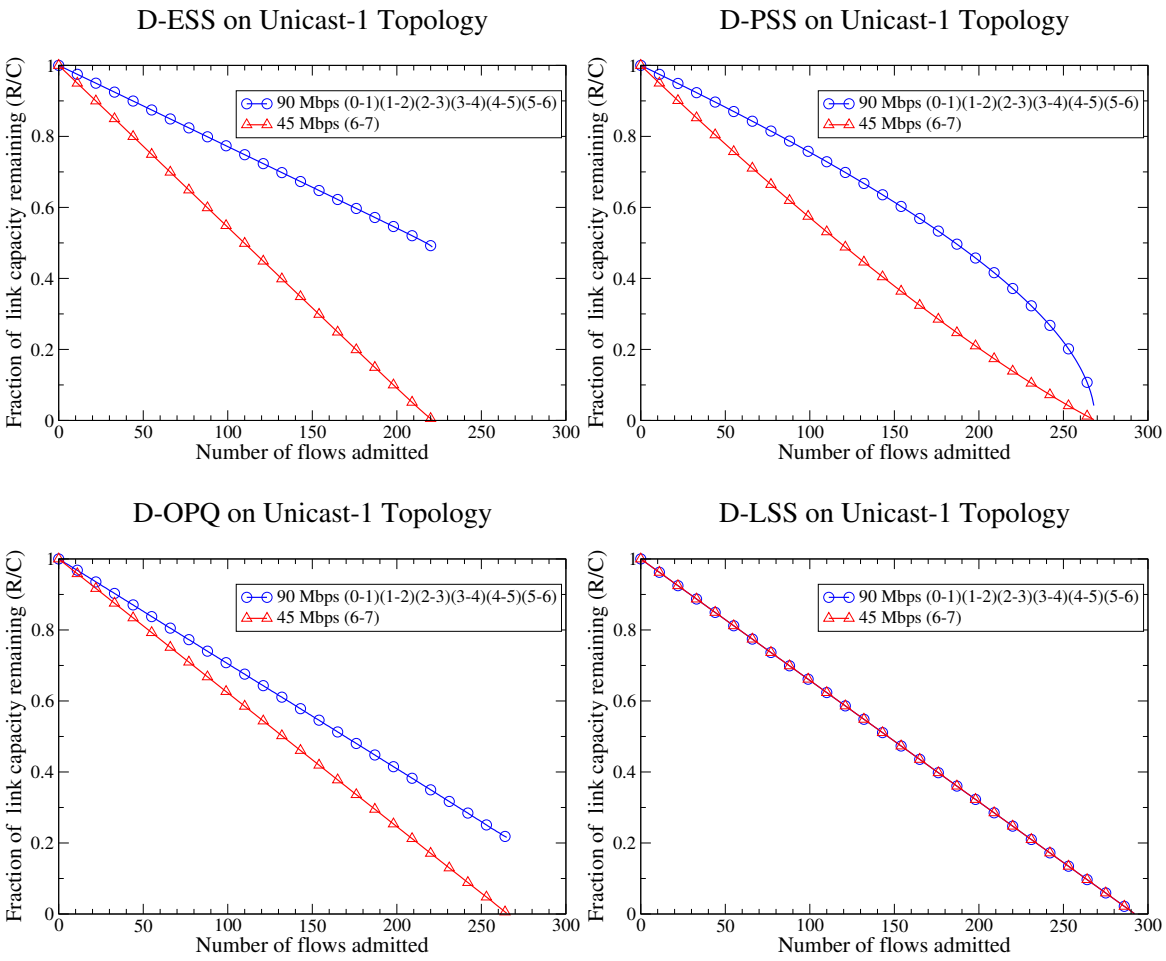

Figure 5.14: Evolution of available link capacity with the number of flows admitted for unicast flows with deterministic delay guarantee. Unicast-1 topology, Hops=7, $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$.

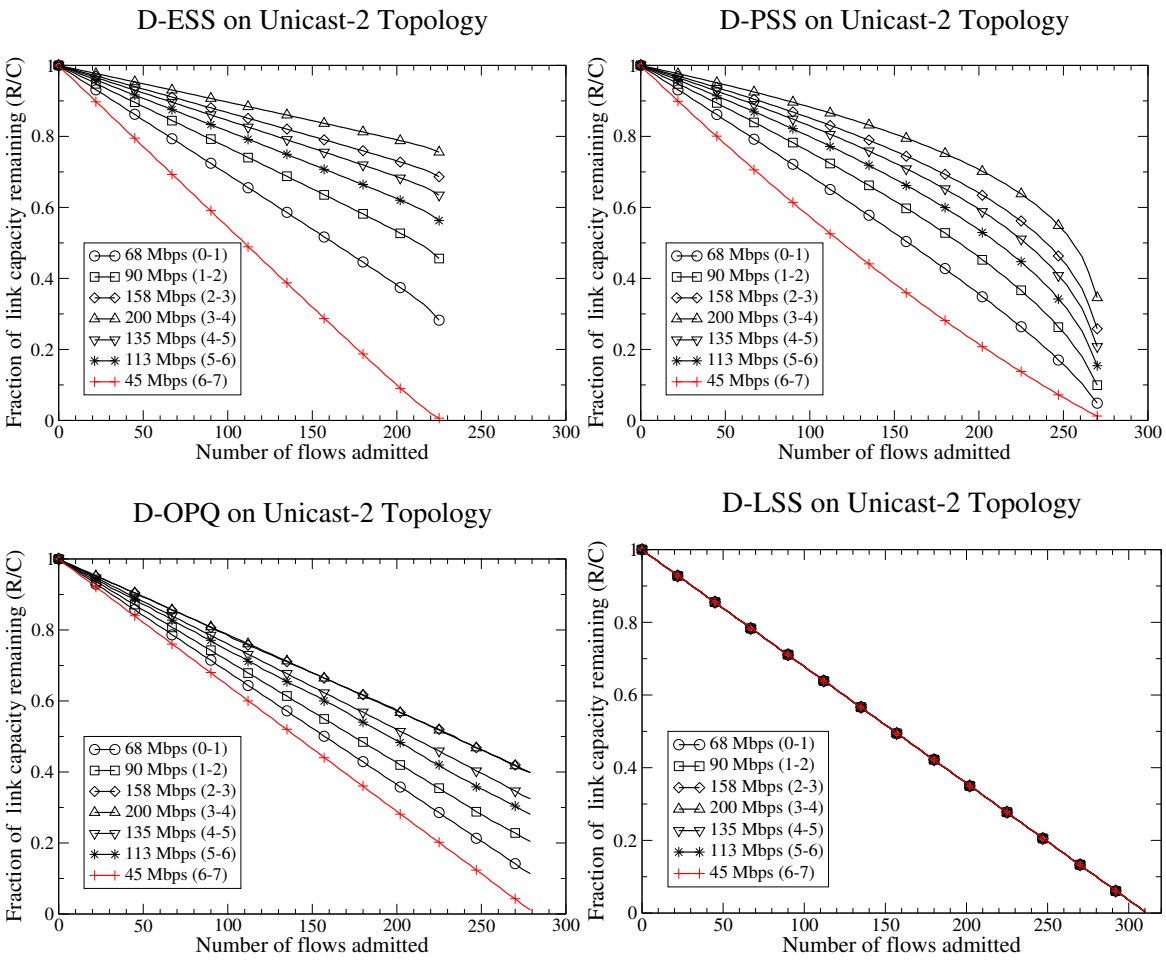

Figure 5.15: Evolution of available link capacity with the number of flows admitted for unicast flows with deterministic delay guarantee. Unicast-2 topology, Hops=7, $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$.

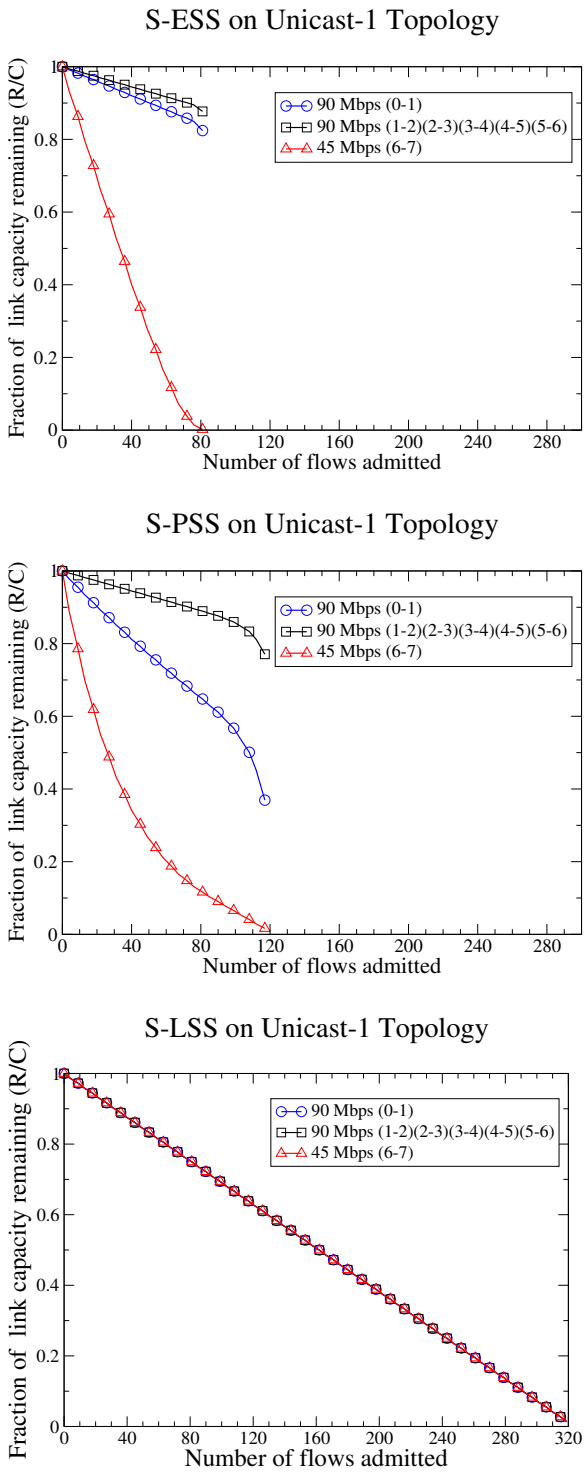

Figure 5.16: Evolution of available link capacity with the number of flows admitted for unicast flows with statistical delay guarantee. Unicast-1 topology, Hops=7, $\rho_i^{avg} = 100Kbps$, $D_i = 45ms$, $P_i = 10^{-5}$ and $\sigma_i = 5\ Kbits$.

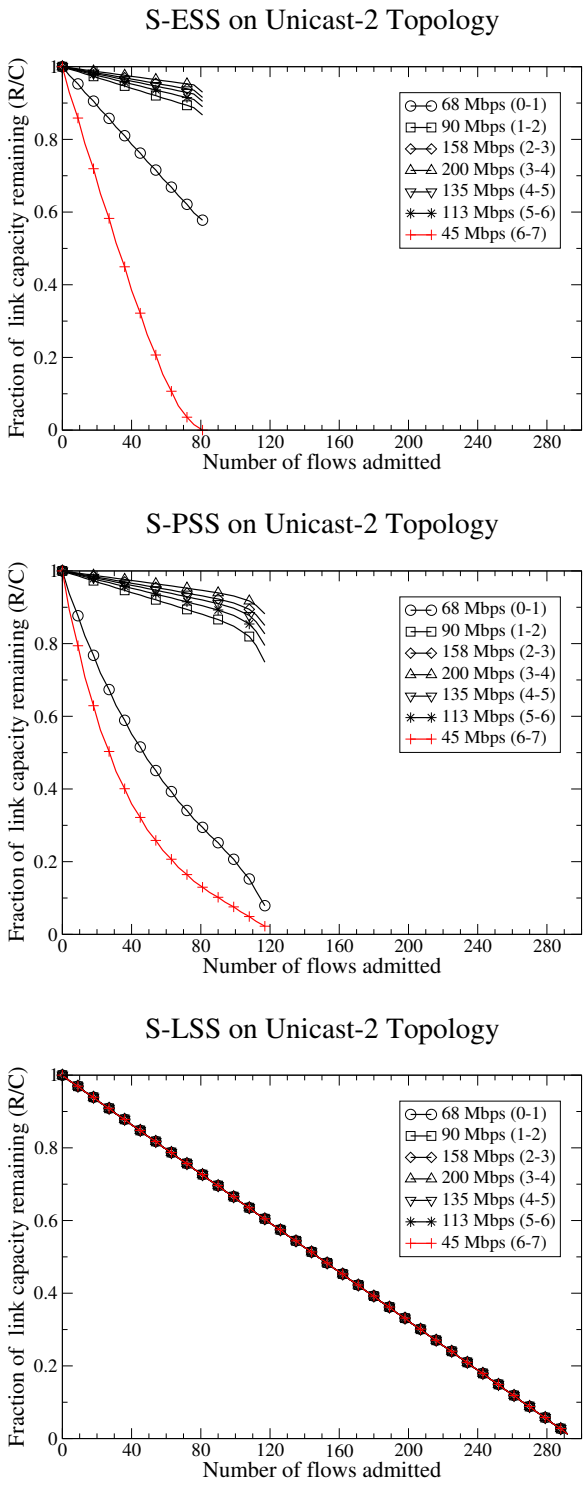

Figure 5.17: Evolution of available link capacity with the number of flows admitted for unicast flows with statistical delay guarantee. Unicast-2 topology, Hops=7, $\rho_i^{avg} = 100Kbps$, $D_i = 45ms$, $P_i = 10^{-5}$ and $\sigma_i = 5\ Kbits$.

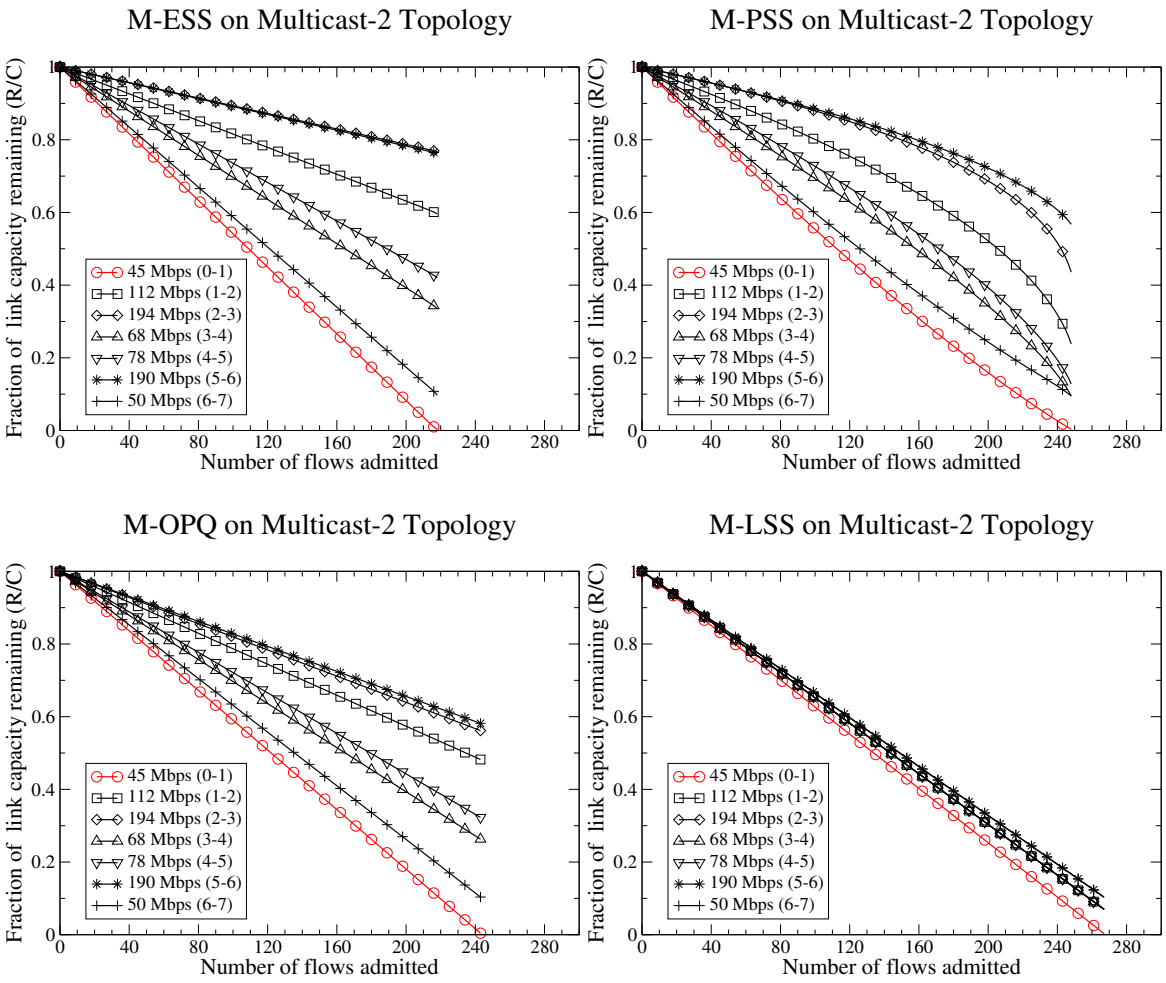

Figure 5.18: Evolution of available link capacity with the number of flows admitted for multicast flows with deterministic delay guarantee. Multicast-2 topology, Tree depth=7, $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$.

and Unicast-2 topologies when flows require a deterministic end-to-end delay bound of 60ms. Figures 5.16 and 5.17 plot the same curves when flows require statistical end-to-end delay bound of 45ms and delay violation probability of 10^{-5}. We used a smaller statistical delay bound of 45ms compared to the deterministic delay bound of 60ms in Figures 5.14 and 5.15 because, for statistical case, the difference in performance among different algorithms is evident only at smaller delay bounds. Figure 5.18 plots the same curves for Multicast-2 topology with tree depth of 7 when multicast flows require end-to-end deterministic delay bound of 60ms.

At any point in time, LSS is able to explicitly balance the loads on constituent links. On the other hand, the link loads are imbalanced in the case of ESS, PSS and OPQ algorithms. Specifically, the bandwidth of the links with lower capacity is consumed more quickly than that of links with higher capacity and consequently fewer number of flows are admitted. Note that a single link with insufficient residual capacity is enough to render the entire network path unusable for newer reservations. Among ESS, PSS and OPQ algorithms, OPQ and PSS have similar performance followed by the ESS. The differences in performance arise from the extent to which each algorithm accounts for load imbalance between links. For multicast flows in Figure 5.18, the capacity evolution curves are not perfectly balanced in the case of D-MLSS due to the fact that the assigned bandwidth on some of the links is lower-bounded by the 100 Kbps average rate of flows which is larger than the 'delay-derived' bandwidth required to satisfy the delay budget at those link.

5.6.5 Effect of End-to-end Delay

Figure 5.19 plots the variation in number of admitted flows over different topologies as their end-to-end deterministic delay requirement is varied. With increasing delay, all the four algorithms admit more number of flows, since a less strict end-to-end delay bound translates to lower resource requirement at intermediate links. Again, LSS admits more flows than others since it performs load balanced partitioning of the slack in end-to-end delay. A maximum of 450 flows with 100 Kbps average data rate can be admitted over the Unicast-2 and Multicast-2 topologies by any of the algorithms since the smallest link capacity is 45 Mbps.

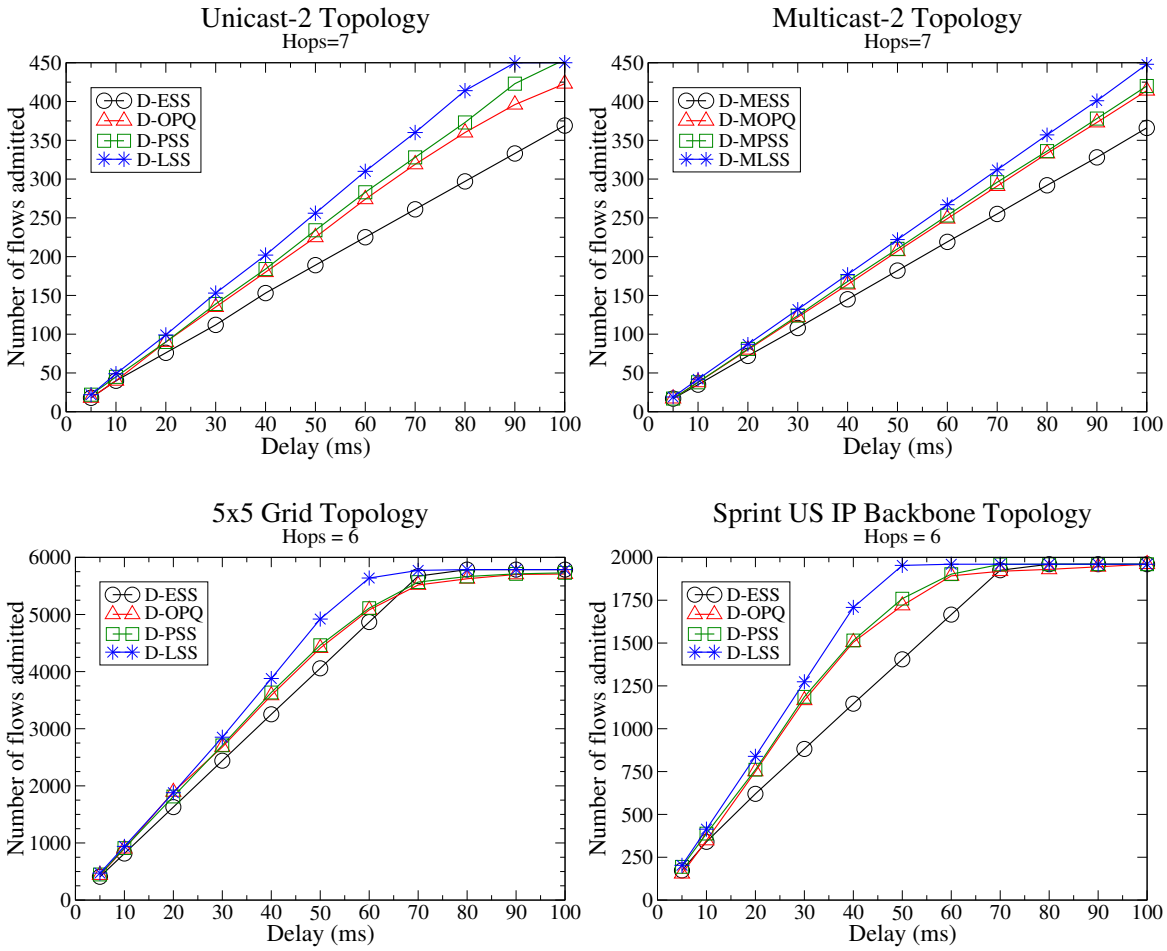

Figure 5.19: Number of flows admitted vs. deterministic end-to-end delay bound for unicast and multicast paths. $\rho_i^{avg} = 100Kbps$ and $\sigma_i = 5\ Kbits$.

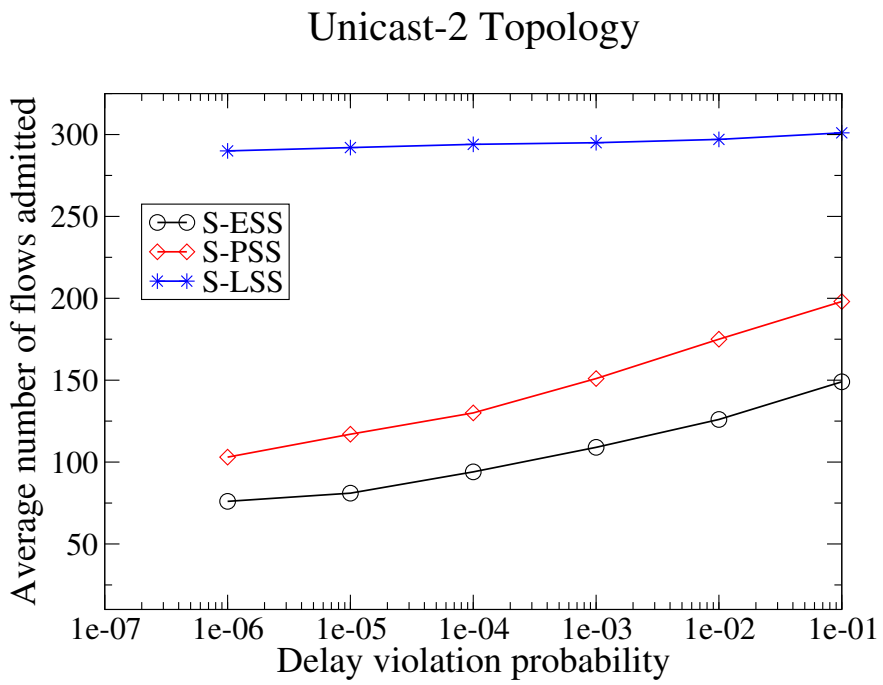

Figure 5.20: Average number of flows admitted vs. end-to-end delay violation probability bound. Hops=7, $D_i = 45ms$, $\rho_i^{avg} = 100Kbps$ and $\sigma_i = 5\ Kbits$. Averages computed over ten simulation runs with different random seeds.

5.6.6 Effect of End-to-end Delay Violation Probability

Figure 5.20 plots the variation in average number of admitted flows over unicast and multicast paths as their delay violation probability bound is varied from 10^{-6} to 10^{-1} for a 45ms end-to-end delay bound. The LSS algorithm is able to admit far more flows than ESS and PSS algorithms since it can perform load balanced slack sharing along both the dimensions of delay as well as delay violation probability. The performance gain for LSS over ESS and PSS is much larger than in the case of deterministic delay requirements (Figure 5.19) because even a small increase in delay violation probability yields a significant reduction in resources assigned to a flow.

5.6.7 Effect of Path Length

Figure 5.21 plots the variation in number of admitted flows having deterministic delay requirements of 60ms, as the path length/tree depth increases. For all the four algorithms,

there is a drop in the number of flows admitted with increasing path length because the same end-to-end delay now has to be partitioned among more number of intermediate hops. Thus increasing the path length has the effect of making the end-to-end requirement more strict, which in turn translates to higher resource requirement at the intermediate links. For Unicast-2 and Multicast-2 topologies, the LSS algorithm still outperforms the other three algorithms in terms of number of flows it can support since it manages the decreasing slack in delay more efficiently than others. For Grid and Sprint topologies, performance of LSS is similar to ESS algorithm for small paths lengths (less than 5); PSS and OPQ perform worse than both ESS and LSS for small path lengths. This is due to the fact that at smaller path lengths, flow reservations are mainly guided by their average rate constraint rather than delay constraints and using ESS turns out to be more prudent in a general network context. However, with increasing path lengths (larger than 5), LSS admits up to 12% more flows than PSS, which performs the next best. This is due to the fact that at path lengths greater than 5, flow reservations begin to be guided by their delay constraints (rather than average rate) and the benefits of using LSS becomes apparent.

5.7 Summary of Contributions

In this chapter, we have addressed the problem of partitioning the end-to-end delay constraints into per-link delay constraints. We made the following research contributions.

- Firstly, we proposed a new algorithm, called Load-based Slack Sharing (LSS), to solve the delay budget allocation problem, and describe its application to unicast and multicast flows with deterministic and statistical delay guarantees.

- Secondly, LSS handles multiple simultaneous flow QoS requirements such as end-to-end delay bounds and delay violation probability bounds. A simple algorithmic structure allows LSS to be potentially extended to handle more additive/multiplicative QoS requirements as well. Earlier approaches handled only a single end-to-end QoS requirement at a time.

- Thirdly, we introduce the notion of partitioning *slack* in end-to-end QoS rather than directly partitioning the entire end-to-end QoS as in earlier approaches. Slack

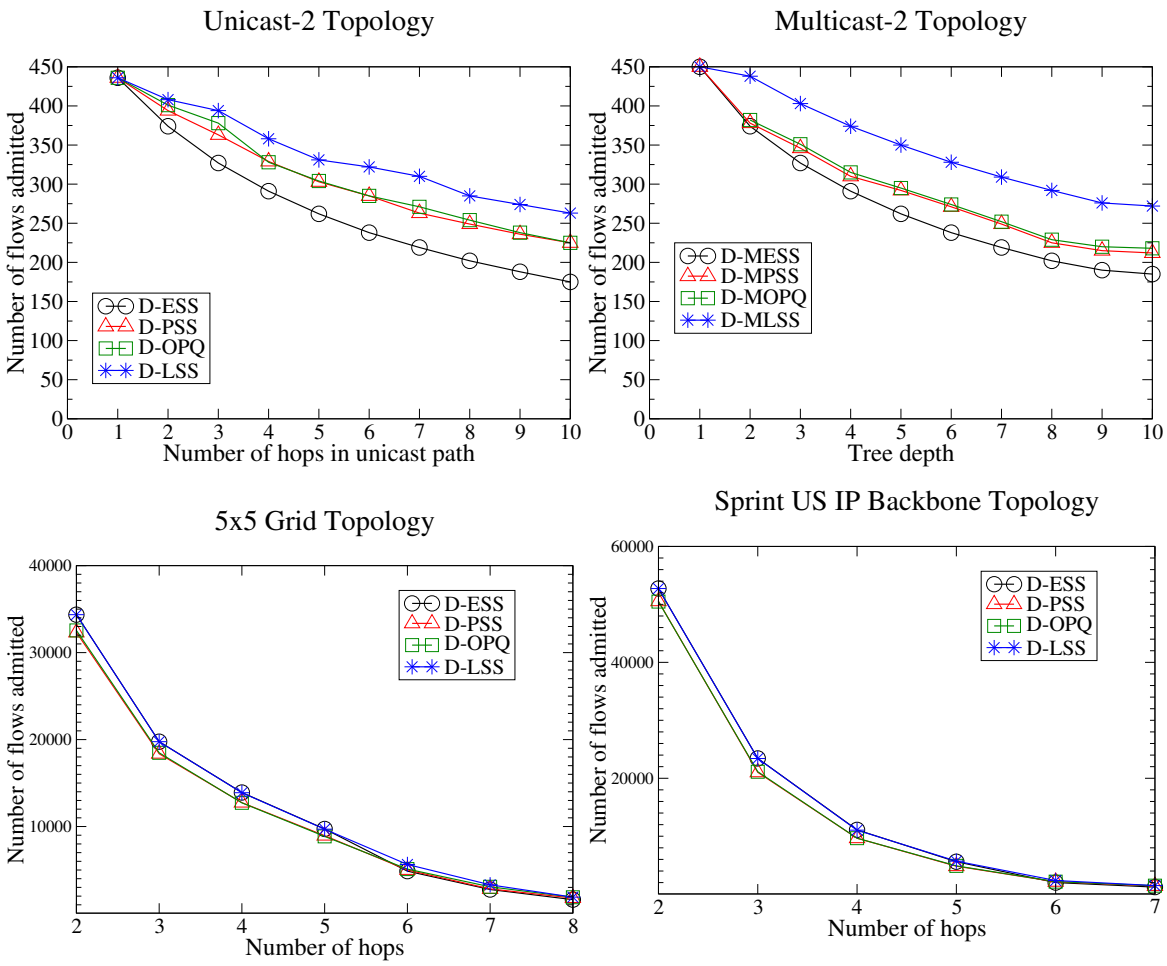

Figure 5.21: Number of flows admitted vs. path length. $D_i = 60ms$, $\rho_i^{avg} = 100Kbps$ and $\sigma_i = 5\ Kbits$.

quantifies the amount of flexibility available in balancing the loads across links of a multi-hop path and was introduced in Section 5.2.

- Finally, we develop the detailed admission control algorithms for unicast and multicast flows that can be used in conjunction with any scheme for partitioning end-to-end delay and delay violation probability requirements.

Through a detailed simulation study, we have shown that the LSS algorithm can admit larger amount of traffic in comparison with three other algorithms proposed in the literature. The improvement is up to 1.2 times for flows with deterministic delay bounds and 2.8 times for statistical delay bounds. LSS achieves higher resource utilization because it explicitly maintains intra-path load balance and ensures that that none of the constituent links of a path exhausts its capacity long before others. Since smaller delay budget implies higher link-load and vice versa, LSS allocates a larger share of the delay slack to more loaded links than to less loaded links, thus reducing the load deviation among these links. Similar loads on constituent links implies that capacity on all links degrades at the same rate and no single link becomes a bottleneck preventing future flows requests from being admitted. By avoiding such load imbalance among the links of a path, resource fragmentation is less likely and more flows can be admitted in the long term.

Chapter 6

Primary and Backup Route Selection

6.1 Problem Description

Traffic engineering deals with the problem of utilizing network resources efficiently while supporting traffic with QoS constraints. The process of determining the route taken by each traffic flow is the most critical component of the traffic engineering process since it involves resource allocation decisions at the scale of the entire network - in contrast to just link-level or intra-path decisions that were addressed in earlier chapters - and has the maximum impact on network resource usage efficiency. The problem of route selection with QoS constraints arises in the context of Multi-Protocol Label Switching Networks (MPLS) where routes for Label Switched Paths (LSP) need to be explicitly set up between network endpoints. Each LSP carries an aggregated traffic flow, where aggregation is performed at the network edge. The traffic carried over an LSP may require different forms of QoS guarantees depending upon the traffic type. For instance, if the LSP acts as a long-distance VoIP trunk that carries real-time voice traffic, then the traffic will require long-term bandwidth and end-to-end delay guarantees.

Explicit routing of LSPs provides a significant benefit over the traditional hop-by-hop routing, which forwards packets along shortest-paths based purely on destination address in the packet header. Hop-by-hop routing might tend to forward packets along already congested shortest paths while other uncongested paths may never be utilized. Traffic engineering provides the opportunity to explicitly select the route taken by an LSP such

that different parts of the network remain equally loaded, thus maintaining a high network resource utilization efficiency.

Earlier research efforts on traffic engineering, such as [87, 47, 66, 107], have primarily addressed the bandwidth dimension of QoS requirements for traffic flows. However, with the increasing number of network applications that generate time-sensitive network traffic, the delay dimension of the QoS requirements is having a more significant impact on traffic engineering algorithms. Solutions proposed in [50, 86, 34, 74] consider the delay dimension as well. However their focus is not on the impact of the delay dimension on traffic engineering considerations. These solutions either deal with fixed propagation delays and fixed link costs [50, 86], or consider finite values of discrete link delays and delay-dependent costs [34, 74]. Furthermore, they do not handle multiple delay constraints such as end-to-end delay and delay violation probability and cannot incorporate the fault-tolerance consideration of provisioning a backup path for each primary path.

In this chapter, we address the problem of selecting primary and backup routes for real-time network flows that have both long-term bandwidth and end-to-end delay bound requirements. We propose an algorithm called *link criticality based routing* (LCBR) to select primary and backup routes that satisfy flow QoS requirements while improving the long-term resource usage efficiency of the network. One of the unique aspects of LCBR is that it explicitly incorporates traffic engineering considerations into the delay-constrained path selection process. The basic intuition behind LCBR is to select those routes that best maintain the load balance across the entire network.

An important consideration in LCBR algorithm is to incorporate the impact of intra-path and link-level resource provisioning algorithms on the network-level route selection decisions. Specifically, when a new flow is assigned a route, we need to partition the end-to-end delay requirements of the flow along every link in the route. Similarly, we need to make sure that the delay requirement assigned to each link can indeed be supported at those links. Such link-level and intra-path provisioning algorithms were presented in Chapters 4 and 5. Provisioning resources for a flow at the lower levels has a direct impact of increasing the loads on the links of the route selected. For instance, a route with very low network load before admitting a new flow might become fully loaded after admitting the flow. LCBR takes this dependence into account by consulting with intra-path and link-

level provisioning algorithms. On the other hand, prior approaches that rely upon some form of shortest path algorithm, such as Widest Shortest Path (WSP) [47] and MIRA [66], cannot account for these inter-dependencies during online route-selection.

The rest of this chapter is organized as follows. In Sections 6.2 we present the LCBR algorithm for selecting primary route alone. In Section 6.3 we extend the algorithmic framework of Section 6.2 for simultaneously selecting both primary and backup routes. Section 6.4 presents performance evaluation of LCBR and finally, Section 6.5 provides a summary of contributions from this chapter.

6.2 Link Criticality Based Routing for Primary Route Selection (P-LCBR)

First we consider the problem of selecting a primary route that carries real-time traffic for a new flow. In the next section, we will see how the same algorithmic framework can be extended to perform simultaneous primary and backup route selection. In the following discussion, we focus on the end-to-end delay bound and average rate requirements of new flows. LCBR can be trivially extended to incorporate the delay violation probability requirement as well.

We need to select a primary route and provision resources for a new flow request F_N between source node s and destination node d. F_N requires an end-to-end delay bound of D_N, and an average bandwidth of ρ_N^{avg}. We represent this requirement by the tuple $(s, d, D_N, \rho_N^{avg})$, where N is a network-wide unique identifier. We need to select a primary route X_N and allocate resources along the route for the flow F_N, such that the following conditions are satisfied.

- $\sum_{l \in X_N} D_{N,l} \leq D_N$
- The bandwidth reservation at every link along the route X_N is at least ρ_N^{avg}.

The main intuition behind LCBR is to find a route that balances the loads across different parts of the network and keeps the more critical links available for future flow requests. Note that the algorithm does not have any advance knowledge of any future requests,

except the one currently being provisioned, and does not reallocate already admitted flows in order to accommodate a new request. In this sense, our proposal is an *on-line* route selection algorithm.

6.2.1 Expected Load and Network Cost

In this section, we define a network-wide cost metric $cost(G)$ that measures the extent of load on the resources in a network G. An important factor in computing $cost(G)$ is the notion of *expected load* ϕ_l on each link l. The expected load ϕ_l indicates the importance of a link l in terms of how critically the different source-destination pairs in the network need the link for carrying their traffic. Assume that a total of x network routes are possible between a source-destination pair (s, d). Out of these, say y routes pass through link l. The criticality $\phi_l(s, d)$ of the link l with respect to source-destination pair (s, d) is defined as the fraction of routes between s and d that pass through link l, i.e.,

$$\phi_l(s, d) = y/x \tag{6.1}$$

Assume that we have the knowledge of expected traffic demand $B(s, d)$ which represents the total bandwidth demand expected between the source-destination pair (s, d). For instance, such demands could be measured on a daily basis or derived from service-level agreements. The expected load ϕ_l on link l is defined as the sum of fractional expected demands on the link from all possible source-destination pairs in the network, i.e,

$$\phi_l = \sum_{(s,d)} \phi_l(s, d)B(s, d) \tag{6.2}$$

Only a small subset of all the nodes in the network are typically possible sources or destinations for flow traffic. Hence computing ϕ_l does *not* involve an exhaustive computation of all the n^2 possible values of $\phi_l(s, d)$, where n is the number of nodes in the network. Furthermore, link criticality $\phi_l(s, d)$ is largely static since it is completely determined by topology of the network and changes only when the topology changes. Also, $B(s, d)$ changes relatively infrequently, such as on a daily basis. Thus the values of ϕ_l can be periodically pre-computed offline and kept ready for use in the online route selection phase described next.

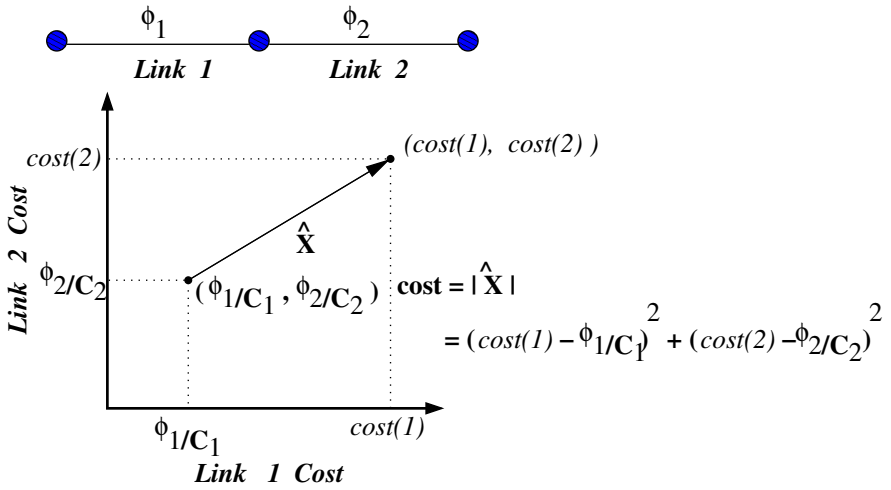

Figure 6.1: Cost of a 2-link route in terms of link costs. Cost is simply the squared magnitude of the vector $\hat{X}$ connecting point of minimum link cost $(\frac{\phi_1}{C_1}, \frac{\phi_2}{C_2})$ to the current cost point $(cost(1), cost(2))$.

Let C_l be the total capacity and R_l be the remaining capacity of the link l at any instant. The cost $cost(l)$ of link l is defined as follows.

$$cost(l) = \frac{\phi_l}{R_l} \tag{6.3}$$

The metric $cost(l)$ represents the expected load per unit of available link capacity at the link. A link l with small residual capacity R_l or large expected load ϕ_l is more expensive for use in flow routes. Note that the most dynamic component of the link cost $cost(l)$ is the residual link capacity R_l. The minimum value of link cost is ϕ_l/C_l when the residual capacity is maximum at $R_l = C_l$.

The cost metric $cost(G)$ for the entire network G is defined as the sum of $(cost(l) - \frac{\phi_l}{C_l})^2$ for all links in the network G.

$$cost(G) = \sum_{l \in G} \left(cost(l) - \frac{\phi_l}{C_l} \right)^2 \tag{6.4}$$

The metric $cost(G)$ represents the squared magnitude of distance vector between the actual link costs and the link costs in the network G. Figure 6.1 geometrically illustrates the significance of $cost(G)$ for a route with two links that have expected loads ϕ_1 and ϕ_2. Ideally, we would like the state of the network to be as close to the idle-state operating

point $(\phi_1/C_1, \phi_2/C_2)$. Squared sum has the advantage that it captures the impact of both magnitude of individual link costs and the variations among them.

6.2.2 The P-LCBR Algorithm

The *link criticality based routing* (P-LCBR) algorithm for selecting primary route alone for a flow F_N is presented in Figure 6.2. The goal of P-LCBR algorithm is to select a route that minimizes the metric $cost(G)$. The algorithm first computes $cost(l)$ for each link l in network G. For each route r that belongs to the set of k shortest-path routes between s and d, P-LCBR applies the admission control algorithm in Section 5.3.1 to check if the QoS requirements (D_N, ρ_N^{avg}) can be satisfied by the available resources along the route r. If there are sufficient resources along a route, then P-LCBR invokes the route-level delay partitioning algorithm in Figure 5.2 to divide the end-to-end delay D_N among the links of route r. The next step recomputes the projected remaining capacity R'_l at each link l that would result if flow F_N were to be admitted. The projected R'_l values are then used to recompute the metric $cost(G)$.

Flow F_N is rejected if either (a) no route has sufficient resources to satisfy F_N's QoS requirements or (b) the minimum value of $cost(G)$ is greater than a pre-defined fraction cost threshold α. Latter case indicates that admitting F_N would take the network to a highly critical state that may not be conducive for admitting future flow requests. Effectiveness of the P-LCBR algorithm depends on two main factors. First parameter is k - the number of shortest-path routes whose network-wide cost impacts are examined. The second parameter α determines the cost threshold for rejecting flow requests. Well known algorithms exist for discovering k-shortest paths [33, 48] between a given source-destination pair. The logic behind using the k-shortest paths framework is that shorter paths tend to utilize fewer network resources in comparison to longer paths. Thus it is very likely that the route which best minimizes the vector distance from idle-state network cost is one of the k-shortest candidate paths.

Input: Network topology G.

New flow specification (s, d, D_N, ρ_N)

Expected load ϕ_l, residual capacity R_l and total capacity C_l for all links l in the network.

Set S of k shortest path routes between source-destination pair (s, d).

Output : Primary route X_N

For all links l in G do

$cost(l) = \frac{\phi_l}{R_l}$

$cost_{min} = \infty$; $X_N = nil$

For each route r in the set S do

If (D_N, ρ_N) cannot be satisfied along route r, then skip to next route.

Partition the end-to-end delay D_N among individual links of route r.

Recompute the resulting residual capacities R'_l for each link $l \in r$.

Recompute the link costs $cost(l) = \frac{\phi_l}{R'_l}$ for each link $l \in r$.

Recompute the metric $cost(G) = \sum_{l \in G} \left(cost(l) - \frac{\phi_l}{C_l} \right)^2$.

If $cost(G) < cost_{min}$ then

$cost_{min} = cost(G)$; $X_N = r$.

If $(cost_{min} > \alpha)$ then

Reject F_N since either resources are insufficient

or F_N makes the network critical over acceptable threshold.

else

Select route X_N as primary route for F_N and reserve resources along X_N

Figure 6.2: *The P-LCBR algorithm* selects the primary route for a flow F_N between source s and destination d. F_N requires an end-to-end delay bound of D_N and average bandwidth of ρ_N^{avg}.

6.3 Link Criticality Based Routing for Primary-Backup Route Selection (PB-LCBR)

Now let us consider the problem of finding both a primary route X_N and a backup route Y_N for a new flow request F_N. We refer to a link or node in the network as a *network element*. The backup route provides the guarantee that if at most one network element fails and the network element happens to lie on the primary route X_N, then the flow F_N's traffic would be diverted to backup route Y_N. The backup route Y_N provides the same QoS guarantees as the primary route X_N on delay bound D_N, delay violation probability bound P_N, and average bandwidth ρ_N^{avg}. Thus PB-LCBR guards against the possibility of a single network element failure.

Note that the basic requirement for being able to tolerate single-element failures is that the primary and backup routes of a flow must be completely disjoint with respect to all intermediate network elements. This is to ensure complete switch-over to backup route if any one element fails along the primary route. If any network element was to be shared between primary and backup routes, and the shared element happened to fail, then both primary and backup routes would be disabled simultaneously.

6.3.1 Backup Resource Aggregation

A simple technique to provision resources for a backup route is to set up an additional route through the network that is completely disjoint from the primary path and provides the same QoS guarantees. The only difference would be that the backup path would lie dormant till there is a failure along the primary path, at which time the traffic from real-time flow could be switched to the backup path. Given that failures do not occur with a high frequency in typical networks, the above approach also implies that roughly half of the network resources reserved along the backup routes would remain unutilized most of the time.

Since the central goal of this dissertation is to maximize network resource utilization efficiency, the resources allocated along backup routes must be optimized to the maximum extent possible. Towards this end, we now introduce the notion of *backup resource aggregation*, which exploits the fact that flows whose primary routes are completely in-

dependent of each other can share their resource reservations on the common links along their backup routes. Backup resource aggregation was first proposed in the RAFT approach [30] in the context of flows that require bandwidth guarantees alone. Our approach to backup resource aggregation is similar but in the context of delay-bandwidth constrained flows. Our additional contribution here is the application of the resource aggregation concept within the comprehensive admission control and resource allocation framework of LCBR algorithm in the context of bandwidth-delay constrained flows.

Two flows are said to *intersect* with each other at a network element e if both their primary routes pass through element e. Whenever a network element e fails, we need to activate the backup routes for all the flows that intersect at element e. Every link l has one *primary set* $Primary(l)$ that contains the IDs of all flows whose primary routes pass through the link l. In addition, each link has a total of $(m + n)$ *backup sets* of flow reservation, where each set corresponds to one network element; m is the number of links and n is the number of nodes in the entire network. Each backup set at link l is represented by $Backup(l, e)$, $1 \leq e \leq (m + n)$, where each backup set corresponds to one network element e. The backup set $Backup(l, e)$ contains IDs of those flows whose backup routes traverse link l and whose primary routes intersect at the network element e. In other words, $Backup(l, e)$ represents the set of backup reservations at link l that need to be activated in the event of failure of network element e.

Figure 6.3 and Table 6.1 illustrate the concept of primary and backup sets. The network consists of 4 nodes $n1$, $n2$, $n3$, and $n4$. There are three flows: A (from $n1$ to $n4$), B (from $n1$ to $n4$), and C (from $n3$ to $n2$). The primary route for A (solid line) traverses the route $n1$–$n3$–$n4$ and its backup route (dashed line) traverses a completely disjoint route $n1$–$n2$–$n4$. Similarly for flow B and C. Table 6.1 gives the primary set and backup sets at link l_2. Link l_2 carries primary route for flow A and backup routes for flows B and C. Note that the set $Backup(l_2, l_3)$ contains both B and C since their primary routes pass through l_3 whereas their backup routes traverse through l_2.

Flows whose primary routes do not intersect with each other at any network element do not appear together in any backup set. For instance A and B do not appear together in any backup set. A flow F_N, whose primary route X_N traverses h links (and hence $h - 1$ intermediate nodes) from source to destination, will be part of $(2h - 1)$ backup

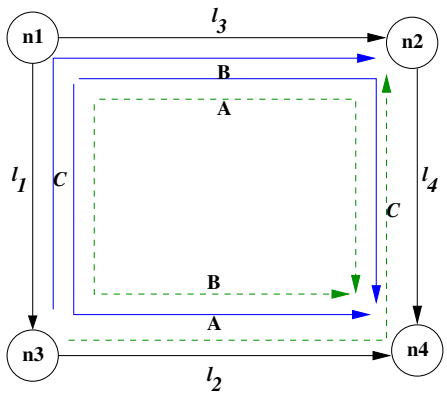

Figure 6.3: Backup path resource aggregation. Example of a network with 4 nodes on which three flows A, B, and C are set up. Primary routes are represented by bold lines and backup routes are represented by dashed lines. Reservations for backup routes are activated only when any network element along the primary route fails.

sets at every link along its backup route Y_N. For instance, B is part of 3 backup sets at link l_2 since its primary route is 2 hops long and contains 3 network elements (2 links and 1 node) that can potentially fail. Note that failure of source and destination nodes of a flow is not covered by the single-failure model since such a failure is fatal to the flow by default.

Recovery from failure of a network element occurs as follows. During normal operations, each link scheduler operates with the reservations for flows in its primary set $Primary(l)$. Whenever a network element e fails, its corresponding backup sets $Backup(l, e)$ are activated at all the links l of the network. Activating a backup set at a link l implies that the link scheduler starts operating with reservations in its primary set $Primary(l)$ plus the reservations in its backup set $Backup(l, e)$.

6.3.2 The PB-LCBR Algorithm

The goal of primary-backup route selection is to select a disjoint route pair that minimizes $cost(G)$. There are two approaches to select a primary path and a backup path. A greedy approach that involves less pre-computation, but might result in inefficient resource allocation and a non-greedy approach that achieves more efficient resource allocation at a slightly higher pre-computation cost.

	Set
$Primary(l_2)$	$\{A\}$
$Backup(l_2, n_1)$	$\{C\}$
$Backup(l_2, n_2)$	$\{B\}$
$Backup(l_2, n_3)$	$\{\}$
$Backup(l_2, n_4)$	$\{\}$
$Backup(l_2, l_1)$	$\{C\}$
$Backup(l_2, l_2)$	$\{\}$
$Backup(l_2, l_3)$	$\{B, C\}$
$Backup(l_2, l_4)$	$\{B\}$

Table 6.1: One primary set and eight backup sets at link l_2 in Figure 6.3.

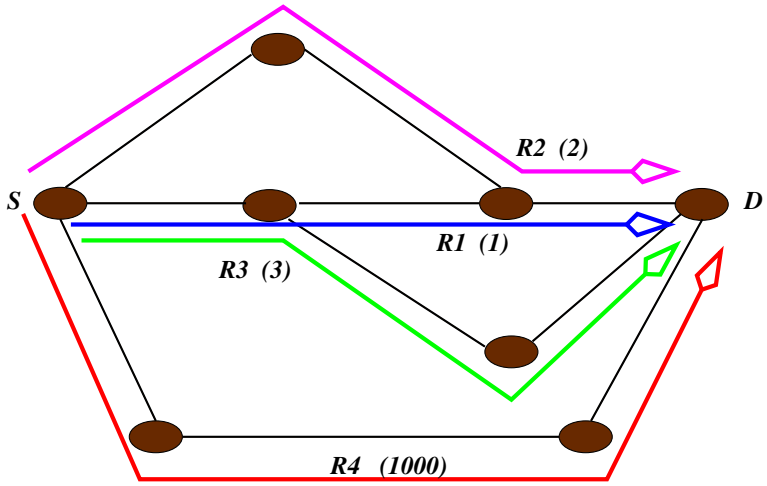

Figure 6.4: Disjoint Route Pair Selection Problem. There are 4 possible routes between S and D: $R1$, $R2$, $R3$ and $R4$. Values in braces represent cost of selecting the corresponding route. Using greedy approach we would first select $R1$ as primary route, $R4$ as the only disjoint backup route, and obtain a total cost of 1001. The best choice for primary and secondary routes are $R2$ and $R3$ respectively with total cost of 5.

6.3.2.1 The Greedy PB-LCBR Algorithm

Let us represent the primary reservation for a flow F_i at link l by $\rho_{i,l}$ and its backup reservation at link k by $\beta_{i,k}$. The metrics $cost(G)$ is defined as in Equation 6.4, the only difference being that we now need to assume that any network element can fail. Hence, the residual capacity R_l of a link l is calculated as follows.

$$R_l = C_l - \sum_{i \in Primary(l)} \rho_{i,l} - \max_{All\ Network\ Elements\ e} \sum_{i \in Backup(l,e)} \beta_{i,l} \tag{6.5}$$

In other words, the residual capacity R_l is obtained by deducting the sum of primary reservations and the maximum of the sum of backup reservations in each backup set from the total link capacity C_l.

A simple greedy approach to selecting primary and backup routes is as follows. First select a primary route X_N that yields smallest value of $cost(G)$ using the algorithm in Figure 6.2. Then derive a reduced network graph G' from G by removing all the network elements in the route X_N. Finally, use the algorithm similar to one in Figure 6.2 to select a backup route Y_N from the reduced graph G' that yields the smallest value of $cost(G)$.

The greedy PB-LCBR algorithm outlined above is simple and ensures that the primary and backup routes chosen are indeed disjoint. However, the approach leads to inefficient and skewed solutions. For instance consider the network shown in Figure 6.4 in which we need to select a primary and a backup route between S and D. There are four possible routes labeled $R1$, $R2$, $R3$ and $R4$. Assume that the value of metric $cost(G)$ upon selection of each of the four routes is 1, 2, 3, and 1000 respectively. The greedy approach would first select route $R1$ as the primary route since it yields $cost(G) = 1$. This would leave route $R4$ as the only choice of disjoint backup route in the residual graph G' resulting in a total primary-backup cost of 1001. However it is obvious that the lowest-cost choice should be $R2$ as the primary route and $R3$ as the secondary route for a total $cost(G) = 5$. The main reason for this inefficiency is the fact the greedy approach decouples the selection of primary and backup routes into two separate phases. Our second non-greedy approach addresses this problem.

6.3.2.2 The Non-Greedy PB-LCBR Algorithm

The non-greedy PB-LCBR algorithm overcomes the problem associated with the greedy approach by simultaneously examining candidate primary-backup route pairs.

The PB-LCBR algorithm for selecting both primary and backup routes is presented in Figure 6.5. As input to the algorithm, we supply the set of pre-computed route pairs (X, Y) of potential primary and backup routes between source s and destination d. Routes X and Y in each pair are physically disjoint from each other to satisfy the single failure recovery constraint. There are $k = k1 \times k2$ pre-computed candidate route pairs which consist of $k1$ shortest primary routes and for each primary route, the $k2$ shortest backup routes. For each primary-backup pair (X, Y) that belongs to the set of k smallest length pairs between s and d, PB-LCBR uses the admission control algorithm in Section 5.3.1 to check if the QoS requirements of flow F_N can be satisfied by the available resources along both the routes X and Y. If there are sufficient resources, then PB-LCBR invokes the route-level delay partitioning algorithm to divide the delay D_N among links of routes X and Y and to recompute the projected residual capacities R'_l at each link l. The projected R'_l values are then used to recompute the metric $cost(G)$. The candidate route pair that yields the minimum value of $cost(G)$ is then selected as the primary-backup route for flow F_N.

Upon failure of a network element, the flows whose primary routes pass through the failed component are switched over to their backup routes. What happens to the flows whose backup routes pass through the failed component? Remember that we guarantee recovery from failure of only single network element at a time. The two alternatives we have are to either recalculate the failed backup routes or to fix the failed component before the next failure occurs. Essentially, the notion of single link failure assumes that the second failure does not occur before the first failure is repaired.

6.4 Performance Evaluation

In this section, we evaluate the improvement in resource usage efficiency due to LCBR algorithm. The measure of resource usage efficiency we use is the number of flows admitted with end-to-end QoS guarantees.

Input: Network topology G.

New flow specification (s, d, D_N, ρ_N)

Expected load ϕ_l, residual capacity R_l and total capacity C_l for all links l in the network.

Set $\mathcal{S}$ of k candidate primary-backup route pairs (X, Y).

Output : Primary route X_N and backup route Y_N

For all links l in G do

$cost(l) = \frac{\phi_l}{R_l}$

$cost_{min} = \infty$; $X_N = Y_N = nil$

For each candidate route pair (X, Y) in set S do

If (D_N, ρ_N) cannot be satisfied along route X or route Y, then skip to next route.

Partition the end-to-end delay D_N among individual links of route X and Y.

Recompute the resulting residual capacities R'_l for each link $l \in X \cup Y$.

Recompute the link costs $cost(l) = \frac{\phi_l}{R'_l}$ for each link $l \in X \cup Y$.

Recompute the metric $cost(G) = \sum_{l \in G} \left(cost(l) - \frac{\phi_l}{C_l} \right)^2$.

If $cost(G) < cost_{min}$ then

$cost_{min} = cost(G)$

$X_N = X$ and $Y_N = Y$.

If $(cost_{min} > \alpha)$ then

Reject F_N since either resources are insufficient

or F_N makes the network critical over acceptable threshold.

else

Select route pair (X_N, Y_N) as the primary-backup route pair for F_N and reserve resources.

Figure 6.5: *PB-LCBR algorithm* selects both the primary and backup routes for a flow F_N between source s and destination d. F_N requires an end-to-end delay bound of D_N and average bandwidth of ρ_N^{avg}.

6.4.1 Earlier Approaches for Comparison

We compare LCBR against two of its variants that are based on earlier traffic engineering approaches in the context of bandwidth constrained route selection. Both the variants use the k-shortest paths framework of LCBR i.e., the route selection is performed by examining a set of candidate routes (or route pairs) among the k-shortest paths and one of the candidates is chosen based on an approach specific criteria. In LCBR, the candidate selection criteria is to minimize the global cost metric $cost(G)$.

WSP-BR: The first variant of LCBR is called the Widest Shortest Path based routing (WSP-BR). As the name suggests, it utilizes the load balancing criteria of Widest Shortest Path algorithm [47] and applies it in the context of delay-bandwidth constrained route selection. WSP-BR works as follows. Given k shortest path candidate routes (or route pairs), WSP-BR examines each candidate in the order of increasing length and selects the *feasible* minimum-length path that has the maximum residual bottleneck link capacity. In other words, among all the feasible paths with minimum length, the path having maximum residual bottleneck link capacity is selected. A *feasible* path here represents a path that can satisfy the end-to-end delay constraints of the flow being admitted. Note that WSP-BR does not take into account any information about ingress-egress pairs in the network; in fact an implicit assumption in WSP-BR is that all nodes are possible ingress and egress nodes. The primary route selection version, called P-WSP-BR, works as described above. The primary-backup route selection version, called PB-WSP-BR, selects the *feasible* route pair with minimum-length primary path that has maximum bottleneck link capacity. If multiple route pairs qualify as having minimum-length primary paths, then the same selection criteria is applied along backup paths of these route pairs with same primary path length..

MIRA-BR: The second variant of LCBR is called MIRA-BR which applies the concept of minimum interference routing proposed in [66] to the delay-bandwidth constrained route selection. MIRA applies the concept of deferred loading of critical network links. The importance of a link is defined as the number of source-destination pairs that are affected by using up resources along the link. Specifically, the weight of a link l in MIRA

is defined as

$$w(l) = \sum_{(s,d):l \in C_{sd}} \alpha_{sd}$$

Here C_{sd} is the set of links that belong to any mincut between the source-destination pair (s, d) and α_{sd} is the weight for ingress-egress pair (s, d). Following the evaluations in [66], we assume the same $\alpha_{sd} = 1$ for all source-destination pairs. Thus $w(l)$ represents the number of source-destination pairs whose mincuts include the link l. The intuition in MIRA behind defining $w(l)$ in this manner is to defer loading those links which are part of large number of mincuts, and thus affect a large number of source-destination pairs. MIRA-BR borrows the link weight from MIRA and utilizes it in the k shortest paths framework of LCBR. Specifically, given k shortest path candidate routes (or route pairs), MIRA-BR examines each candidate and selects the *feasible* candidate with smallest sum of link weights $w(l)$. The primary route selection version, called P-MIRA-BR, works as described above. The primary-backup route selection version, called PB-MIRA-BR, selects the *feasible* route pair that has the smallest sum of link weights across both the primary and backup paths of the route pair. For computing maxflows between different source-destination pairs, we use a publicly available implementation of the Push-Relabel Method [23].

In all the above algorithms, the intra-path delay partitioning is performed using the LSS algorithm described in Chapter 5 since it gives the best performance among all the delay partitioning techniques.

6.4.2 Evaluation Setup

We evaluate the performance of different route selection algorithms using two different network topologies. The first topology is a regular 5x5 Grid topology and the second topology is the North American IP backbone topology of Sprint [105] given in Figure 5.12. Other network parameters are similar to the ones in Chapter 5. In all the topologies, the link capacities are chosen as a mix from 45 Mbps to 200 Mbps. We assume that WFQ service discipline is employed at each link for servicing flow packets and the resource delay relationship at each link is given by Equation 3.7. Intra-path delay partitioning is performed using the framework of bufferless traffic multiplexing described

in Section 3.3.

Sources-destination pairs are selected uniformly among all nodes. The selected source-destination pairs are separated by a fixed shortest path distance to fairly compare the performance of three algorithms by eliminating path length variation as a source of performance difference. Two kinds of traffic profile information are applied between source-destination pairs - uniform and random. In the uniform profile case, each source-destination pair is expected to receive a total bandwidth demand $B(s,d) = 100$ Mbps whereas in the random profile case, the bandwidth demand is randomly between 45 Mbps and 100 Mbps. The profile information is used in computing link costs in LCBR algorithm. The profile information is also used in the LSS algorithm for delay partitioning by setting $\beta_l = 1/u_l$, where u_l is the expected utilization for link l described in Section 5.3.2. Additionally, flow requests are generated within the limits dictated by the traffic profile. Unless otherwise stated, all new flows request a QoS of 100Kbps average rate, end-to-end delay requirement of 50ms, and a burst size of 5Kbits. Also, the value of $k = 15$ candidate routes (or route pairs) are considered for each new flow, except in experiments where k itself is varied. We perform evaluations mainly for the 'static' case in which flow reservations that are provisioned once stay in the network forever. Flows are constantly admitted to the network till the network reaches a saturated state where no more flows can be admitted between any ingress-egress node pair. Algorithms for deterministic delay guarantees are evaluated using C++ implementations. Results for statistical delay guarantees are omitted since the memory requirements for statistical trace-driven simulations on general network topologies do not scale in our current system.

6.4.3 Effectiveness of LCBR Algorithm

We first provide a snapshot comparison of the performance of LCBR algorithm against the WSP-BR and MIRA-BR algorithms. Figure 6.6 plots the number of flows admitted by P-WSP-BR, P-MIRA-BR, and P-LCBR algorithms for primary route selection for 20 simulation runs, each run based on a different random number seed. Random number seed controls the combination of link capacities used in each topology; these are chosen in the range of 45 Mbps to 200 Mbps. Results are shown for the cases when uniform and random traffic profiles are applied between all sources and destinations. Figure 6.7

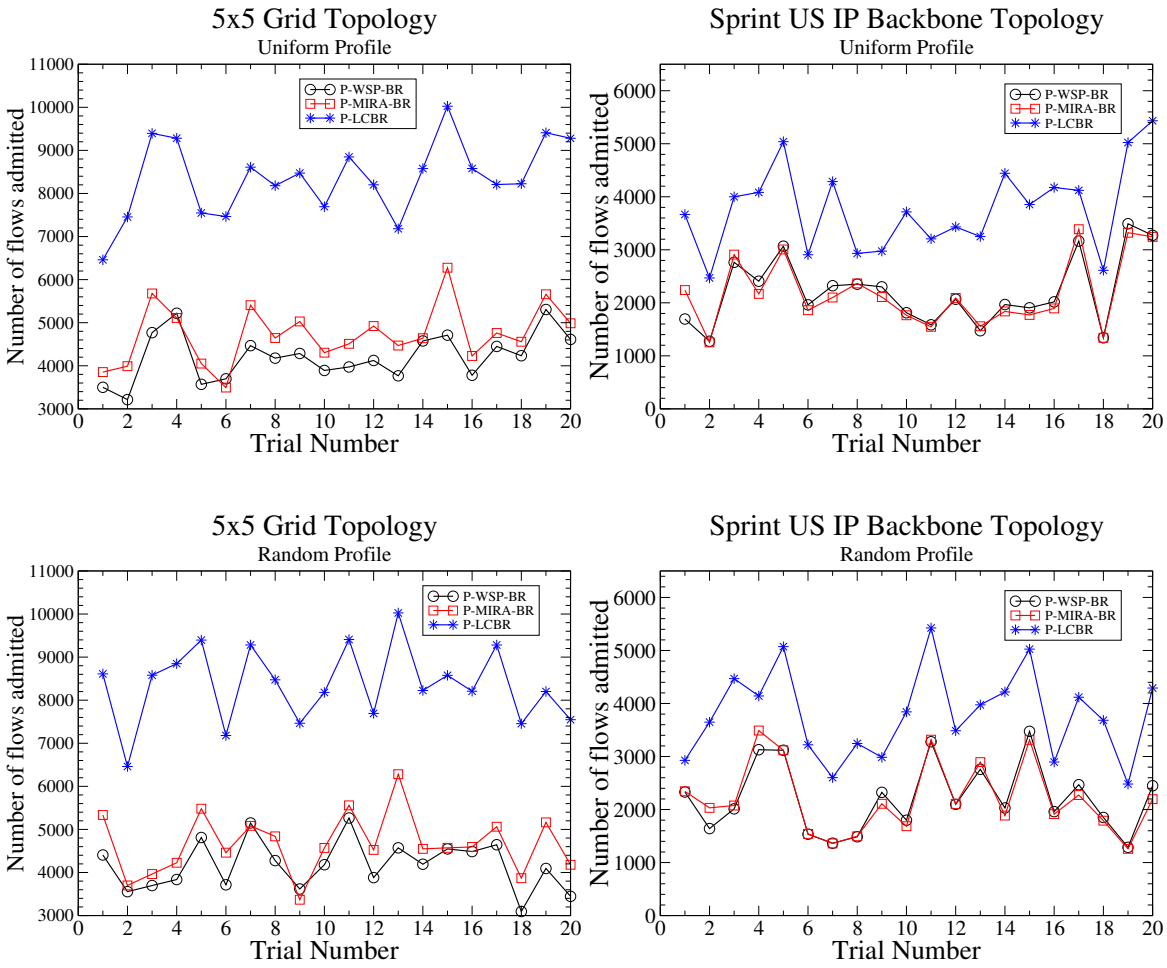

Figure 6.6: Number of flows admitted with various combinations of link capacities for primary route selection algorithms. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$, $k = 15$, min-hop distance$= 6$.

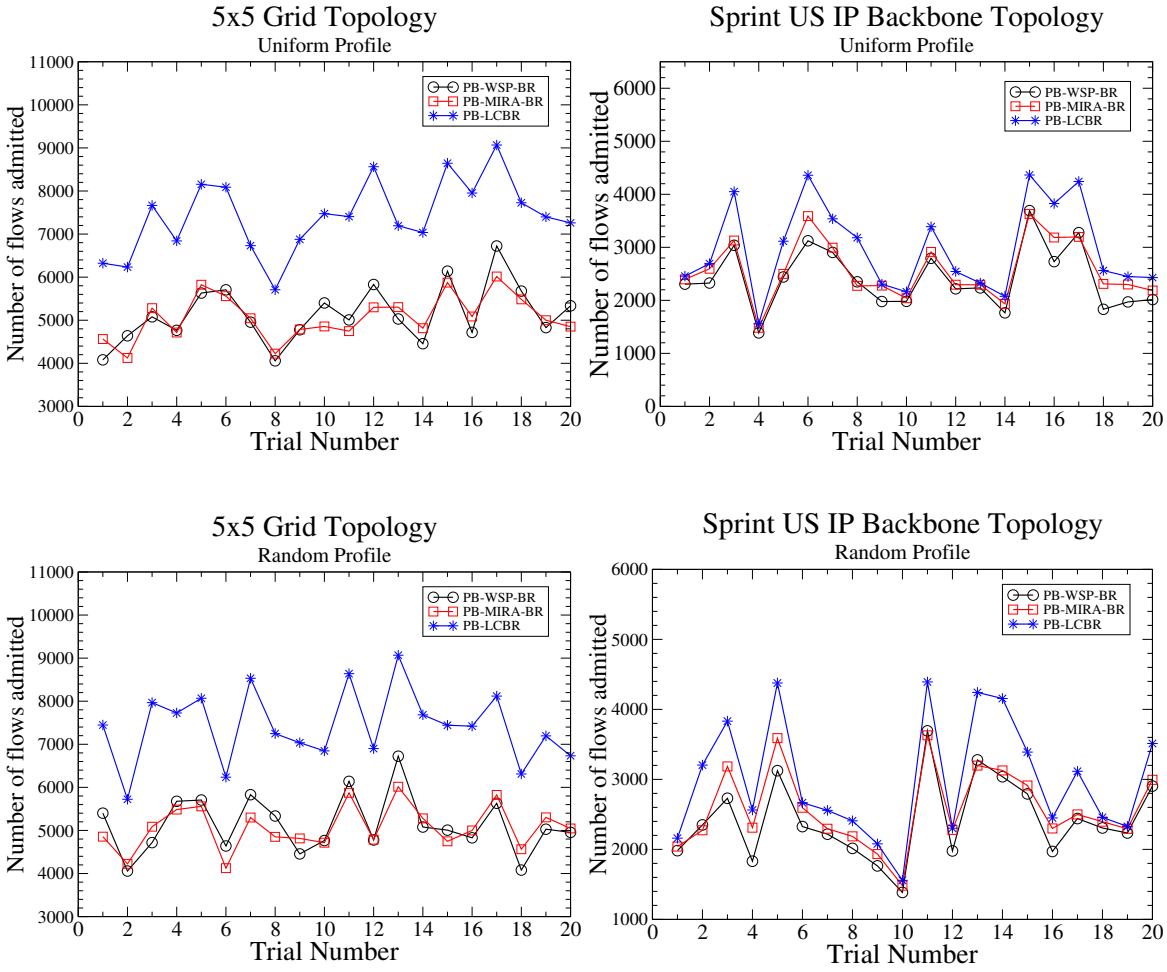

Figure 6.7: Number of flows admitted with various combinations of link capacities for primary-backup route selection algorithms. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$, $k = 15$, min-hop distance$= 6$.

plots the same for the case of primary-backup route selection algorithms (PB-* versions). The figures demonstrate that in all scenarios, LCBR consistently admits more flows than WSP-BR and MIRA-BR algorithms. LCBR performs better since it takes into account the ingress-egress profile information in computing per-link importance and bases route selection decisions on a network-wide load balancing criteria. On the other hand, WSP-BR neither considers the ingress-egress-profile information, nor does it perform network-wide load balancing. The only limited form of load balancing performed by WSP-BR is the selection of widest-bandwidth shortest route among all the available candidate routes. The performance of MIRA-BR is better than WSP-BR on Grid topology for primary route selection and marginally worse for primary-backup route selection. In this experiment, the performance of MIRA-BR and WSP-BR are in general close to each other. MIRA-BR takes into account the ingress-egress information in determining link criticality. In situations where MIRA-BR performs worse, the links along the mincut between ingress-egress pairs do not faithfully represent the most critical links in the network, thus defeating the link criticality metric adopted by MIRA.

6.4.4 Load Balance

In this section, we show that network-wide load balance, as defined by the cost metric $cost(G)$ in Equation 6.4, is indeed maintained better by the LCBR algorithm in comparison to WSP-BR and MIRA-BR. Figure 6.8 plots the value of network cost metric $cost(G)$ for the three algorithms with each successive flow request for the case of primary route selection algorithms. Figure 6.9 plots the same for the case of primary-backup route selection algorithms. Whenever a flow is admitted, $cost(G)$ increases; when a flow is rejected, $cost(G)$ stays constant. Recall that $cost(G)$ represents the vector distance of the current link costs from the idle-state link costs in the network. Thus a smaller value of $cost(G)$ indicates a better load balance across the network. The figures show that, with successive flow requests, LCBR algorithm maintains a lower value of $cost(G)$ compared to WSP-BR and MIRA-BR algorithms. While the $cost(G)$ values of WSP-BR and MIRA-BR saturate after admitting a smaller number of flows, LCBR is able to continue admitting flows for a longer duration.

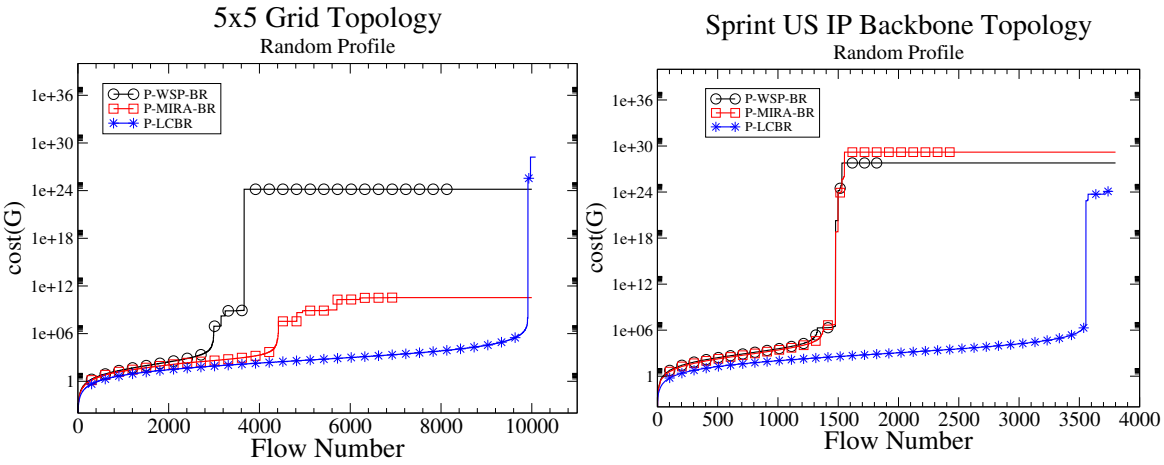

Figure 6.8: Load balance for primary route selection. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$, $k = 15$, min-hop distance$= 6$.

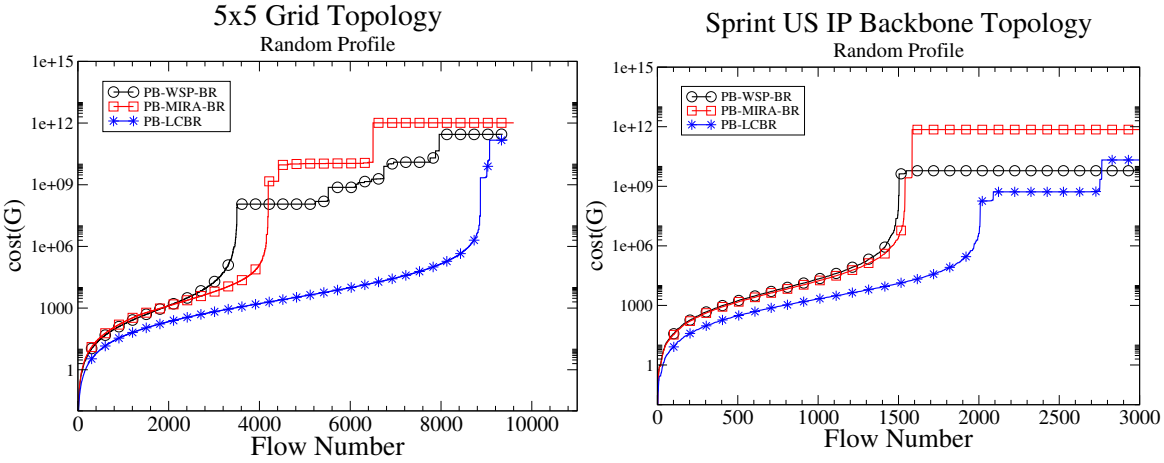

Figure 6.9: Load balance for primary-backup route selection. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$, $k = 15$, min-hop distance$= 6$.

6.4.5 Effect of Candidate Set Size

An important factor that determines the performance of different algorithms is the size k of the candidate set S of routes (or route pairs) that are examined during route selection. Figure 6.10 plots the effect of varying the candidate set size k on the number of flows admitted by the three primary route selection algorithms. Figure 6.11 plots the same for the case of primary-backup route selection algorithms. With increasing set size k, the number of admitted flows increases initially for all the three algorithms. In general, larger value of k implies more candidate choices for selecting a route leading to more admitted flows. The number of admitted routes tends to saturate with increasing k since routes with longer lengths tend to be part of the candidate set. Since longer paths tend to consume more network resources, these additional candidate routes are rarely selected and do not contribute to better network resource usage.

There are also noticeable differences in the behavior of the three algorithms with increasing k. For all candidate set sizes, LCBR algorithm admits significantly higher number of flows compared to WSP-BR and MIRA-BR algorithms, since it always selects the candidate route which best maintains the network load balance. We also observe that LCBR produces the maximum improvement in number of admitted flows with initial increase in k since it has more choice of candidates whose $cost(G)$ values can be compared for level of load balance. The WSP-BR algorithm is least affected by increase in k since it always selects the shortest feasible route with widest bottleneck capacity and hence does not look at the entire candidate set during the selection process. Thus increasing k helps WSP-BR only if it results in inclusion of more feasible candidates with shortest path length. MIRA-BR has the least consistent behavior with increasing k. Specifically, it has a higher tendency to select longer paths as these may contain fewer links that impact the maxflows between different source-destination pairs. Since longer routes tend to consume more network resources, the number of admitted flows with MIRA-BR might sometimes actually drop with increasing k, as observed in the case of primary-backup route selection. For the case of primary route selection, however, MIRA-BR clearly produces better results than WSP-BR with initial increase in k.

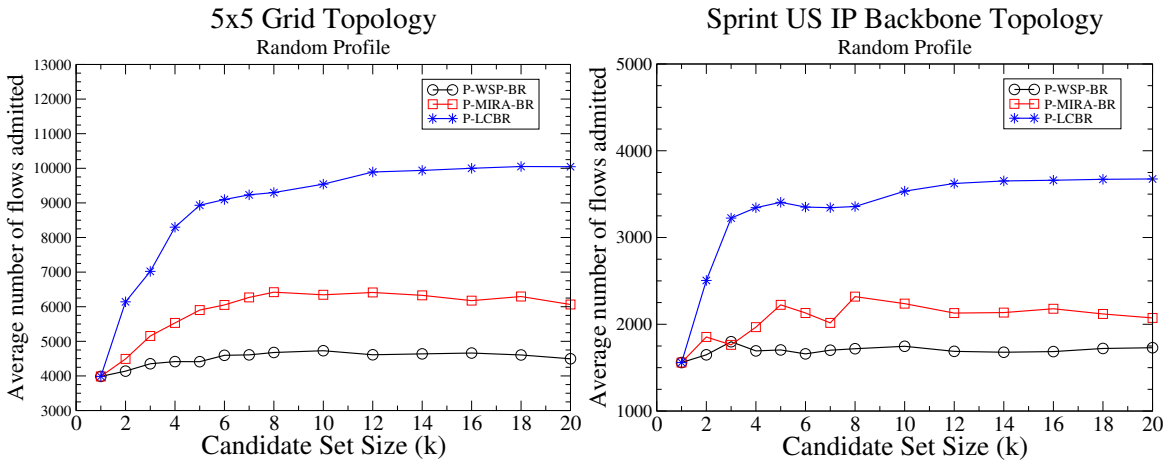

Figure 6.10: Effect of candidate set size k for primary route selection. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$, min-hop distance$= 6$.

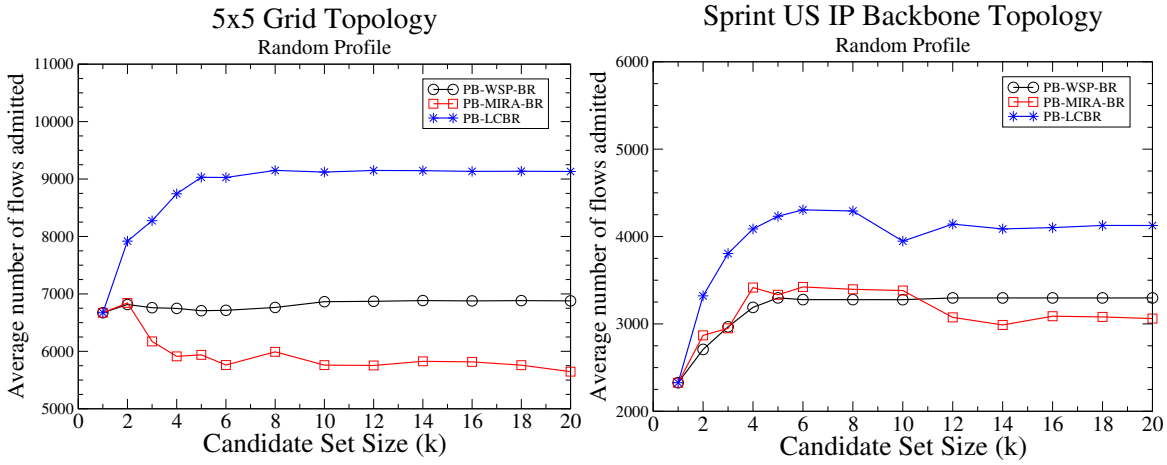

Figure 6.11: Effect of candidate set size k for primary-backup route selection. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$, min-hop distance$= 6$.

6.4.6 Effect of Min-Hop Distance

In all experiments, we select those ingress-egress pairs that are separated by a fixed min-hop distance. In this section, we compare the performance of LCBR against WSP-BR and MIRA-BR as this minimum-hop distance between the source-destination nodes is varied. Figures 6.12 and 6.13 plot this variation for primary route selection and primary-backup route selection algorithms respectively. As expected, the number of admitted flows decreases in all the three algorithms as the min-hop distance is increased. This is due to two reasons. First, there is a decrease in the number of source-destination pairs in the network among which flows can be set up as the min-hop distance increases. Secondly, the length of candidate paths between the source-destination pairs increases with increasing min-hop distance. This in turn implies that the selected routes, which are longer, use more network resources compared to shorter min-hop distance case. As a result, fewer flows are admitted with increasing min-hop distance.

We also observe that LCBR consistently performs better than both WSP-BR and MIRA-BR. With increasing min-hop distance, there is a small increase in factor improvement due to LCBR over WSP-BR and MIRA-BR in the case of primary-route selection; on the other hand there is a small decrease in factor improvement in the case primary-backup route selection. The former increase is due to the fact longer paths consume more resources and hence it becomes more important to maintain load balance with increasing path lengths; LCBR does a better job of maintaining network load balance than WSP-BR and MIRA-BR with increasing path lengths. In the case of primary-backup route selection, increasing path lengths imply that there exist fewer choices of short backup routes corresponding to each primary route. As a result, there is a decrease in the number of "good-quality" route pairs to choose from in the candidate set and the performance of both LCBR and MIRA-BR approaches that of WSP-BR.

6.4.7 Effect of End-to-end Delay

This section examines the performance of LCBR against WSP-BR and MIRA-BR as the the end-to-end delay requirement of flows is varied. Figures 6.14 and 6.15 show that with increasing end-to-end delay, number of flows admitted increases initially and then tends to saturate for all the algorithms. Initially, each flow's delay-derived bandwidth

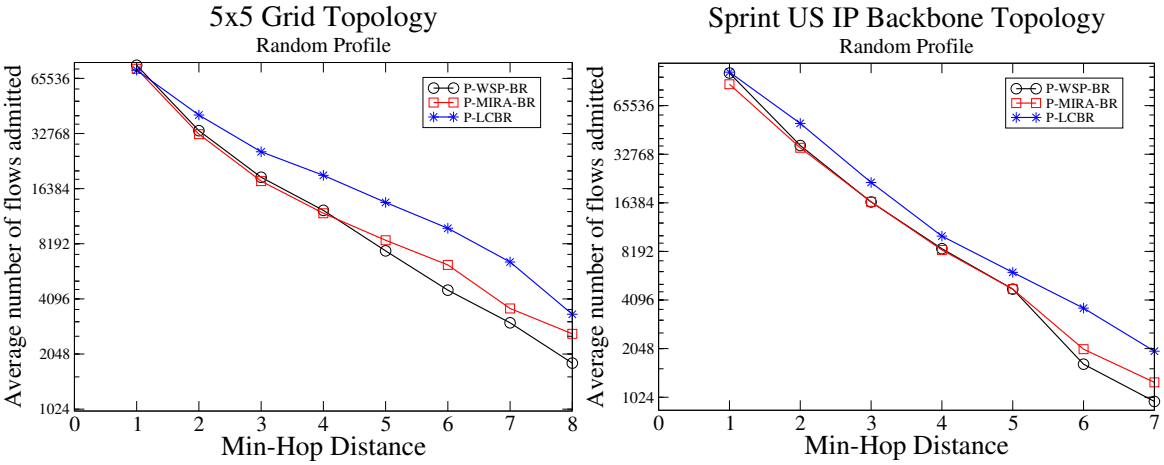

Figure 6.12: Effect of min-hop distance on primary route selection. Y-axis is in log scale. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$, $k = 15$.

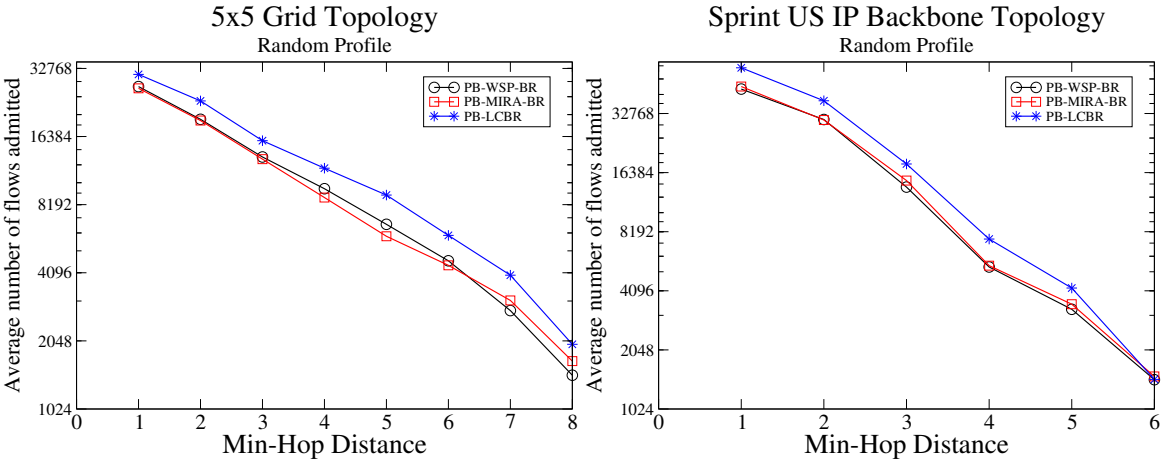

Figure 6.13: Effect of min-hop distance on primary-backup route selection. Y-axis is in log scale. $\rho_i^{avg} = 100Kbps$, $D_i = 60ms$, $\sigma_i = 5\ Kbits$, $k = 15$.

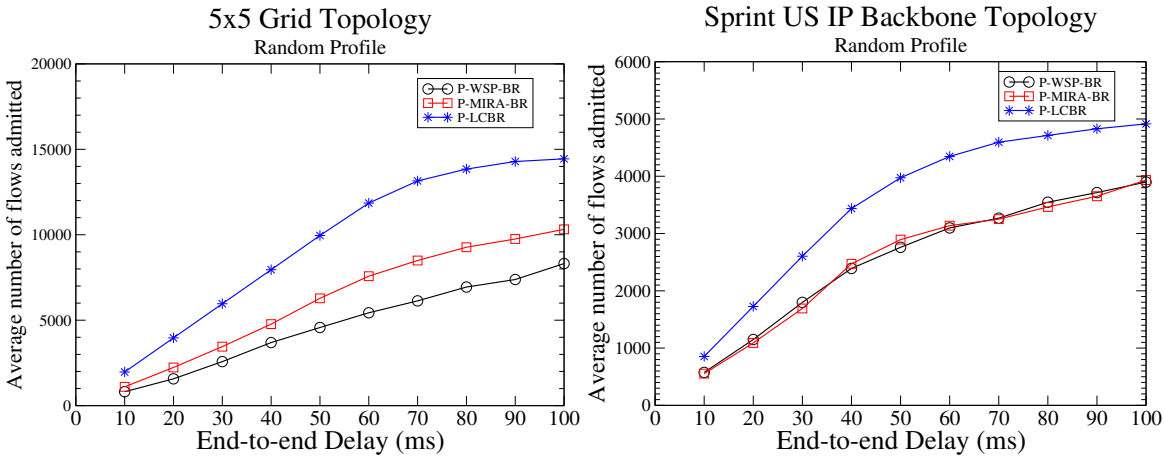

Figure 6.14: Effect of end-to-end delay on primary route selection. $\rho_i^{avg} = 100Kbps$, $\sigma_i = 5\ Kbits$, $k = 15$, min-hop distance$= 6$.

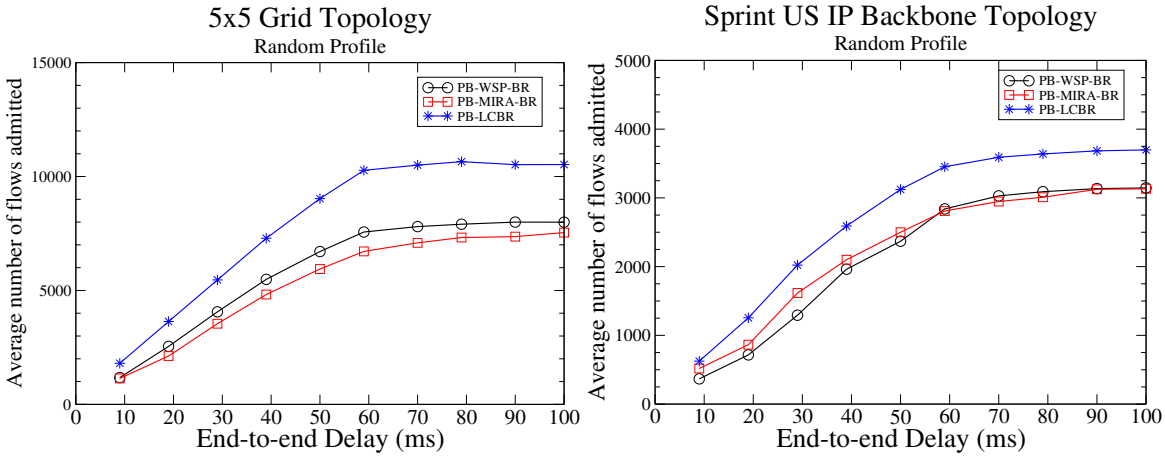

Figure 6.15: Effect of end-to-end delay on primary-backup route selection. $\rho_i^{avg} = 100Kbps$, $\sigma_i = 5\ Kbits$, $k = 15$, min-hop distance$= 6$.

requirement is tighter than the average rate requirement and hence delay is the tighter constraints. For larger delay requirement, the delay-derived bandwidth becomes smaller than the average rate requirement. In the saturation region, flows are essentially being serviced at their average rates. We also observe that the LCBR consistently admits larger number of flows with an almost constant factor improvement over WSP-BR and MIRA-BR over all delay ranges. MIRA-BR performs better than WSP-BR in the Grid topology for primary route selection and worse for primary-backup route selection. The latter effect is due to selection of route pairs with longer backup paths by MIRA-BR. Both have similar performance in the Sprint topology.

6.4.8 Effect of Intra-Path QoS Constraint Partitioning

In earlier experiments, the LSS algorithm described in Section 5.3.2 was used as the intra-path QoS constraint partitioning scheme. In this section, we examine the impact of different partitioning algorithms on LCBR. Figures 6.16 and 6.17 plot the number of admitted flows in different simulation instances when four different partitioning schemes ESS, PSS, OPQ, and LSS are used. As observed in Chapter 5, again LSS consistently admits more number of flows than the other algorithms. However, note that the magnitude of difference between various algorithms is marginally smaller since the number of candidate routes (or route-pairs) examined by LCBR is 15, whereas we considred only a single shortest path between every source and destination in Chapter 5. With larger number of candidate routes, the performance is mainly determined by the route-selection process, with finer level of improvement obtained using the intra-path load-balancing algorithm.

6.4.9 Impact of Early Rejection

In the algorithms P-LCBR and PB-LCBR, we introduced an early rejection factor (α), that can be used to reject an admissible flow if the increment in network cost is too large. In all experiments thus far, we assumed that $\alpha = \infty$, implying that a feasible flow is admitted irrespective of the corresponding increase in network cost. Figures 6.18 and 6.19 show impact of varying α on the performance of P-LCBR and PB-LCBR respectively. For the case of primary path selection on the Grid topology, the number of flows admitted

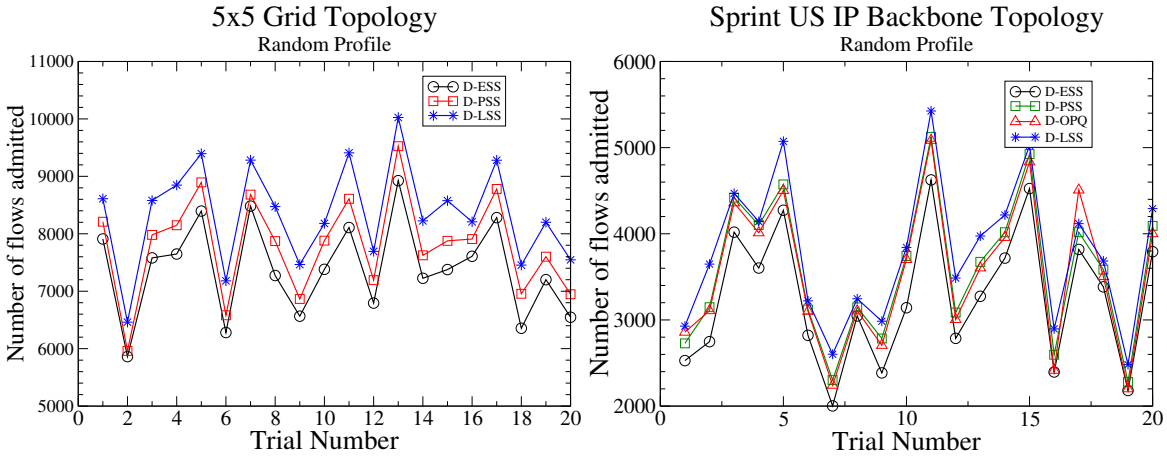

Figure 6.16: Effect of intra-path QoS constraint partitioning on primary route selection. $\rho_i^{avg} = 100Kbps$, $\sigma_i = 5\ Kbits$, $k = 15$, min-hop distance$= 6$.

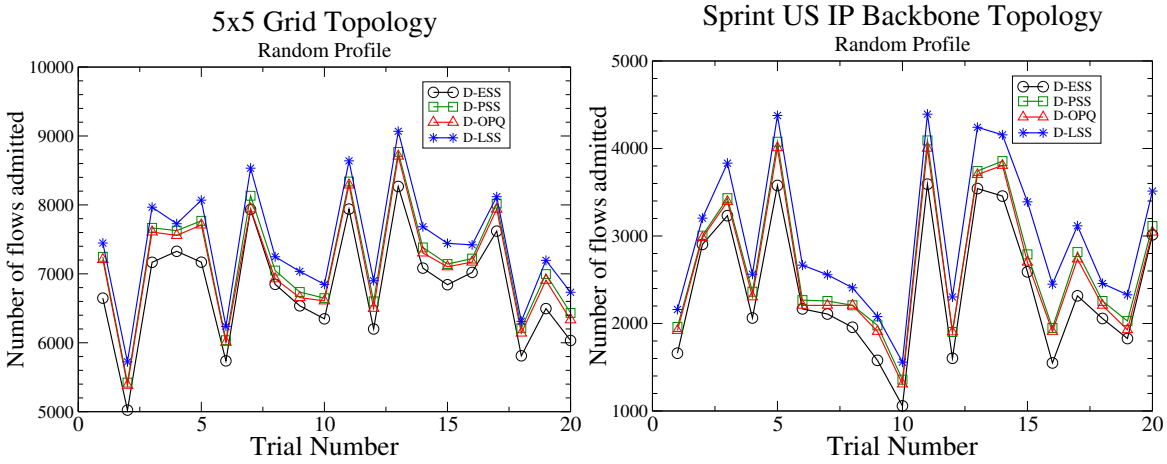

Figure 6.17: Effect of intra-path QoS constraint partitioning on primary-backup route selection. $\rho_i^{avg} = 100Kbps$, $\sigma_i = 5\ Kbits$, $k = 15$, min-hop distance$= 6$.

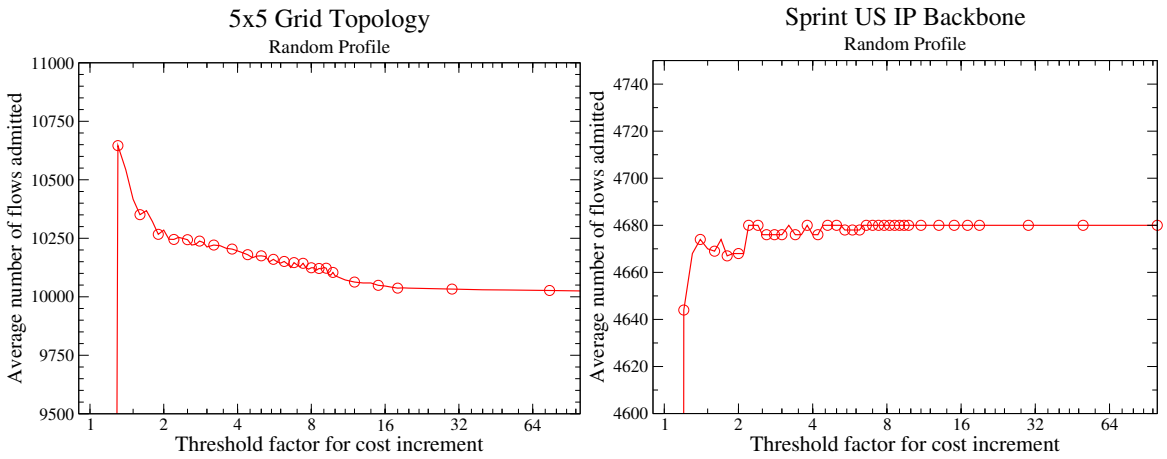

Figure 6.18: Effect of early rejection on primary route selection. X-axis is in log-scale. $\rho_i^{avg} = 100Kbps$, $\sigma_i = 5\ Kbits$, $k = 15$, min-hop distance$= 6$.

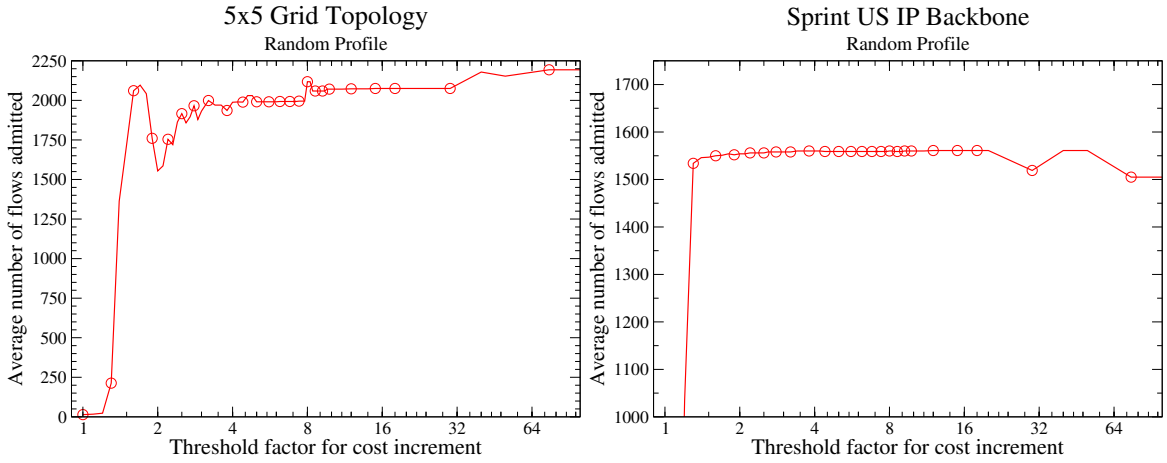

Figure 6.19: Effect of early rejection on primary-backup route selection. X-axis is in log-scale. $\rho_i^{avg} = 100Kbps$, $\sigma_i = 5\ Kbits$, $k = 15$, min-hop distance$= 6$.

increases in the range of $\alpha = 1$ to 3, and then decreases steadily. This is because, with α in the range of 1 to 3, P-LCBR is able to reject flows that might consume large amount of critical resources and is consequently able to admit to admit more flows in the long term. For primary-backup route selection on Grid topology, however, number of admitted flows marginally increases initially and stays steady thereafter. This effect is due to the fact that the number of "good-quality" primary-backup pairs reduces as we move away from primary route selection alone. Thus, any early rejections turn out to be counter-productive in the long run. Similar trend is observed in the cases of both primary and primary-backup route selection with the Sprint topology, which has lower degree of connnectivity compared to the mesh topology. Thus a carefully tuned value of α can help improve network resource utilization for primary route selection process.

6.4.10 Computation Cost

There are two principal components of computation cost – offline pre-computation cost for calculating expected load and online computation cost for finding individual route-pairs. Figures 6.20 and 6.21 plot the pre-computation cost whereas Figures 6.22 and 6.23 plot the online route selection cost. For both primary and primary-backup route selection, the computation time grows almost linearly with candidate set size for both offline and online phases. However, the computation cost is relatively higher for primary-backup route selection due to the fact that route selection and intra-path QoS partitioning need to be performed for both primary and backup routes. The main component of computation cost in offline phase is the process of finding k-shortest candidate routes (route-pairs). For online phase, the main computationally expensive component is the intra-path QoS constraint partitioning that needs to be performed for each candidate route.

6.5 Summary of Contributions

In this chapter, we have addressed the problem of selecting primary and backup routes for network flows that require end-to-end delay and bandwidth guarantees. We made the following research contributions.

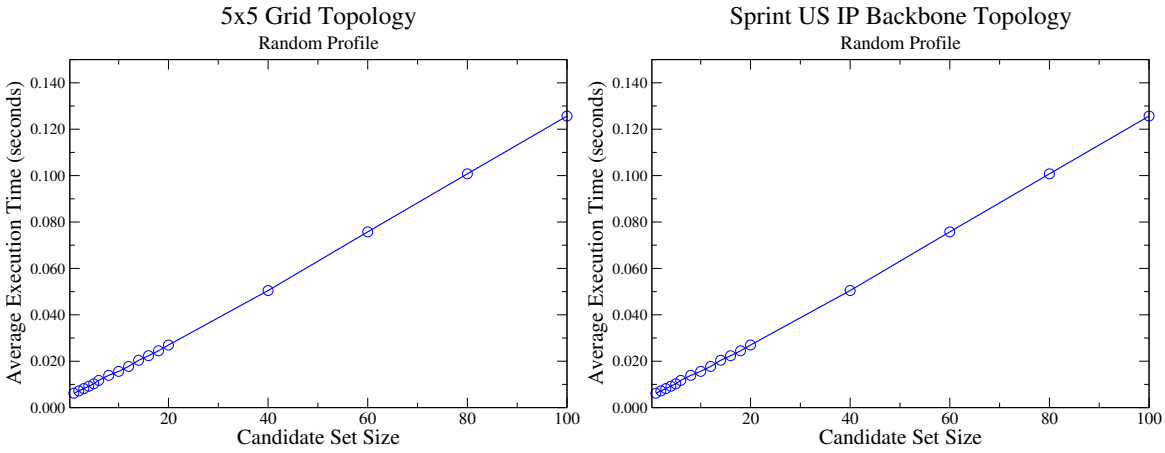

Figure 6.20: Average pre-computation cost vs. candidate set size for primary route selection. $\rho_i^{avg} = 100Kbps$, $\sigma_i = 5\ Kbits$, min-hop distance$= 6$.

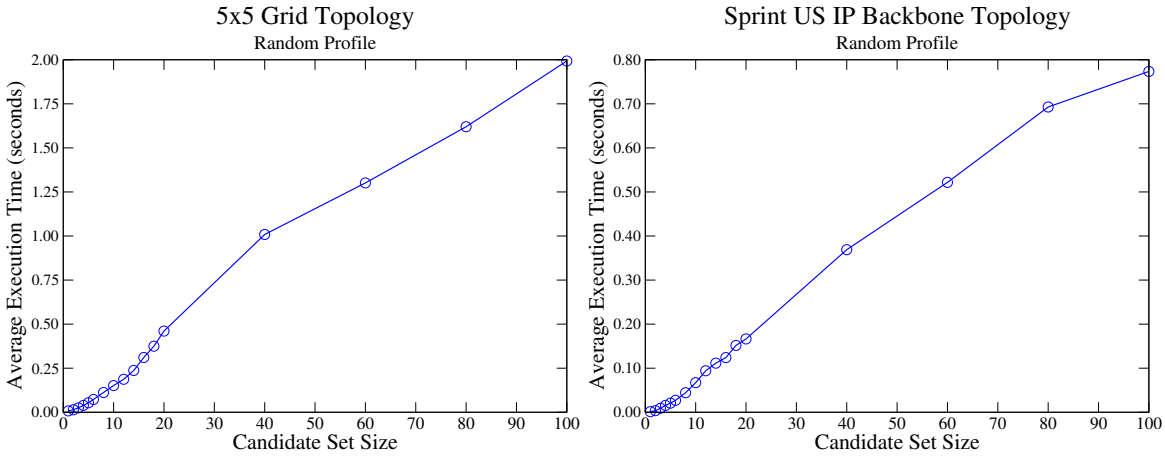

Figure 6.21: Average pre-computation cost vs. candidate set size for primary-backup route selection. $\rho_i^{avg} = 100Kbps$, $\sigma_i = 5\ Kbits$, min-hop distance$= 6$.

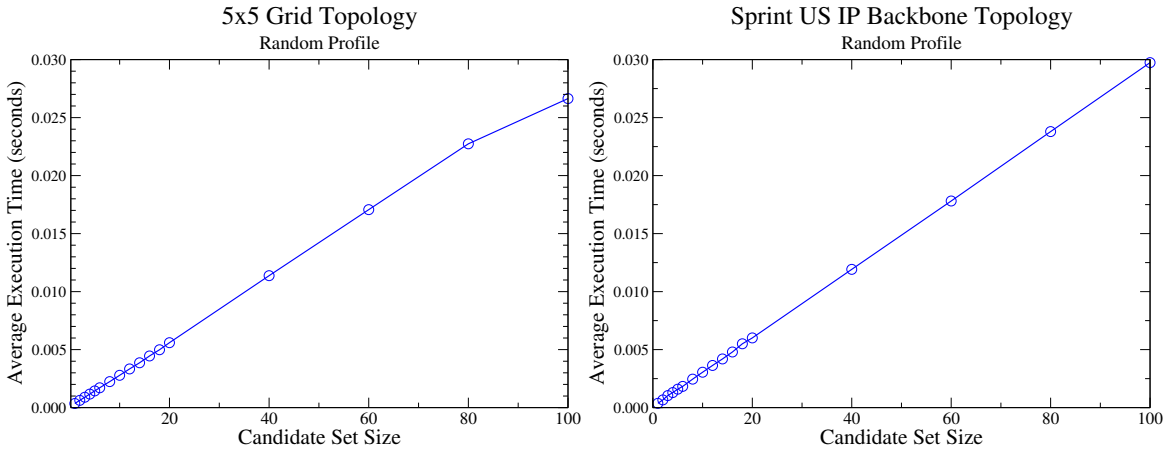

Figure 6.22: Average online computation cost vs. candidate set size for primary route selection. $\rho_i^{avg} = 100Kbps$, $\sigma_i = 5\ Kbits$, min-hop distance$= 6$.

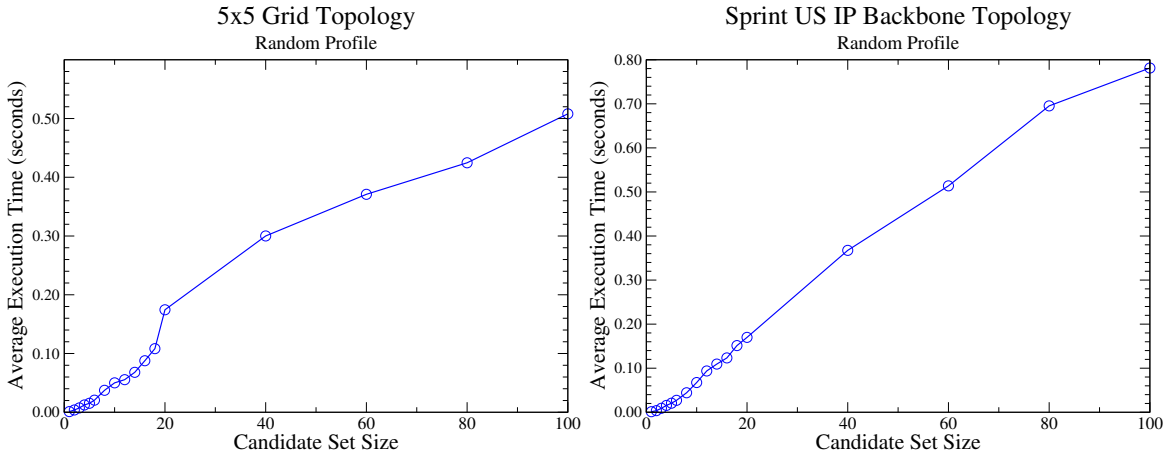

Figure 6.23: Average online computation cost vs. candidate set size for primary-backup route selection. $\rho_i^{avg} = 100Kbps$, $\sigma_i = 5\ Kbits$, min-hop distance$= 6$.

- We proposed a new route selection algorithm for real-time flows, called Link Criticality Based Routing (LCBR), that can satisfy long-term bandwidth requirements, deterministic/statistical end-to-end delay requirements, and protection against single network element failures. The basic idea behind LCBR is to select routes that maintain network-wide load balance as new flows are admitted. The cost metric for network-wide load balance uses a simple notion of link criticality that is based on measured traffic profile and ingress-egress node information.

- We introduced a general and flexible framework for delay-bandwidth constrained route selection that allows us to examine several candidates from a set of k shortest-length routes (or route pairs), and select one that best satisfies a specific optimization objective. This framework can accommodate a range of different algorithms such as the P-LCBR algorithm for primary route selection, PB-LCBR algorithm for primary-backup route selection, and algorithms based on other optimization objectives such as WSP-BR and MIRA-BR for primary as well as backup route selection.

- Finally we demonstrated how the above framework can be used to incorporate inter-level dependencies between intra-path delay partitioning and inter-path load balancing.

Through a detailed evaluation, we have shown that the LCBR algorithm can indeed provide better network resource usage efficiency in comparison to earlier traffic engineering based approaches. The improvements observed are up to 1.9 times over a 5x5 Grid topology and up to 2.2 times over Sprint's IP backbone topology for deterministic delay guarantees. LCBR achieves higher resource usage efficiency because it explicitly maintains network-wide load balance and ensures that none of the critical network links exhausted early in the process of admitting new flows. The link criticality is based on three pieces of information – the current residual link capacity, knowledge of ingress-egress node pairs, and expected traffic profile information. The combination of the three pieces of information coupled with a carefully designed network load balancing criteria enables LCBR to outperform other approaches that are based purely on residual bandwidth information and/or ingress-egress information. By maintaining network-wide load balance, LCBR prevents the formation of bottlenecks at critical network links leading to

more flows being admitted in the long term.

Chapter 7

Conclusions

This chapter concludes the dissertation with a summary of major research contributions and outlines future directions of research.

7.1 Summary of Dissertation

The central subject of this dissertation is efficient provisioning of network resources for network resource virtualization with QoS guarantees. In recent years, network service providers (NSP) have witnessed a surge in the demand from their large organizational customers for dedicated wide-area network connectivity with flexibility to exercise fine-grained control over their virtual network resources. This virtualization of network is being coupled with a rapid growth in the amount of real-time network traffic that the NSP's network infrastructure is required to carry. An additional force that influences resource allocation decisions is the inherent desire of NSPs to maximize their revenue base by accommodating the requirements of as many customers as possible. These three competing forces have created an urgent need for network resource provisioning techniques that maximize the resource utilization efficiency of the network infrastructure.

This dissertation presents a comprehensive set of resource provisioning techniques to satisfy the competing requirements on the QoS-constrained network resource virtualization process. Our techniques rely on two guiding themes in order to maximize network resource usage efficiency. First theme is that load balancing leads to higher resource us-

age efficiency. By ensuring that loads on different links of the network are as balanced as possible, provisioning techniques can prevent critical network links from being exhausted early and becoming a bottleneck for the entire network. The second theme is that statistical QoS guarantees help to achieve higher network resource usage efficiency. Provisioning techniques can reduce the resource requirements for each network flow by exploiting the inherent statistical multiplexing effect in network traffic, as also network applications' tolerance to certain level of delay violations. In this dissertation, we have proposed a set of interdependent resource provisioning techniques for network resource virtualization at the following three levels of the network.

7.1.1 Link-level Statistical Delay Guarantees

At link level, we have proposed a measurement-based technique, called *Delay Distribution Measurement* (DDM) based admission control, that allocates link bandwidth for traffic flows having distinct statistical delay requirements (i.e. both delay bound and a tolerance bound to delay violations). We have developed a quantitative framework that exploits the well known fact that the actual delay experienced by most packets is far smaller than their worst-case delay bound requirement. DDM exploits statistical multiplexing by dynamically measuring the distribution of the ratio of actual packet delay and worst-case delay bound. The measured distribution is then used to reduce the bandwidth requirements of newly admitted flows. DDM is the first approach that applies the concept of measurement-based admission control to provide distinct per-flow statistical delay guarantees. Compared to deterministic admission control, DDM can potentially increase the link utilization with VoIP traffic by a factor of 3 when tolerance to delay violations is as low as 10^{-5}.

7.1.2 Intra-path QoS Constraint Partitioning

At path-level, we have developed a new algorithm called *Load-based Slack Sharing* (LSS), that partitions the end-to-end statistical delay requirements of a flow into delay requirements at each link along a multi-hop path. LSS works in the context of both unicast and multicast flows having deterministic or statistical delay requirements. LSS applies

the concept of intra-path load balancing to assign larger share of end-to-end delay to the more loaded links along the path, thus ensuring that network load balance is maintained in the long term. A novel feature of LSS is that it handles multiple QoS requirements such as end-to-end delay bounds and delay violation probability bounds, in contrast to earlier approaches that handle only a single QoS requirement.

Two other contributions not specifically tied to the LSS algorithm are as follows. Firstly, we developed the notion of partitioning *slack* in end-to-end QoS requirement, rather than directly partitioning the end-to-end QoS as in earlier approaches. Slack quantifies the amount of flexibility available in balancing the loads across links of a multi-hop path. Secondly, we developed the detailed admission control algorithms for unicast and multicast flows that can be used in conjunction with any of the earlier QoS constraint partitioning schemes. Compared to earlier approaches, LSS improves the number of admitted flows by up to a factor of 1.2 for deterministic delay bounds and up to 2.8 for statistical delay bounds.

7.1.3 Network-level Route Selection

At the network level, we have addressed the end-to-end QoS and fault-tolerance requirements in the network resource virtualization process. The goal is to select primary and backup routes for virtual network connections that have end-to-end delay, long-term bandwidth, and fault-tolerance requirements. We have developed a new algorithm, called Link Criticality Based Routing (LCBR), to select primary and backup routes for real-time aggregate flows carried by virtual connections such that the network resource utilization efficiency is maximized. Like the intra-path LSS algorithm, LCBR maintains network-wide load balance while selecting routes by using a simple notion of link criticality that is based on measured traffic profile and ingress-egress node pair information. It also accounts for inter-level dependencies between path-level and network-level load balancing. LCBR provides a general framework for delay-bandwidth constrained route selection that allows examination of several candidates from a set of k shortest-length routes and select one that best satisfies a specific optimization objective. The framework accommodates a range of different algorithms such as primary route selection and primary-backup route selection using LCBR's load balancing criterion, as well as route selection criteria from

earlier approaches such as MIRA and WSP. By means of evaluation over two general network topologies, we showed that LCBR can improve the number of admitted flows by a factor of 2 compared to the delay-constrained variants of WSP and MIRA.

7.2 Future Research Directions

Several short-term and long-term research directions emerge from the issues addressed in this dissertation. This section summarizes some of these directions.

7.2.1 Statistical Bandwidth Guarantees

In Chapter 4, we have addressed the problem of providing per-flow statistical delay guarantees. However the bandwidth guarantee provided to each flow is deterministic in nature. A number of earlier works, such as [88, 60, 38, 42, 27, 14], have addressed the problem of exploiting statistical multiplexing along the bandwidth dimension. Simultaneous statistical guarantees along both bandwidth and delay dimensions, however, introduces some new challenges. In particular, we now need to design a resource correlation function that maps the statistical delay as well as statistical average bandwidth guarantees for flows into meaningful service bandwidth at the link. Techniques to derive such correlation functions using measurement-based approach needs to be investigated.

7.2.2 Delay Partitioning for Many-to-many Multicast Connections

We addressed the problem of partitioning end-to-end delay guarantees for unicast and single-source multicast flows in Chapter 5. A natural extension of this work is for the case of many-to-many multicast connections in which end-to-end delay guarantees must be provided between every pair of multicast members. A common context in which this problem arises is that of IP telephony, where multiple parties simultaneously communicate with each other in real-time. The problem is more complicated than the single-source multicast problem since the tree no longer has a clear root. Cost-function based solution to this problem was proposed in [73]. We plan to investigate extensions of the LSS framework to many-to-many multicast connections with the objective of maintaining load

balance in the network.

7.2.3 Pre-computation Cost in Route Selection

The k-shortest paths framework for primary-backup route selection, which is used by LCBR algorithm and its variants, pre-computes a candidate set of k disjoint route pairs between all possible sources and destinations in an offline pre-computation phase. While this simplifies the online process of comparing relative costs of different primary-backup candidates, it is still an inelegant brute-force technique that could potentially be optimized. Efficient solutions exist [108, 109, 98] for the problem of finding the minimum-cost disjoint route-pair (MinCostPP), where the cost metric is simply the sum of weights for each link in the disjoint route-pair. LCBR considers the k-shortest disjoint route-pairs as candidates for primary and backup routes and selects the pair which minimizes $cost(G)$, as opposed to the MinCostPP formulation which yields only a single disjoint route-pair with minimum cost. For this reason we cannot directly take advantage of solutions to the MinCostPP problem. One of the open research issues from Chapter 6 is to investigate lower complexity algorithms to identify the k disjoint route-pairs that constitute the candidate set.

7.2.4 Scalability of Backup Resource Aggregation

Another open issue in the primary-backup route selection problem in Chapter 6 concerns the scalability of maintaining multiple backup-sets at each link as the network grows in size. In the worst case, every link would end up keeping track of $m+n$ backup sets, where m is the number of links and n is the number of nodes. In order to make the algorithm more scalable, we plan to investigate techniques to combine the $m + n$ backup sets into smaller number of backup sets. A simple approach is to combine all backup sets into one large backup set. In this case, the failure of any one network element would lead to all the backup reservations being activated, which in turn implies that smaller number of virtual connections would be admitted in the long-term. Another approach is to combine only those backup sets whose network elements are likely to fail at the same time. Here we need some efficient way to identify such elements that can be combined. Any combination

of backup sets leads to reduced efficiency of resource allocation but appears necessary for improving scalability.

7.2.5 Inter-Domain Resource Provisioning with QoS Constraints

The techniques proposed in this dissertation are applicable mainly in the context of intra-domain network, where an NSP owns an entire physical network infrastructure and has complete administrative control over its resources. This suffices for many practical scenarios, such as branch locations of a large organization interconnected via VPN or VON and serviced by the same NSP. It is likely, however, that network flows with QoS guarantees may straddle across the administrative boundaries of an NSP's network. In such cases, the NSP may have access only to aggregate information about neighboring domains. Though resource allocation decisions may no longer be as fine grained, it is still important to maximize the collective network resource utilization efficiency. Hence further development of ideas presented in this dissertation in the inter-domain context would be of considerable practical significance.

Bibliography

[1] Intermediate system to intermediate system intra-domain routeing exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (iso 8473). International Standard 10589:2002, Second Edition.

[2] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas. LDP specification. RFC 3036, January 2001.

[3] M. Andrews. Probabilistic end-to-end delay bounds for earliest deadline first scheduling. In *Proc. of IEEE INFOCOM 2000*, March 2000.

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

[5] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of internet traffic engineering. RFC 3272, May 2002.

[6] D. Awduche, J. Malcom, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over mpls. RFC 2702, Sept. 1999.

[7] D.O. Awduche, L. Berger, D. Gan, T. Li, G. Swallow, and V. Srinivasan. *Extensions to RSVP for LSP Tunnels*. Internet Draft, Sept. 1999.

[8] D.O. Awduche, D. Gan, T. Li, G. Swallow, and V. Srinivasan. *Extension to RSVP for Traffic Engineering*. Internet Draft, Aug. 1998.

[9] A. Banerjea. Simulation study of the capacity effects of dispersity routing for fault-tolerant real-time channels. In *Proc. of ACM SIGCOMM'96*, volume 26(4), pages 194–205, Oct. 1995.

[10] A. Banerjea, C. Parris, and D. Ferrari. Recovering guaranteed performance service connections from single and multiple faults. In *Proc. of IEEE GLOBECOM'94, San Francisco, CA*, pages 162–168, Nov. 1994.

[11] J.C.R. Bennett and H. Zhang. WF^2Q: Worst-case fair weighted fair queuing. In *Proc. of IEEE INFOCOM 1996*, pages 120–128, March 1996.

[12] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, K. Nichols, and M. Speer. *A Framework for Use of RSVP with DiffServ Networks*. Internet Draft, Nov. 1998.

[13] S. Blake, D. Black, D. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, Dec. 1998.

[14] R. Boorstyn, A. Burchard, J. Leibeherr, and C. Oottamakorn. Statistical service assurances for traffic scheduling algorithms. *IEEE Journal on Selected Areas in Communications*, 18(13):2651–2664, December 2000.

[15] J. Boyle, V. Gill, A. Hannan, D. Cooper, D.Awduche, B. Christian, and W.S. Lai. Applicability statement for traffic engineering with MPLS. RFC 3346, August 2002.

[16] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: An overview. RFC 1633, June 1994.

[17] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP). RFC 2205, September 1997.

[18] L. Breslau, S. Jamin, and S. Shenker. Comments on performance of measurement-based admission control algorithms. In *Proc. of IEEE INFOCOM 2000*, March 2000.

[19] R. Callon and N. Feldman. A framework for multiprotocol label switching. Internel Draft, June 1999.

[20] B.A. Chambers. The Grid Roofnet: A rooftop ad hoc wireless network. Master's thesis, Dept. of Electrical Engg. and Computer Science, Massachusetts Institute of Technology, Cambridge, 2002.

[21] S. Chen and K. Nahrstedt. On finding multi-constrained paths. In *In Proc. IEEE ICC'98*, June 1998.

[22] S. Chen and K. Nahrstedt. Distributed quality-of-service routing in highspeed networks based on selective probing. In *In Proc. of 23rd Annual Conference on Local Area Networks (LCN'98)*, Oct. 1998.

[23] B.V. Cherkassky and A.V. Goldberg. On implementing push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.

[24] The ATM Forum Technical Committee. ATM user-network interface signalling specification 4.0, March 1996.

[25] The ATM Forum Technical Committee. Private network-network interface specification 1.0, March 1996.

[26] O. Crochat and J.Y.L. Boudec. Design protection for WDM optical networks. *IEEE Journal on Selected areas in Communication*, 16(7):1158–1166, Sept. 1998.

[27] S. Crosby, I. Leslie, B. McGurk, J.T. Lewis, R. Russell, and F. Toomey. Statistical properties of a near-optimal measurement-based admission CAC algorithm. In *Proc. of IEEE ATM'97*, June 1997.

[28] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm. In *Proc. of ACM SIGCOMM'89*, pages 3–12, 1989.

[29] B.T. Doshi, S. Dravida, P. Harshavardhana, O. Hauser, , and Y. Wang. Optical network design and restoration. *Bell Labs Technical Journal*, Jan.–March 1999.

[30] K. Dovrolis and P. Ramanathan. Resource aggregation for fault tolerance in integrated services packet networks. *ACM Computer Communications Review*, 28(2):39–53, April 1998.

[31] A. Elwalid, C. Jin, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *in Proc. of IEEE INFOCOM, Anchorage, Alaska*, April 2001.

[32] A. Elwalid and D. Mitra. Design of generalized processor sharing schedulers which statistically multiplex heterogeneous qos classes. In *Proc. of IEEE INFOCOM'99*, pages 1220–1230, March 1999.

[33] D. Eppstein. Finding the k shortest paths. In *In Proc. of the 35th Annual Symposium on Foundations of Computer Science*, pages 154–155, November 1994.

[34] F. Ergun, R. Sinha, and L. Zhang. QoS routing with performance dependent costs. In *Proc. of INFOCOM 2000, Tel Aviv, Israel*, March 2000.

[35] P.M. Fernandez, N. McKeown, and H. Zhang. Is IP going to take over the world (of communications)? In *Proc. of HotNets-I, Princeton, NJ, USA*, Oct. 2002.

[36] D. Ferrari and D.C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, April 1990.

[37] V. Firoiu and D. Towsley. Call admission and resource reservation for multicast sessions. In *Proc. of IEEE INFOCOM'96*, 1996.

[38] S. Floyd. *Comments on measurement-based admission control for controlled load services*. Technical Report, Lawrence Berkeley Laboratory, July 1996.

[39] A. Fumagalli, I. Cerutti, M. Tacca, F. Masetti, R. Jagannathan, and S. Alagar. Survivable networks based on optimal routing and WDM self-healing rings. In *Proc. of INFOCOM'99, New York, NY*, pages 726–733, March 1999.

[40] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman, 1979.

[41] L. Georgiadis, R. Guerin, V. Peris, and K. N. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, August 1996.

[42] R. Gibbens and F. Kelly. Measurement-based connection admission control. In *Proc. of 15th Intl. Teletraffic Conference*, June 1997.

[43] S. Golestani. A stop-and-go queuing framework for congestion management. In *In Proc. of ACM SIGCOMM'96, Philadelphia, PA*, pages 8–18, Sept. 1990.

[44] S.J. Golestani. A self-clocked fair queuing scheme for broadband applications. In *Proc. of IEEE INFOCOM 1994*, pages 636–646, June 1994.

[45] K. Gopalan and T. Chiueh. Multi-resource allocation and scheduling for periodic soft real-time applications. In *Proc. of Multimedia Computing and Networking 2002, San Jose, CA, USA*, pages 34–45, Jan. 2002.

[46] R. Guerin and A. Orda. QoS-based routing in networks with inaccurate information: Theory and algorithms. *IEEE/ACM Transactions on Networking*, 7(3):350–364, June 1999.

[47] R. Guerin, A. Orda, and D. Williams. QoS routing mechanisms and OSPF extensions. In *Proc. of IEEE GLOBECOM'97, Phoenix, AZ*, volume 3, pages 1903–1908, Nov. 1997.

[48] L. Guo and I. Matta. Search space reduction in qos routing. In *In Proc. of the International Conference on Distributed Computing Systems*, pages 142–149, May 1999.

[49] S. Han and K.G. Shin. Fast restoration of real-time communication service from component failures in multihop networks. In *Proc. of ACM SIGCOMM'97*, pages 77–88, 1997.

[50] R. Hassin. Approximation schemes for restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, February 1992.

[51] C. Hedrick. Routing information protocol. RFC 1058, June 1988.

[52] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding phb group. RFC 2597, June 1999.

[53] S. Herzog. Rsvp extensions for policy control. RFC 2750, January 2000.

[54] C.-J Hou. Routing virtual circuits with timing requirements in virtual path based ATM networks. In *in Proc. of IEEE INFOCOM'96, pages =*.

[55] T. Ibaraki and N. Katoh. *Resource Allocation Problems: Algorithmic Approaches*. MIT Press, 1988.

[56] IETF: Differentiated Services Working Group, http://www.ietf.org/html.charters/diffserv-charter.html, January 2001.

[57] IETF: Integrated Services Working Group, http://www.ietf.org/html.charters/intserv-charter.html, September 2000.

[58] V. Jacobson, K. Nichols, and K. Poduri. An expedited forwarding phb. RFC 2598, June 1999.

[59] J.M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14:95–116, 1984.

[60] S. Jamin, P. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. *IEEE/ACM Transactions on Networking*, 5(1):56–70, February 1997.

[61] W. Jiang and H. Schulzrinne. Analysis of On-Off patterns in VoIP and their effect on voice traffic aggregation. In *Proc. of ICCCN 2000*, March 1996.

[62] C. Kalmanek, H. Kanakia, and S. Keshav. Rate controlled servers for very high-speed networks. In *In Proc. of IEEE Globecom, San Diego, CA*, pages 300.3.1–300.3.9, Dec. 1990.

[63] S. Kim, D. Qiao, S. Kodase, and K.G. Shin. Design and evaluation of routing schemes for dependable real-time connections. In *Proc. of Intl. Conference on Dependable Systems and Networks (DSN'01), Goteborg, Sweden*, July 2001.

[64] E.W. Knightly and N. B. Shroff. Admission control for statistical QoS. *IEEE Network*, 13(2):20–29, March 1999.

[65] M. Kodialam and S.H. Low. Resource allocation in a multicast tree. In *Proc. of INFOCOM 1999, New York, NY*, March 1999.

[66] M.S. Kodialam and T.V. Lakshman. Minimum interference routing with applications to MPLS traffic engineering. In *Proc. of INFOCOM'2000, Tel Aviv, Israel*, pages 884–893, March 2000.

[67] J. Kurose. On computing per-session performance bounds in high-speed multi-hop computer networks. In *Proc. of ACM Sigmetrics'92*, pages 128–139, 1992.

[68] LAN/MAN Standards Committee of the IEEE Computer Society. Ieee 802.11a standard, 1999.

[69] Y.J. Lin and M.C. Chan. A scalable monitoring approach based on aggregation and refinement. *IEEE Journal on Selected Areas in Communications*, 20(4):677–690, May 2002.

[70] D.H. Lorenz and A. Orda. Optimal partition of QoS requirements on unicast paths and multicast trees. *IEEE/ACM Transactions on Networking*, 10(1):102–114, February 2002.

[71] D.H. Lorenz and A. Orda. QoS routing in networks with uncertain parameters. *IEEE/ACM Transactions on Networking*, 6(6):768–778, December 1998.

[72] D.H. Lorenz and A. Orda. Optimal partition of QoS requirements on unicast paths and multicast trees. In *Proc. of INFOCOM'99*, pages 246–253, March 1999.

[73] D.H. Lorenz, A. Orda, and D. Raz. Optimal partition of QoS requirements for many-to-many connections. In *Proc. of INFOCOM'2003, San Francisco, CA*, April 2003.

[74] D.H. Lorenz, A. Orda, D. Raz, and Y. Shavitt. Efficient qos partition and routing of unicast and multicast. In *In Proc. of IWQoS 2000, Pittsburgh, PA*, pages 75–83, June 2000.

[75] J.M. McQuillan, I. Richer, and E.C. Rosen. The new routing algorithm for the ARPANET. volume 28(5), pages 711–719, May 1980.

[76] Mesh Network – Corporate and Technology Overview, http://www.meshnetworks.com/pdf/wp_corpoverview.pdf.

[77] E. Modiano and A. Narula-Tam. Survivable routing of logical topologies in WDM networks, anchorage, alaska. In *Proc. of IEEE INFOCOM 2001*, pages 348–357, April 2001.

[78] J. Moy. Ospf version 2. RFC 2328, April 1998.

[79] R. Nagarajan, J. Kurose, and D. Towsley. Local allocation of end-to-end quality-of-service in high-speed networks. In *Proc. of 1993 IFIP Workshop on Perf. analysis of ATM Systems, North Holland*, pages 99–118, Jan. 1993.

[80] P. Newman, G. Minshall, and T. Lyon. IP switching. *IEEE/ACM Transactions on Networking*, 6(2):117–129, April 1998.

[81] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of differentiated services field (ds field) in IPv4 and IPv6 headers. RFC 2474, Dec. 1998.

[82] Nokia. *Nokia RoofTop Wireless Routing*. White Paper, October 2002.

[83] A.K. Parekh. *A generalized processor sharing approach to flow control in integrated services networks*. PhD thesis, Massachusetts Institute of Technology, Feb. 1992.

[84] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case. *IEEE/ACM Transactions on Networking*, 2(2):137–152, April 1994.

[85] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[86] C. Philips. The network inhibition problem. In *In ACM Symposium on Theory of Computing*, May 1993.

[87] S. Plotkin. Competitive routing of virtual circuits in ATM networks. *IEEE J. Selected Areas in Comm.*, 13(6):1128–1136, 1995.

[88] J. Qiu and E. Knightly. Measurement-based admission control with aggregate traffic envelopes. *IEEE/ACM Transactions on Networking*, 9(2):199–210, April 2001.

[89] S. Ramamurthy and B. Mukherjee. Survivable WDM mesh networks: Part II – Restoration. In *Proc. of IEEE International Conference on Communications (ICC '99), Vancouver, Canada*, June 1999.

[90] S. Ramamurthy and B. Mukherjee. Survivable WDM mesh networks: Part I – Protection. In *Proc. of INFOCOM'99, New York, NY*, pages 744–751, March 1999.

[91] A. Raniwala, K. Gopalan, and T. Chiueh. *Multi-Channel Multi-Hop Wireless Mesh Networks*. Technical Report TR-142, Experimental Computer Systems Labs, Stony Brook University, Stony Brook, NY, June 2002.

[92] D. Raz and Y. Shavitt. Optimal partition of QoS requirements with discrete cost functions. In *Proc. of INFOCOM 2000, Tel Aviv, Israel*, March 2000.

[93] M. Reisslein, K.W. Ross, and S. Rajagopal. A framework for guaranteeing statistical QoS. *IEEE/ACM Transactions on Networking*, 10(1):27–42, February 2002.

[94] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). RFC 1771, March 1995.

[95] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, Jan. 2001.

[96] S. Sahni. Algorithms for scheduling independent tasks. *Journal of ACM*, 23:116–127, 1976.

[97] H. F. Salama, D. S. Reeves, and Y. Viniotis. A distributed algorithm for delay-constrained unicast routing. In *In Proc. IEEE INFOCOM'97, Japan*, April 1997.

[98] A. Sen, B.H. Shen, S. Bandyopadhyay, and J.M. Capone. Survivability of lightwave networks – path lengths in WDM protection scheme. *Journal of High Speed Networks*, 10(4):303–315, 2001.

[99] S. Sharma, S. Nanda, K. Gopalan, and T. Chiueh. *Viking: A Multi-Spanning-Tree Ethernet Architecture for Metropolitan Area and Cluster Networks*. Technical Report TR-141, Experimental Computer Systems Labs, Stony Brook University, Stony Brook, NY, June 2002.

[100] N. Shen and H. Smit. Calculating IGP routes over traffic engineering LSPs. Internel Draft, June 1999.

[101] A. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. RFC 2212, Sept. 1997.

[102] K. G. Shin and C.-C. Chou. A distributed route selection scheme for selecting real-time channel. In *in Proc. of Sixth Intl. Conf. on High Performance Networking*, pages 319–329, Sept. 1995.

[103] V. Sivaraman and F.M. Chiussi. Providing end-to-end statistical delay guarantees with earliest deadline first scheduling and per-hop traffic shaping. In *Proc. of IEEE INFOCOM 2000*, March 2000.

[104] H. Smit and T. Li. IS-IS extensions for traffic engineering. Internel Draft, August 2003.

[105] Sprint North American IP Backbone, http://www.sprintworldwide.com/english/maps/northamerica.pdf, June 2003.

[106] Q. Sun and H. Langendorfer. A new distributed routing algorithm with end-to-end delay guarantee. In *In Proc. of IWQoS'97*, May 1997.

[107] S. Suri, M. Waldvogel, and P.R. Warkhede. Profile-based routing: A new framework for MPLS traffic engineering. In *Proceedings of QofIS 2001, Portugal*, pages 138–157, Sept. 2001.

[108] J.W. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.

[109] J.W. Suurballe and R.E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14:325–336, 1984.

[110] Cisco Systems. *Understanding Delay in Packet Voice Networks*. White Paper, Sept. 2002.

[111] D. Verma, H. Zhang, and D. Ferrari. Guaranteeing delay jitter bounds in packet switching networks. In *Proc. of Tricomm 1991, Chapel Hill, North Carolina*, pages 35–46, April 1991.

[112] Y. Wang and Q. Zhu. Error control and concealment for video communication: A review. *Proceedings of IEEE*, 86(5):974–997, May 1998.

[113] Z. Wang and J. Crowcroft. Bandwidth-delay based routing algorithms. In *Proc. of IEEE GLOBECOM'95, Singapore*, pages 2129–2133, Nov. 1995.

[114] Z. Wang and J. Crowcroft. Qos routing for supporting resource reservation. *IEEE Journal on Selected Areas in Communication*, 14(7):1228–1234, Sept. 1996.

[115] J. Wroclawski. Specification of the controlled-load network element service. RFC 2211, Sept. 1997.

[116] X. Xiao, A. Hannan, and B. Bailey. Traffic engineering with MPLS in the internet. *IEEE Network*, 5:28–33, March/April 2000.

[117] X. Yuan. Heuristic algorithms for multi-constrained quality of service routing. *IEEE/ACM Transaction on Networking*, 10(2):244–256.

[118] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. of IEEE*, 83(10):1374–1396, October 1995.

[119] H. Zhang and D. Ferrari. Rate controlled static priority queuing. In *In Proc. of IEEE Infocom, San Francisco, CA*, pages 227–236, April 1993.

[120] L. Zhang. Virtual Clock: A new traffic control algorithm for packet-switched networks. *ACM Transactions on Computer Systems*, 9(2):101–124, May 1991.

[121] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource ReSerVation protocol. *IEEE Network*, 31(9):8–18, Sept. 1993.

[122] Q. Zheng and K.G. Shin. Fault-tolerant real-time communication in distributed computing systems. In *Proc. of 22nd Intl. Symposium on Fault-Tolerant Computing*, pages 86–93, July 1992.