Quality-of-Service Routing in Integrated Services Networks

Qingming Ma

January 1998

CMU-CS-98-138

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:
Peter Steenkiste, Chair
Garth Gibson
Allan Fisher
Hui Zhang
Lixia Zhang, UCLA

This research was sponsored by the Defense Advanced Research Projects Agency monitored by Air Force Material Command (AFMC), Hanscom AFB, under contract F19628-92-C-0116, and by Naval Command, Control and Ocean Surveillance Center (NCCOSC) under contract number N66001-96-C-8528.

Keywords: Integrated Services Networks, Quality-of-Service, routing, resource management, congestion control, max-min fair share, resource reservation, service disciplines, algorithms.

For my Mother

Abstract

Future integrated services networks will support multiple classes of service to meet the diverse quality-of-service (QoS) requirements of applications. To meet these end-to-end QoS requirements, strict resource constraints may have to be imposed on the paths being used. *QoS routing* refers to a set of protocols and algorithms that can select paths that satisfy such constraints while achieving high network throughput. QoS routing is challenging because (1) different service classes employ different resource sharing models, (2) service classes dynamically share link resources, and (3) selecting paths that meet multiple QoS constraints is a complex algorithmic problem.

This dissertation shows QoS routing in integrated services networks is both *desirable* and *feasible*. To support this claim, this dissertation develops an integrated QoS routing framework that has two components. The first component consists of routing algorithms for individual service classes that support either bandwidth guarantees, delay guarantees, or high throughput. By exploiting the relationship between QoS constraints, we develop polynomial routing algorithms for traffic classes that require stringent end-to-end performance guarantees. By coupling routing with finer-time scale resource management mechanisms such as congestion control and scheduling, we develop routing algorithms that achieve high throughput for best-effort traffic and low blocking rate for guaranteed traffic. By striking an appropriate balance between per-flow resource consumption and the distribution of network load, these algorithms improve resource utilization efficiency and network throughput under dynamic load conditions.

In a network that supports multiple classes of service, best-effort flows can experience congestion or even starvation if guaranteed flows are not routed appropriately. The second component of the proposed QoS routing framework is an effective inter-class resource sharing mechanism that also takes into consideration the link load of best-effort traffic while routing guaranteed flows. This mechanism is simple in the sense that it influences routing decisions by changing the link costs used for guaranteed traffic without requiring any change to the routing algorithms employed for individual service classes. In various scenarios, we observed significant performance improvements for best-effort traffic without sacrificing any performance for guaranteed traffic.

Acknowledgements

This dissertation would have never been possible without the care and support of many wonderful people.

First, I would like to thank my advisor, Peter. I started working with Peter after I had focused on programming language semantics for more than four years at CMU. The transition from doing research in programming language theory to doing research in computer networking turned out to be much harder than I ever imagined. Peter has been very patient and understanding and is always there no matter how busy his schedule. His confidence in me made it possible for me to go through this difficult time. Networking is such a fast growing and rapidly changing area. Peter has been very flexible to let me pursue my interests and fulfill my dream.

The other members in my thesis committee have also been providing continuous support. My co-advisor, Garth Gibson, spent a lot of time with me in the early stage of my thesis research. He helped me understand parallel file systems and their networking needs. Understanding this demanding application led me to the world of routing. His advice and challenges have motivated me to do high quality research. I am indebted to Allan Fisher for numerous conversations, continuous support, and encouragement. Hui Zhang is a wonderful person and a steady friend. I have benefited a great deal from his insights in networking and from numerous discussions with him. Special thanks go to Lixia Zhang for her friendship, enthusiasm, and encouragement while I was working as a summer intern at Xerox PARC. These discussions finally led to my decision to work in the area of computer networking. I also thank her for the numerous telephone conversations we had in the past few years and for being on my thesis committee.

For the first four years at CMU, I learned a lot from many world-class researchers in programming languages. I would like to express my deep thanks to John Reynolds, my former advisor, for his guidance. John taught me how to do research and how to do good research. He constantly reminded me to be very careful about things that seemed obvious. I spent four wonderful years with John and his family. Dana Scott has been taking care of me for all of my years at CMU, including the last few years while I was working on networking. Bob Harper has not only been the person with whom I had many stimulating technical discussions but also been one of my closest friends in my many years at CMU. Peter Lee taught me the advanced implementation techniques of programming languages. His friendliness and working style set an excellent example for the rest of my life. I would also like to thank Steve Brookes and Frank Pfenning for many discussions in these years.

I am grateful to many other faculty members at CMU for their guidance and friendship through the years. I benefited a great deal from many instructive discussions with Bruce Maggs. His insights in algorithms and complexity helped me in understanding the nature of routing algorithms. Dave Johnson is another wonderful person I would like to thank. Dave spent a lot of time with me on improving my presentation and technical writing skills and helped me to overcome many difficulties. I enjoyed numerous dinners with Dave as well as late nights when we were the only two people in the corridor. I thank Carl Love for his support and understanding while I used almost all available compute cycles on all CMCL machines to carry out my simulation experiments. I also thank Seth Goldstein, Ragunathan Rajkumar, and Doug Tygar for their friendship. Sharon Burks deserves my special thanks. Without her help, I would probably not have been able to concentrate on my work.

I warmly thank my colleagues in CMU Nectar and Darwin groups: Prashant Chandra, David Eckhardt, Jun Gao, Corey Kosak, Kam Lee, Andrew Myers, Eugene Ng, Sanjay Rao, Kay Sripanidkulchai, Donpaul Stephens, and Ion Stoica. They are all talented individuals and it has been a pleasure to work with them. I very much enjoyed their friendship. I am also grateful to my colleagues in CMU PDL group: Khalil Amiri, Fay Chang, Bill Courtright, Mark Holland, Hugo Patterson, Daniel Stodolsky, Jiawen Su, and Jim Zelenka. Six month of intern experience (1996) at AT&T research with K.K. Ramakrishnan proved to be beneficial to understanding traffic management and congestion control. Special thanks go to Sue Li at Geogia Tech and Tom Gee at Cisco Systems for spending much of their precious time on proofreading an early thesis draft and providing useful feedback.

A large and wonderful circle of friends has made my time at CMU special. In addition to the people mentioned already, I would like to thank Juergen Dingel, Maria Ebling, Andrzej Filinski, Fuchun Jiang, Longji Lin, Qi Lu, Berzsenyi Miklos, Benjamin Pierce, Henry Shum, Doug Smith, Shanghua Teng, Lun Wang, Ye-Yi Wang, Zhengyu Wang, Rex Xu, Jie Yang, and Matthew Zekauskas for their friendship and all of the good times. Matt kindly provided me with his LaTeX template which greatly speeded up my thesis writing.

Finally, I would like to thank my family for their constant emotional support during the past years. This allowed me to complete my graduate studies.

Contents

Chapter 1

Introduction

This dissertation is concerned with provisioning a set of service requirements, or *Quality-of-Service (QoS)*, in integrated services networks. *QoS routing* is a set of routing mechanisms under which flow path selection is based on knowledge of network resource availability as well as flow QoS requirements. This dissertation argues that QoS routing is *desirable* if we want to meet user requirements and achieve high network resource utilization efficiency. The novel algorithmic solution to QoS routing and resource management in integrated services networks presented in this thesis also shows that QoS routing is *feasible*.

The remainder of this chapter is organized as follows. Sections 1.1, 1.2, and 1.3 describe QoS provision, services models, and resource management in integrated services networks. Section 1.4 motivates the study of QoS routing. Section 1.5 outlines the approach, the main contributions, and the structure of the remainder of the dissertation.

1.1 Quality-of-Service Provision

The advent of broadband networking technology has dramatically increased the capacity of packet-switched computer networks from a few megabits per second to hundreds or even thousands of megabits per second. This increased data communication capacity makes it feasible to support new applications such as video conferencing, scientific visualization, internet telephony, multimedia email, medical imaging, video on demand, and distance learning. In the mean time, the volume of traditional best-effort data traffic continues to grow with Web applications, digital libraries, and I/O intensive scientific applications. In this environment, the data payload can vary from a few megabytes to several gigabytes.

This transformation of the Internet into an important and ubiquitous commercial infrastructure has significantly changed consumer expectation in terms of performance, security, and services. Unfortunately, best-effort service is still the only class of service offered in today's Internet. With best-effort service, all packets are typically treated equally in the network. Any congested link can induce increased packet delivery times, which, in turn, can result in generally poor performance, data jitter, or even packet loss.

Different applications require different QoS: some require stringent end-to-end delay, some require a minimal transmission rate, while others with no strict delay and/or bandwidth requirements may simply require high throughput[1]. Although numerous service models have been proposed to deal with QoS requirements most of them can be described as a function of delay, bandwidth, and throughput. The rest of this section describes these QoS requirements in detail.

1.1.1 Delay Guarantees

A broad class of applications, e.g., interactive multimedia, internet telephony, and video conferencing, may require stringent delay, delay jitter (or delay variation), and loss guarantees. For example, in real-time playback applications, packets arriving after the playback point will be useless, and the loss of a certain number of packets will serious degrade the quality of voice and pictures. When the application is not merely passive, as it is in playback, but interactive, these effects can be even more insidious and, if severe, can render the application useless.

End-to-end delay and delay jitter are cumulative (or additive) attributes of all links in the path. The end-to-end delay includes propagation delay, transmission delay, and queueing delay. While propagation delay is determined by the physical distance between the source and the destination, transmission delay is determined by the capacity of the bottleneck link on the path. Queueing delay, on the other hand, is determined by the network load, the burstiness of the traffic source, and the service disciplines employed in the network. Typically, the transmission delay is only seen for the first packet transmitted because of the pipelining of all other packets transmitted after it, while propagation delay is only a very small fraction of the end-to-end delay because of the increased link capacity. Queueing delay is potentially the dominant component in the end-to-end delay in a packet-switched network and can be limited through the use of different scheduling algorithms, conforming traffic sources, and bandwidth reservations ([105]).

[1]In this dissertation, the term QoS is used both for stringent QoS (e.g., delay, delay jitter, and packet loss) and for requirements to achieve high throughput. When we say that a session requires QoS guarantees, if the session is a best-effort session, it means that the session requires high throughput.

1.1.2 Bandwidth Guarantees

Transmission of multimedia streams requires a minimum bandwidth to ensure end-to-end QoS guarantees. Bandwidth guarantees can be requested for different time intervals depending on applications. For example, if an application is adaptive and has sufficiently large buffer space at its source and destination, the bandwidth provided by the network can vary over time, as long as the average bandwidth provided is higher than the minimum bandwidth required by the application. If an application is less adaptive, the network may have to reserve more bandwidth than the amount that matches the average packet sending rate. Recent studies [41] suggest that the network should deploy a mechanism to support bandwidth renegotiation, which allows bandwidth reservation to be provided at a finer time scale than per-session bandwidth guarantees.

1.1.3 High Throughput

Traditional best-effort applications, e.g., Remote Procedure Call (RPC), electronic mail, ftp, and telnet, usually send messages as small as a few kilobytes. The main performance index for these applications is the end-to-end per packet delay. As the sophistication of networked applications has grown, so too has the amount of data transmitted. It is now not uncommon to observe applications whose data payload reaches from several hundred megabytes to several terabytes [25, 37, 36]. In contrast to the transmission of small messages, these new applications can consume as much network bandwidth as is available. It is the end-to-end throughput rather than the per packet end-to-end delay that is the main performance concern. The throughput is determined by the total number of bytes transmitted over the elapsed time, where the elapsed time includes the end-to-end delay experienced by the first packet and the time interval from the arrival of the first packet to the arrival of the last packet. Because of their payload variances and their ability to consume arbitrary amounts of network bandwidth, these flows should be treated differently inside the network.

1.2 Integrated Services Networks

Historically, transmission of voice, video, and data has been carried over different types of networks: telephony networks for voice, cable networks for video, and computer networks for data. Each network provides its own dedicated resource management that is optimized for its particular usage.

With the increased capacity of data networks and the emergence of multimedia applications, it is desirable and feasible to have a single network over which voice, image, and data streams can all be sent at the same time with appropriate end-to-end user QoS. Since these streams with different QoS requirements share network resources, it is extremely challenging to achieve high multiplexing efficiency while ensuring end-to-end QoS guarantees for individual streams.

To balance the complexity for QoS management within the network against the diversity of applications, the network should provide multiple classes of service with different QoS guarantees. Such a network is called an *integrated services network* because it provides services (e.g., real-time voice, video, and best effort) that previously required separate dedicated networks.

The major networking standards bodies including the Internet Engineering Task Force (IETF) and the Asynchronous Transfer Mode (ATM) Forum as well as the networking research community at large have devoted a large body of effort to creating diversified services for integrated services networks. The current approach is to classify traffic according to its performance requirements and traffic characteristics. Two such multiple-classes-of-service models—the Internet service model [11, 93, 103, 16] and the ATM service model—are being defined by the IETF and ATM Forum [4], respectively. In this section, we introduce these service models.

1.2.1 Internet Service Model

New service classes being defined in the IETF include *guaranteed service* [93], *controlled load service* [103], and more recently *differentiated service* [16, 76]. While the differentiated service model is still in an embryonic stage, the definition of the guaranteed and controlled load service classes is well developed. Both service models involve the establishment of connections through the network with the help of a resource reservation protocol, such as the ReSerVation Protocol (RSVP) [107, 12], and *admission control mechanisms* [17, 54] based on measured network state information. The network ensures that sufficient resources are available once a flow (or a session[2]) is admitted.

The guaranteed service model is based on recent studies [77, 26, 17, 105, 98, 39] of a class of *rate-proportional* packet scheduling algorithms. Under these service disciplines, packets of different connections sharing the same output link are sent in an order that ensures a weighted fair sharing of link capacity among these connections. As a result, a guaranteed rate is ensured for flows that use the guaranteed service. Packets transmitted on such flows are thus protected from either ill behaved applications or intentional link sabotage. Moreover, mathematically provable worst-case delay, jitter, and buffer space bounds exist if the traffic source conforms to its traffic specification given by a leaky bucket (see Chapter 6 for the detail definition of leaky bucket).

[2]In this dissertation, we use the terms "flow", "session", and "connection" interchangeably. The precise definition of flow appears in Chapter 2.

To invoke the guaranteed service, an application specifies its traffic characteristics and desired performance guarantees. The network, on the other hand, reserves a certain amount of resources at each node (switch or router) on its path. Thus, the traffic specification and the QoS guarantees constitute a “contract” between the network and the application: once a flow is admitted into the network and the traffic source conforms to its traffic specification, the network will provide guaranteed QoS.

The controlled load service is an enhanced best-effort service intended to support applications requiring performance better than what is provided by traditional best-effort service. It limits the amount of traffic entering the network to ensure that once a flow is admitted, it enjoys service equivalent to best-effort service in a lightly loaded network. Even under congestion, network nodes offering controlled load service are expected to provide flows with low delay and low packet loss. To limit the number of flows receiving the service, it requires applications to make explicit requests for service. Such requests for service can be made using a reservation setup protocol, such as RSVP, or some other means. Each network node that receives a request for service can either accept or reject the request. Therefore, a flow can receive a guaranteed average rate with no per packet delay guarantees.

1.2.2 ATM Service Model

ATM applies cell switching technology. It employs connection-oriented virtual circuits to maintain the connection state inside the network so as to make it easier to provide QoS guarantees. The ATM forum defined five service classes [4]: Constant Bit Rate (CBR), real-time Variable Bit Rate (rt-VBR), non-real-time Variable Bit Rate (nrt-VBR), Available Bit Rate (ABR), and Unspecified Bit Rate (UBR).

The CBR service class makes available a fixed quantity of bandwidth for each connection. It provides bounds on delay and delay jitter to traffic that can be characterized by its peak rate. The rt-VBR service class provides tight constraints on delay and delay variation. It is designed for soft real-time applications, such as compressed video, that have a variable transmission rate. The nrt-VBR service class provides guarantees on the average delay and maximum loss rate of a connection. It is designed for extremely bursty, time-critical applications such as banking transactions and airline reservations. The ABR service class enforces a bound on the minimum throughput of connections and additionally divides unused bandwidth fairly among connections sharing a link. It is designed for applications that can adapt their traffic rate in accordance with the changing availability of network resources. The UBR service class does not provide any performance guarantees.

1.3 Resource Management

To successfully provide new services to meet applications' QoS requirements, the network must employ effective resource management mechanisms. There are two key objectives for the resource management mechanisms employed in integrated services networks:

- Ensure per-flow end-to-end QoS as long as applications abide by the traffic contract agreed to when connections are set up.
- Achieve high network resource utilizations efficiency through efficient multiplexing, resource sharing, and load balancing.

At the core of resource management are scheduling algorithms, congestion control algorithms, resource reservation, and routing. These mechanisms manage the network resources on different time scales and with different granularities. In this section, we explain these resource management components.

1.3.1 Scheduling Disciplines

Packet scheduling is the resource management mechanisms operating on the finest time scale. Its main function is to schedule packets from different flows sharing the same output link in such a way that every flow gets its appropriate share of the link bandwidth.

The most popular packet scheduling algorithm used in today's data network is the First-In-First-Out (FIFO) algorithm. With FIFO scheduling, packets are scheduled according to the order in which they are received. It is clear that a burst of packets from one flow can lead to large delay for packets from other flows. Other popular packet scheduling algorithms include priority-based scheduling and Round-Robin scheduling (see [105] and its references). More recently, a class of rate-proportional service disciplines have been identified and carefully analyzed [77, 26, 17, 105, 98, 39]. One example of a rate-proportional scheduling algorithm is *Weighted Fair Queueing (WFQ)*. With WFQ [77], packets are sent in an order that ensures that each flow using the link gets a fraction of the link capacity that is proportional to its weight.

1.3.2 Congestion Control

The main function of congestion control is to ensure that the total amount of traffic from all sources sharing the same link does not exceed the link capacity. Congestion control is becoming more important and more challenging because of increasing bandwidth-delay products in high speed networks.

The most popular congestion control schemes use *window-based* congestion control [80, 48, 19, 23] such as the congestion control deployed in today's Internet. The sender maintains a congestion window to limit how fast data is sent and a time out to detect lost packets. Inside the network, packets are dropped when buffers are full. The sender interprets lost packets as a sign of congestion and reduces the congestion window. Recently, new congestion control mechanisms, e.g. Random Early Detection (RED) [30] and Explicit Congestion Notification (ECN) [84], queue management schemes, buffer sharing policies, e.g., Fair Random Early Drop (FRED) [65], have been the focus of much research.

With the availability of per-connection state in ATM networks, explicit rate-based congestion control has been proposed for the ATM Available-Bit-Rate service [4]. With rate control [82, 86, 85, 14, 52, 55], sources periodically send a resource management (RM) cell. As the RM cell travels from the source to the destination, an explicit rate is calculated and inserted in the RM cell by the switches. Upon receiving an RM cell, the destination node sends it back to the source, and the source adjusts its sending rate according to the explicit rate information contained in the RM cell. A queue management algorithm is developed in [66]. Parallel to the development of rate-based congestion control, *credit-based* congestion control [60, 57, 58, 59, 83] has been developed. In such a scheme, an upstream node can send a packet only if it has received sufficient credits from the downstream node, indicating that there is enough buffer space.

1.3.3 Resource Reservation and Admission Control

For the network to deliver quantitatively specified QoS (e.g., a bound on delay) to a particular flow, it is usually necessary to set aside certain resources, such as a share of the bandwidth. The function of resource reservation is to ensure that a flow will get its requested resources once the flow is admitted into the network. Several resource reservation protocols have been developed [107, 12, 100]. These reservation protocols provide a general facility for creating and maintaining distributed reservation states across a mesh of multicast or unicast delivery paths.

Because the network's resources are finite, it cannot grant all resource reservation requests. In order to maintain the network load at a level where all QoS commitments can be met, the network must employ an admission control algorithm that determines which requests to grant and which to deny, thereby maintaining the network load at an appropriate level [17, 53].

Upon the arrival of a new flow request, the reservation protocol employed by the network sets up a connection along the path that is selected by the routing mechanism. At each hop, the admission control scheme determines if the link has sufficient resources to accommodate this new flow. If so, the resources needed for the flow are set aside and the connection setup packet is sent to the next hop. If insufficient resources are available, either the flow is rejected or other alternative paths are tried again through the use of a *crankback* mechanism [27].

1.3.4 Routing

Routing is the coarsest grain resource management mechanism. The goal of routing is to direct traffic from source to destination in accordance with the traffic's service requirements and the network resource constraints, while maximizing network performance and minimizing the cost of the path based on a link cost function. The main routing functions are as follows.

- State information advertising: assembling and distributing user traffic state information which is used in generating and selecting paths. This state information includes service requirements, and services provided by and resources available within the network. It may comprise both measured and predicted values.
- Path selection: selecting *feasible* or even *optimal* paths based on user and network state information. A path is feasible if it satisfies all user-imposed and network-imposed service constraints. Optimal paths are feasible paths that are "best" with respect to a specific performance objective.
- Traffic forwarding: forwarding the user traffic along the selected paths.

Path selection algorithms are classified as either "static" or "dynamic" based on their use of real time network state information. As its name implies, static path selection algorithms (or static routing algorithms) do not require or use real time state information. While simple to implement and use, these schemes are subject to obvious limitation, which make them largely impractical for QoS routing. Dynamic path selection algorithms (or dynamic routing algorithms), on the other hand, make use of real time network state information to generate and maintain an accurate real time picture of network topology. With dynamic routing, it is possible to select paths that are both feasible and optimal. Different routing algorithms may require different state information to be advertised (e.g., hop count, bandwidth, or buffer space) and different advertising mechanisms (e.g., flooding the whole network or exchanging information with local neighbors) may be used.

It is worth mentioning that no matter how well a network is designed, routing is always an important network function that should respond to changes in the topology and the network state. *Network design* often relies on long-term measurements of user traffic and load. Its objective is to produce a network topology at minimal equipment cost that can, in conjunction with routing, accommodate the expected user traffic under specified network conditions. Thus, static routing cannot be expected to work as well as dynamic routing since the dynamic traffic load will not always precisely match the long-term average load the network was designed for.

1.4 Quality-of-Service Routing

For traffic requiring stringent performance guarantees (e.g., delay, delay jitter, and bandwidth), Quality-of-Service routing selects paths that meet the resource constraints imposed by the user's QoS requirements. If there are several feasible paths available, the one that leads to higher network throughput should be selected.

While many studies on QoS provisioning have looked at on scheduling, congestion control, and resource reservation, QoS routing has received much less attention. In this section, we discuss why QoS routing is important for an integrated services network, how QoS routing is different from routing in a circuit-switched network or in today's packet-switched data networks, and what challenges are associated with QoS routing.

1.4.1 Routing in Circuit-switched Networks

The most familiar circuit-switched network is the global telephone system, which carries primarily voice traffic. Circuit-switched networks establish physical circuits from sources to destinations in response to call requests. Transmission resources are dedicated to the user for the duration of the call.

Historically, the telephone networks have relied on static, preconfigured paths that are computed off-line at a central facility and subsequently loaded into the switches. To maximize the likelihood of providing the service under dynamic condition, the central facility may provide each switch with multiple paths to each destination. These paths are of two types: (1) alternative routes for use when calls are blocked on the primary path and (2) time-dependent routes for use at different hours of the day.

With the introduction of stored-program-control switches with high speed processing capabilities, dynamic routing has become a feasible alternative for telephone network routing. An important class of dynamic routing strategies are *state-dependent routing* algorithms, that attempt to route each call so as to minimize the blocking probability of future calls. Trunk Reservation and Dynamic Alternate Routing [35], and Least Load Routing [45] are some of the algorithms that have been deployed in major telephone networks. One of the main reasons why these algorithms work well is that the telephony network is often fully connected. This is typically not the case for data networks.

1.4.2 Routing in Packet-switched Networks

Most packet-switched networks employ some form of shortest-path routing. The objective of shortest-path routing is to determine a least-cost path between a source and a destination. The cost can be hop count, delay, bandwidth, queue size, or combinations thereof.

Shortest-path routing algorithms are divided into two classes: *distance vector* and *link state* algorithms [95]. Distance-vector algorithms are derived from the Bellman-Ford shortest-path algorithm [20], and base path selection on local estimates of path costs. They are usually implemented in a distributed asynchronous fashion. Link-state algorithms, on the other hand, are derived from Dijkstra's shortest-path algorithm [28]. They are usually implemented in a replicated fashion (each node performs an independent route computation) and construct paths based on global estimates of individual link costs. The original ARPAnet routing scheme was a distance vector protocol with a delay-based cost metric [73]. Such a scheme was shown to be prone to route oscillations [8]. For this and other reasons, a link state delay-based routing scheme was later developed for the ARPAnet [73].

As packet-switched networks grow to accommodate an increasing user population, the amount of routing information that must be distributed, stored, and manipulated in these networks also grows. Research on routing in large packet-switched networks has focused on ways to reduce the quantity of routing information, without sacrificing the quality of a selected path. The majority of proposed and deployed solutions use algorithms for hierarchical clustering of topology information, abstraction of routing information relating to these clusters, and packet forwarding within the hierarchy [56, 32, 27, 96].

Routing algorithms deployed in today's Internet focus on basic connectivity and typically support only one type of datagram service—best effort service. Current Internet routing protocols, *e.g.*, BGP [88], OSPF [74], RIP [47], use "shortest path routing", *i.e.*, routing that is optimized for a single metric, e.g., administrative weight or hop count. These routing protocols are also "opportunistic," *i.e.*, they use the current shortest path or route to a destination. In order to avoid loops, alternative paths with adequate but less than optimal cost can not be used to route traffic (shortest path routing protocols do allow a router to alternate among several equal cost paths to a destination).

1.4.3 QoS Routing: Challenges

None of the routing schemes developed for circuit-switched telephony networks and for packet-switched networks can be directly applied to QoS routing in integrated services networks. For example, the trunk reservation and the Dynamic Alternate Routing algorithms developed for circuit-switched telephone networks are not applicable to data networks because data networks are rarely fully connected, the bandwidth requirements are disparate, and the delay in a packet-switched network includes not only propagation delay but also queueing delay. Similarly, routing in today's Internet is still based on connectivity.

Most recently, QoS routing has begun to receive attention in both the ATM community and the Internet community [63]. Activities include the development of the PNNI routing protocol [27] for ATM networks and a QoS routing framework for the future Internet [21, 104] in the IETF. There is also a proposed extension to the current OSPF routing protocol to support QoS routing [44]. These protocols, along with some appropriate routing algorithms, once deployed, will support QoS based routing. However, QoS routing algorithms are not yet well understood. The main challenge comes from the complexity of selecting a path with multiple constraints, the diverse intra-class resource sharing behavior in individual service classes and the complexity of dynamic inter-class resource sharing among multiple service classes. These factors make it difficult to achieve high resource utilization efficiency.

- *Route Computation Complexity*. Selecting a feasible path with a single QoS constraint can be realized by any shortest-path algorithm. However, selecting a path with multiple QoS constraints, which is required for some service classes, is in general NP-complete [33, 102]. For example, both delay and delay jitter are additive QoS constraints, and finding a feasible path with given delay and jitter bounds is NP-complete. Several studies [49, 90] have proposed heuristics to approximate the NP-complete problem, while others [81] suggest solutions for a subset of the NP-complete problem. In practice, even a polynomial algorithm may not be computationally efficient for a large topology.

- *Achieving High Resource Utilization Efficiency*. With increased network connectivity, more than one feasible path is often available. This raises the question of which path should be selected to achieve high network throughput. The question shows up in a number of cases:

  - With multiple QoS constraints, it is unclear which QoS parameter should be the main objective of optimization.
  - For different traffic classes, the resource sharing models are different. Different routing algorithms may need to be used to achieve high intra-class sharing efficiency.
  - When multiple classes of traffic are simultaneously present in the network, it is difficult to partition the link capacity over multiple traffic classes while achieving high inter-class resource sharing efficiency.

  There are typically two methods to achieve high network throughput: *conserving resources* and *balancing traffic load*. To conserve resources, we would like to select paths with as few hops as possible or requiring as little bandwidth as possible. To balance the network load, we would like to select the least loaded paths. In practice, these two methods often conflict with each other. This raises the question of when routing algorithms should put more weight on load balancing and when they should they put more weight on conserving resources. Can routing algorithms automatically make the right tradeoff between conserving resources and balancing the traffic load?

- *Impact of Inaccurate Routing Information*. Since the state information used by routing algorithms is advertised over the network periodically, the node making routing decisions is usually using stale state information. This raises the question: how frequently should the routing information be advertised to ensure low operational cost, while at the same time achieving accurate path selection and high resource utilization efficiency? Should different routing information update intervals be used for different traffic classes?

These issues have to be well understood before QoS routing can be a reality, and they are the focus of this dissertation. Other important issues, such as the scalability and the role of state and flow aggregation in QoS routing, are outside of the scope of this thesis, but they are briefly covered in our discussion on future work in Chapter 8.

1.5 The Thesis

This thesis presents a novel solution to QoS routing in integrated services packet-switched networks. It addresses issues related to not only path selection complexity but also resource utilization efficiency and it evaluates the impact of inaccurate routing information.

1.5.1 Approach

A central theme throughout the thesis is the coupling of routing with finer grain resource management mechanisms. While resource management can be split into a number of components operating on different time scales and with different functionality, focusing on one component while ignoring others can result in poor performance or overly expensive solutions. In this dissertation, we tightly couple the routing algorithms with the scheduling and congestion control algorithms used in the network. By taking scheduling and congestion control algorithms into consideration, we can find more efficient and simpler solutions to QoS routing.

Our approach is pragmatic: we use realistic service models and focus on practical routing algorithms. Theoretically optimal routing algorithms can be attractive because they can potentially achieve high resource utilization efficiency. However, this efficiency may not be achievable without introducing high operating costs, such as complex route computations. More sophisticated protocols may also have to be deployed to support these algorithms. Simplicity has been one of the main reasons for the success of today's Internet. Incremental changes to the existing protocols are crucial in keeping the Internet functioning well, while starting with simple and practical algorithms is an important first step in understanding the full complexity of QoS routing.

The evaluation is carried out using an extensive simulation study. Theoretical analytic models can obtain provable results. However, with the complexity of today's network topology, traffic distributions, and the use of inaccurate routing information, it is an extremely difficult task to do analytic modeling without sacrificing realism. In our simulation study, we choose realistic network topologies and diverse traffic load distribution. We incorporate realistic routing information distributions into our performance evaluation.

The study is carried out in two steps. We first develop routing algorithms for individual traffic classes. We investigate the performance impact when different intra-class sharing models are employed and when different routing information update intervals are used. We then study how to integrate the routing algorithms obtained for individual traffic classes into a global multi-class routing architecture.

1.5.2 Contributions

In this dissertation, we present a QoS routing framework for an integrated services network that provides services with delay guarantees, bandwidth guarantees, and high throughput. The framework has of two levels. At one level, we develop, for each individual service class, practical routing algorithms that not only find feasible paths but also achieve high resource utilization efficiency. At the other level, we introduce an integrated routing framework that avoids starvation and congestion of low priority traffic without sacrificing the performance of high priority guaranteed traffic.

To the best of our knowledge, our simulation study is the first that considers not only the throughput for the traffic class being routed but also its impact on the performance of best-effort traffic. This thesis makes the following original contributions:

- For best-effort traffic that requires high throughput, we developed routing algorithms that use congestion information as link state. By using a distance function that balances the path length and width, significant improvements in average per-session throughput can be achieved. To achieve even higher throughput without reducing the performance of single-path sessions, we proposed a novel prioritized multi-path routing algorithm, in which lower priority paths share the bandwidth left unused by higher priority paths. This leads to a new multi-level prioritized max-min fairness model. Simulation results show an increase of up to 35% in throughput by using a second path, while other single-path sessions also enjoy an increase up to 5%.

- For traffic requiring bandwidth guarantees, we systematically evaluate several promising routing algorithms. The evaluation considers not only the call blocking rate but also the fairness to requests for different bandwidths, robustness to inaccurate routing information, and sensitivity to the change of the routing information update interval. We show that routing algorithms that favor fewer hops perform best. The bandwidth efficiency of using pre-computed paths for several bandwidth intervals is comparable to that of computing paths on demand, which implies the feasibility of class-based routing.

- For traffic requiring delay guarantees, the general QoS routing problem of finding a path with delay, delay-jitter, and buffer space constraints is NP-complete. However, we show that the complexity of QoS routing is polynomial if the network service disciplines are rate-proportional and the relationship between QoS constraints introduced by these service disciplines are exploited. The resulting algorithms simply iterate the Bellman-Ford algorithm over all different residual bandwidth values in the network. To achieve high resource utilization efficiency, we identified four candidate optimality criteria and show that an efficient path must consider both load balancing and resource consumption in order to perform consistently well. Furthermore, we propose and evaluate an approximation algorithm that iterates the Bellman-Ford algorithm only a constant number of times.

- For an integrated services network, a routing architecture that simply combines routing algorithms obtained for individual classes may cause serious congestion for low priority traffic. We introduce a simple inter-class resource sharing architecture that takes into consideration the load of lower priority traffic while routing a higher priority session. The resulting multi-class routing algorithm performs consistently well regardless of the traffic load, network topology, and mixture of different traffic classes. In cases where the network load is unevenly distributed, it can improve the performance for low priority traffic significantly without sacrificing the performance of high priority traffic.

1.5.3 Thesis Statement

Our results can be summarized in the following thesis statement:

> Quality-of-Service routing for integrated services networks is both desirable and feasible. By striking an appropriate balance between per-flow resource consumption and network load distribution, it can achieve low call blocking rates for guaranteed traffic and high throughput for best-effort traffic. By coupling routing with finer-grain resource management mechanisms such as congestion control and scheduling, practical QoS routing algorithms can be developed that can be introduced with few changes to the existing routing protocols.

1.5.4 Thesis Outline

The remainder of this dissertation is structured as follows. In the next chapter, we define the network and routing models that are used in the remainder of the thesis. Chapter 3 describes the simulation design, including topologies, traffic loads, and performance metrics. Chapter 4, 5, and 6 study routing for traffic requiring high throughput, bandwidth guarantees, and delay guarantees, respectively. Chapter 7 describes an inter-class resource sharing mechanism that integrates the routing algorithms for individual traffic classes. Finally, in Chapter 8, we summarize our contributions and discuss some future work.

Chapter 2

Network Model

In this chapter, we present a network model to be used in the rest of the dissertation. Instead of using a particular network as a model, such as the Internet or an ATM network, we employ a model that can encompass a variety of networks.

This chapter is organized as follows: Section 2.1 describe the flow-based network model to be used in the rest of the dissertation. Section 2.2 defines four classes of traffic we study. Section 2.3 discusses the resource sharing model employed by each of these traffic classes. Section 2.4 presents a brief discussion on a number of routing algorithms deployed in today's data networks. We summarize in Section 2.5.

2.1 Flow-based Model

We consider networks with arbitrary topologies and heterogeneous link capacity. Each link in the network has a fixed propagation delay, which is determined by the physical distance between the two nodes.

2.1.1 Flows in Packet-switched Networks

There are two types of packet-switched networks: virtual circuit networks and datagram networks. In a virtual-circuit network, *e.g.*, an ATM network, a connection is setup before data transmission starts and is torn down after data transmission is completed. All packets for a connection are transmitted in sequence from the source to the destination. In such a network, state information is maintained at every node along the path of the virtual circuit. This state information can be made available to admission control, resource reservation, and QoS routing components of the network.

In a connectionless datagram network, *e.g.*, the Internet, all information needed to deliver a packet is carried in the packet header and individual forwarding decisions are made as packets arrive in a switch or a router. No connection state information is maintained inside the network. The concept of a connection still exists on the end-systems, but there is no explicit connection setup and tear down inside the network. Different packets of the same connection may be sent over (possibly) different paths from the source to the destination.

Without state information inside the network, it is hard for the network to provide end-to-end QoS guarantees. In order to provide these guarantees, the current trend is to maintain per-session state information or the aggregation of this state information in the network. State information can be maintained in two ways. *Soft state* refers to state that is periodically refreshed by the reservation protocol deployed in the network, so, when lost, the state will be automatically reinstated. For example, in RSVP, two kinds of soft state information are maintained: path state and reservation state. Each data source periodically sends a *Path message* that establishes or updates the path state, while each receiver periodically sends a reservation message that establishes or updates the reservation state. In contrast, *hard state*, such as the state maintained in an ATM network, is established at connection setup time and is maintained for the connection life time. A recent suggestion [16] is to maintain per-flow state only at the edge of the network, while the network core maintains only aggregated state information.

The next generation of the Internet Protocol (IPv6) [24] provides enhancements over the capabilities of the existing Internet Protocol version 4 (IPv4) service. IPv6 adds capabilities to label packets belonging to particular traffic "flows". A flow in IPv6 is a sequence of packets sent from a particular source to a particular destination. The 24-bit Flow Label field in the IPv6 header may for example be used by a source to label those packets for which it requests special handling by the IPv6 routers, such as non-default quality of service or "real-time" service. The nature of that special handling might be conveyed to the routers by a control protocol, such as RSVP, or by information within the flow's packets, e.g., in a hop-by-hop option.

2.1.2 Flow Definition

We use the term "flow" to capture the concept of a virtual circuit in ATM networks and a flow in IP networks. In other words, a flow can be a hard-state virtual circuit as in an ATM network, a soft-state flow as defined in IPv6 [24], or a stateless connection as a TCP connection in today's IP network.

A flow (or a session) can use a single path or multiple paths. For traffic requiring resource guarantees, we assume that the traffic source specifies its traffic characteristics and desired performance guarantees. The network does admission control to decide if the traffic should be admitted. Once a flow is admitted, resources along its path are reserved through a reservation protocol, e.g., RSVP [107].

In this dissertation, we focus on point-to-point per-session flows. Understanding QoS routing for such flows is a fundamental step towards realizing QoS routing in future integrated services networks and is also an essential step towards understanding routing support for more complex services such as multicast and multipoint-to-multipoint flows.

2.2 Traffic Classes

We study two different types of flows: guaranteed flows and best-effort flows. Guaranteed flows require QoS guarantees in terms of delay, delay jitter, and bandwidth. Best-effort flows require no firm QoS guarantees but would like the network to deliver packets to their destinations as quickly as possible. For guaranteed flows, we consider two different types of QoS requirements: *bandwidth guarantees* and *delay guarantees*. For best-effort flows, we distinguish between two subclasses: *high-bandwidth* traffic and *low latency* traffic.

$$\text{traffic classes} \begin{cases} \text{best}-\text{effort sessions} \begin{cases} \text{low latency} \\ \text{high bandwidth} \end{cases} \\ \text{guaranteed sessions} \begin{cases} \text{bandwidth guarantee} \\ \text{delay guarantee} \end{cases} \end{cases}$$

2.2.1 Best-Effort Sessions

Best-effort traffic has been and will continue to be the main traffic class in data networks for the foreseeable future. However, with the increasing data sizes used by applications and the increased network capacity, large data sets from a few megabytes to several gigabytes are often transmitted over the network. In contrast to traditional best-effort traffic, such as electronic mail, telnet, RPC, and small file transfers, this new class of applications can send data at a high burst rate and can make use of the high bandwidth available in the network. We call this new class of traffic *high-bandwidth* traffic while referring to traditional best-effort traffic as *low-latency* traffic.

Both high-bandwidth and low-latency traffic want data to be sent with minimum elapsed time, but the factors that affects the elapsed time is different. A low-latency message may consist of only a few packets, and the packets should be sent to their destination with the minimum per-packet end-to-end delay to achieve the minimum elapsed time. For high-bandwidth traffic, however, a message can consist of hundreds or thousands of packets. When sending these packets to their destinations, the available bandwidth on the path will dominate the elapsed time. Given the different traffic characteristics and the different dominating factors determining the performance of low-latency traffic and high-bandwidth traffic, the network should provide explicit support to optimize user satisfaction as well as network resource utilization efficiency for these two types of traffic.

2.2.2 Guaranteed Sessions

Among guaranteed flows, we make a distinction between those that require only bandwidth guarantees and those that require also delay guarantees. For a flow requiring bandwidth guarantees, the requested bandwidth can either be the minimal bandwidth or the average bandwidth. For a flow requiring delay guarantees, the QoS requirements also include delay, delay jitter, and loss. To provide such QoS guarantees, the network needs to reserve a certain amount of bandwidth at each node on the path from the source to the destination. In many cases, much more bandwidth than the average traffic rate may have to be reserved if the requested delay bound is tight and the traffic source is bursty. The reserved but unused bandwidth may be used opportunistically by other flows in the network. The responsibility of the scheduling algorithms is to ensure that the link share for each flow corresponds to its requested QoS. We will discuss scheduling algorithms in more details in Chapter 6.

2.3 Sharing Polices

One of the main characteristics of packet switched networks is the statistical multiplexing of link resources among flows using the same link. In such a network, link resources are dynamically shared among the flows that use the same link. In a network that provides multiple classes of service, different styles of sharing can be enforced among sessions of different service classes. Our model uses two styles of resource sharing: reservation and fair sharing.

2.3.1 Resource Reservation

Resource reservation extends the provisioning of guaranteed services from circuit switched networks into today's packet switched networks. This enables resource management mechanisms to be deployed that isolate sessions from one another. Our model assumes that all guaranteed sessions use resource reservation. For traffic requiring bandwidth guarantees, the requested bandwidth can be a minimum bandwidth or an average bandwidth. The requested bandwidth is reserved by the network. For traffic requiring delay guarantees, the traffic source specifies an average rate and the maximal burst size. More bandwidth than the average source sending rate may have to be reserved in order to achieve a tight delay bound when the burst size is large.

Since the actual bandwidth consumed by a session is lower than the requested bandwidth, the overbooked bandwidth can be either used by best-effort traffic.

The network employs an admission control algorithm to determine if there are adequate resources available to admit a new flow without affecting the flows that already admitted. The decision can be based either on a measurement-based mechanism [54] or on an analytic model such as equivalent capacity [42].

2.3.2 Max-min Fair Sharing

Our goal of routing best-effort traffic is to improve the performance of high-bandwidth sessions. There are several alternatives that can achieve this goal. One alternative is to differentiate high-bandwidth traffic from low-latency traffic and give high-bandwidth traffic preferential treatment at the expense of lower performance for low-latency sessions [18]. Another alternative is to devise reservation-based schemes that reserve a certain amount of bandwidth for high-bandwidth best-effort sessions. The difficulty in reservation-based schemes lies in determining the amount of bandwidth to reserve. A session may be blocked by the network if the requested bandwidth is not available.

We focus on a sharing policy that treats all best effort sessions equally, and does not discriminate against any best-effort sessions in the network. This sharing policy is closer to the dynamic resource sharing in today's Internet and in ATM networks [46, 14].

Max-min fair sharing is defined so that flows sharing the same link will each get an equal share of the link bandwidth. If a flow cannot use its share, the excess bandwidth is split "fairly" among all other flows who can use more bandwidth. A flow may not be able to use its share either because it has a lower source rate or it is bottlenecked at another link. This process is repeated until all flows in the network become bottleneck.

There are several definitions of "fair share". In the basic definition, if N flows share one output link, each of the N flows competing for the link or excess bandwidth gets one N^{th} of the bandwidth. Other definitions, such as weighted fair sharing and multi-class fair sharing, require pricing or administrative policies to assign a weight or class to each connection. Since these policies are still an active area of research, we expect the basic max-min fair share model to be the first one to be supported by commercial networks, and we will use it in the rest of the discussion.

There are several ways to realize max-min fair sharing in a network. One option is to have all switches use a Fair Queueing [26] scheduler, which requires per-flow queueing and congestion control to avoid high packet drop rate. Another option is to have switches/routers explicitly compute the *max-min fair rate* for each connection and inform each source of this rate; sources are required to send no more than their max-min fair rate [14, 15]. This option is particularly attractive because explicit rates are calculated by switches/routers, and can be directly distributed as link-states and used in load sensitive routing. The ATM ABR service model uses this option. A third option is to employ Round Robin scheduling in the network with end-to-end window based congestion control algorithms [46]. The Internet will support max-min fair sharing if it employs the Round Robin scheduling algorithm in all network nodes. Hence, max-min fair share networks cover a wide range of network types.

The concept of max-min fair share was first proposed by Jaffe to address the issue of fair allocation of resources in networks based on virtual circuits (VC) or connections [50]. It has been identified by the ATM Forum [14, 10] as one of the main design goals for ABR traffic management algorithms.

2.3.3 Inter-class Sharing

When both guaranteed sessions and best-effort sessions are present in the network, the network resources are shared among sessions of different traffic classes. All (conformed) guaranteed sessions have higher priority than best-effort sessions. In other words, the best-effort sessions use the bandwidth left over from guaranteed sessions. However, when the load of guaranteed sessions is heavy and unevenly distributed, all resources available on a link may be reserved, resulting in starvation of best-effort sessions. To avoid this problem, we assume that a certain fraction of the capacity of each link is reserved for best-effort traffic. We also look at more dynamic resource sharing mechanisms in Chapter 7.

2.4 Routing Model

Many different routing mechanisms have been deployed in data networks. These protocols are characterized as being: static or dynamic routing, distance-vector or link-state routing, and hop-by-hop or source routing. In Chapter 1, we discussed and compared static routing with dynamic routing, and distance-vector routing with link-state routing. In *source routing* either the traffic source or the edge router selects the entire path. As discussed in [13], link-state source routing is particularly suitable for load-sensitive routing and makes it feasible to select paths on a per-flow basis for datagram networks where no per-flow state is maintained in the network. In ATM networks, a hierarchical link-state source routing protocol [27] is adopted.

In this work, we study both static and dynamic routing. For static routing, either link-state or distance-vector routing can be used in a straightforward way. For dynamic routing, the nature of the link state may make it difficult or impossible to use distance-vector protocols. As a result, while many of the path finding algorithms discussed in this dissertation can be implemented in either distance-vector routing or link-state routing protocols, others can only be used in link-state routing protocols.

While source routing requires paths to be selected on demand upon flow arrival, hop-by-hop routing typically selects paths periodically and maintains the path information in the routing table, so routing requests result in simple table look up. For QoS routing, since different flows with the same destination may request different QoS, more than one path may have to be selected and stored. *Class-based* routing can be deployed as an approach to reduce the number of entries in the routing table. How many classes should be used in routing table construction depends on the distribution of requested QoS parameters and the memory size of each routing/switch node. We will discuss class-based routing in Chapter 5 for routing with bandwidth guarantees.

2.5 Summary

We described the flow-based network model that will be used in our routing study. This model captures the concept of flows in both connection-oriented and connectionless data networks. A flow can be a virtual circuit as in ATM networks, a IPv6 flow, or a TCP connection. We defined four different traffic classes and their related resource sharing models. We study routing algorithms for these different traffic classes after we describe our simulation design in the next chapter.

Chapter 3

Simulation Design

While formal analytic models can be used to develop and evaluate routing algorithms at an abstract level, simulation-based evaluation allows us to consider not only diverse network topologies, service models, and traffic load distributions but also the delays in connection setup and routing information propagation. The evaluation of our QoS routing framework is based on an extensive set of simulations, through which we are able to address different performance issues for routing algorithms: call blocking rate, average throughput, distribution of achieved bandwidth, and routing information inaccuracy.

In this chapter, we discuss our simulator design in Section 3.1, the topologies used in the simulations in Section 3.2, the traffic load generation in Section 3.3, and the performance metrics used in our evaluation in Section 3.4. We summarize in Section 3.5.

3.1 Simulator

For our simulation, we developed an event-driven simulator written in C. The simulator consists of the following modules: routing algorithms, congestion control, scheduling algorithms, traffic load generation, and event processing.

Since our focus is the evaluation of the resource utilization efficiency of different routing algorithms at the session level, the simulation models the data streams at the session level. In order to accurately account for the delay associated with connection management and routing information distribution, the simulator models control signals at the packet level. These control packets are processed with high priority, so the associated delay is thus mainly processing delay and propagation delay. We assume that a small fraction of link capacity is used for the transmission of control packets. This results in a two level simulation. At the session level, it selects routes, generates new sessions, and calculates max-min fair rates. At the packet level, it manages connection setup and teardown, applies admission control, makes resource reservation, and distributes routing information.

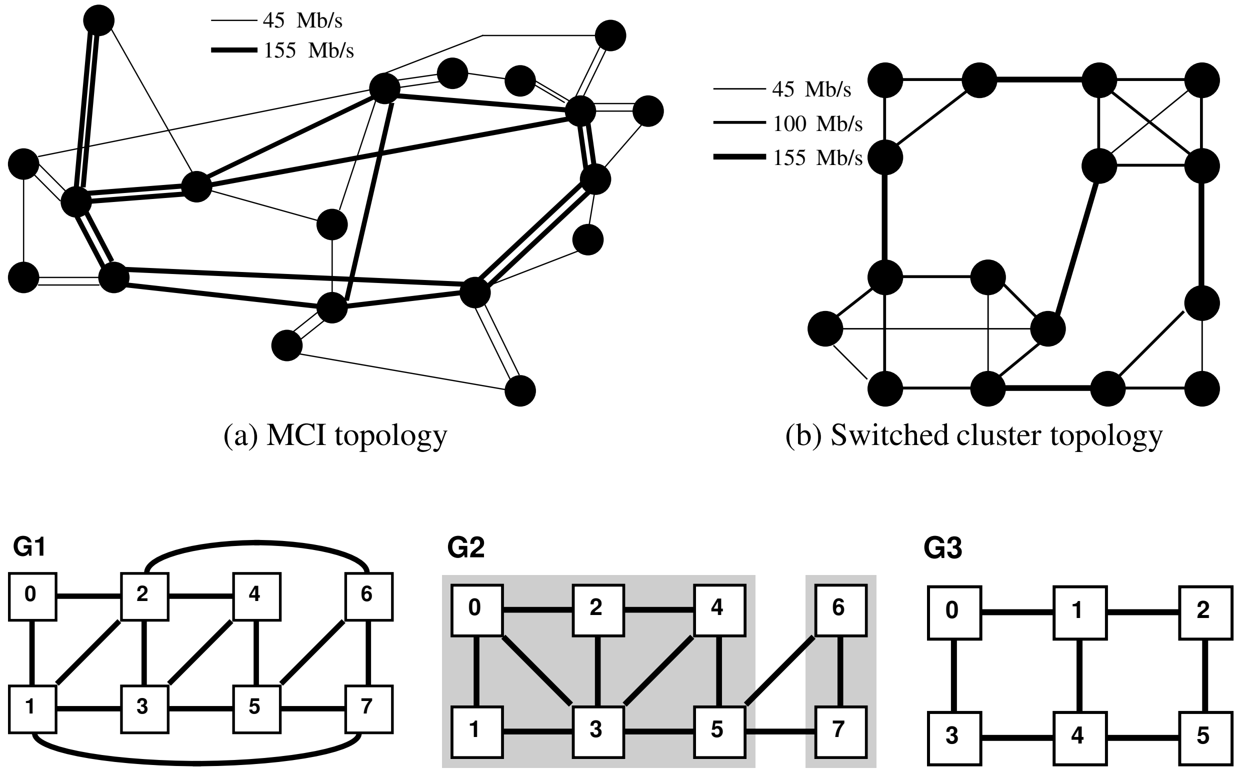
The simulator manages connections as follows. For best-effort traffic, an incoming flow specifies the number of bytes to be sent and possibly a maximum rate that the source can sustain. For guaranteed traffic, an incoming flow specifies its traffic characterists and QoS requirements (e.g., delay and bandwidth). Paths are selected by executing the routing algorithm and connections are set up; both operations can have a cost associated with them. To reflect signaling delay, we use connection set up and tear down costs of 3 ms/hop and 1 ms/hop, respectively, while the routing cost for all sessions other than low-latency sessions is 10 ms. There is no routing cost for low-latency sessions, since they use minimum-hop routes. Connections are torn down when the specified amount of data has been sent.

In a data network, accurate network state information used in selecting routes is often not available. Instead, this state information is advertised periodically and distributed among all nodes in the network. For example, the routing protocol OSPF uses a reliable flooding mechanism to periodically exchange routing information among all nodes. The network state used in our routing algorithms includes the residual bandwidth and the max-min fair share rate of each link in addition to static topology information. To understand the impact of the inaccurate routing information, we implemented a simple version of flooding in our simulator: each node updates its link state (bandwidth) at regular time intervals (default value of 30 seconds) and sends it to all its neighbors. Upon receiving a routing information update, a node updates its database and forwards the information to all its neighbors, except the one from which the update was received. Duplicate routing updates are discarded.

3.2 Topologies

A routing algorithm can behave quite differently for different network topologies. It is crucial to select appropriate network topologies in a simulation based evaluation of routing algorithms. Size, heterogeneity of link capacity, symmetry, and connectivity are four important factors to consider when selecting topologies to ensure that simulation results are as general as possible. Our focus is to understand the essence of a routing algorithm: how can a routing algorithm achieve high network throughput for various topologies and traffic load distributions?

We selected the following topologies (see Figure 3.1). One is the MCI Internet backbone topology, which is the most frequently used topology in our study. The second topology is a switch cluster, which can be viewed as an interconnected set of switch clusters in different building. Both topologies have links with heterogeneous capacity. Relatively speaking, the MCI topology is less symmetric than the switched cluster topology. We also use three other much smaller topologies with different degrees of network connectivity, which allowing us to



(a) MCI topology (b) Switched cluster topology

(c) G1 topology (d) G2 topology (e) G3 topology

Figure 3.1: Topologies used in our simulation evaluation

carry out some experiments more efficiently in terms of simulation running time. In each of the topologies, host nodes are attached to each network node with link capacities of 155 Mb/s or 622 Mb/s.

3.3 Traffic Loads

We use two classes of traffic: guaranteed sessions and best-effort sessions. The total traffic entering the network is split between these two traffic classes according to a predetermined ratio. Given a traffic load and the ratio between guaranteed sessions and best-effort sessions, the traffic load for each class can be determined.

Traffic load can be either *evenly* or *unevenly* distributed. By considering different load distributions, we are able to evaluate the performance of a routing algorithm for both scenarios when the traffic load distribution matches the network topology and when the traffic load distribution mismatches the network topology. For evenly distributed load, a new session selects any pair of nodes as its source and destination with equal probability. In the case of unevenly distributed load, a percentage of the sessions selects from a preselected set of source and destination pairs and the rest of the sessions randomly pick any pair of nodes as the source and the destination. For example, in the MCI topology, two paths with 622 Mb/s exist between the West and East Coasts. An unevenly distributed load can be generated by letting more sessions use the nodes on these two paths as their sources and the destinations. When the network topology is asymmetric, a unevenly distributed traffic pattern may match the network topology better, therefore, yielding higher network throughput.

3.3.1 Best-effort Sessions

For best-effort traffic, we follow the model used in [70]. There are two types of best-effort traffic: high-bandwidth sessions and low-latency sessions. The low latency traffic represents both the low-latency and rate-limited traffic. The balance between the two traffic types is controlled by a parameter **HBFraction**, which represents the fraction of bytes of data sent that belong to high-bandwidth connections.

Since the key performance index for low-latency traffic is per-packet end-to-end delay, we assume that all low-latency sessions are routed along the minimum-hop paths (*the widest-shortest path*, see Chapter 4 for details). Since low-latency sessions have a low peak rate, their impact on the performance of high-bandwidth sessions is small, as we will show. We thus sometimes consider scenarios in which all best-effort sessions are high-bandwidth sessions.

For each best-effort session, we assume that the number of bytes to be transmitted is known at the time when the request arrives. The number of bytes in each request is uniformly distributed over $[1\text{KByte}, \mathbf{LLvsHB}]$ for low latency traffic and $[\mathbf{LLvsHB}, 1\text{GByte}]$ for high bandwidth traffic; most of the simulations use a threshold **LLvsHB** of 1 MByte. We believe this a good approximation of longtail distribution [9, 78] of message sizes. For high-bandwidth requests, the source can make full use of the bandwidth assigned by the network. For a low-latency connection, the source specifies the maximum rate at which it can send data over the connection; this peak rate is in the range of 3 to 5 Mb/second. A session sends its data at its max-min fair share rate, and stays alive until its last byte has been sent. Given the much larger message sizes of high-bandwidth traffic, the ratio of the number of low-latency sessions to the number of high-bandwidth sessions can be very high, even if the ratio of the volume of low-latency traffic to the volume of high-bandwidth traffic is set low.

Knowing the load for best-effort sessions, the percentage of bytes to be sent in high-bandwidth traffic versus low-latency traffic, and the average message size, we can determine the mean arrival rate of best-effort sessions. Session arrival follows a Poisson distribution. The change of traffic load for best-effort sessions results in a change of the mean arrival rate. In addition to both evenly and unevenly distributed load, we also consider a client-server traffic load, which represents the case of a distributed application (clients) making heavy use of a high-performance file system (servers). In such a client-server scenario, the low-latency load is still evenly distributed, but most of the high-bandwidth load is between clients and servers. Servers are randomly selected from the pool of hosts at the start of the simulation.

3.3.2 Guaranteed Sessions

The traffic load for guaranteed traffic is determined by the distributions of the call holding time, the average data rate of traffic sources, and session arrival process. The average rate of a traffic source is the requested bandwidth for traffic requiring bandwidth guarantees and the token rate specified in the leaky bucket for traffic requiring delay guarantees.

Exponential call holding time distribution has been used in most simulation studies of real-time traffic, while others assume that all incoming calls last forever [87, 34]. Recent studies [9] show that the call holding time distribution for conversations, facsimile, and voice mail connections has a large portion of very short calls and lognormal longtail distributions. We follow the model suggested in [9]: most of our experiments use a holding time distribution that is a mixture of two normal distributions (F_1 and F_2) on a logarithmic time scale with a mixing probability α

$$F(x) = \alpha \cdot F_1(x) + (1 - \alpha) \cdot F_2(x)$$

We also occasionly use an exponential call holding time distribution with the same mean call holding time as the lognormal long-tail distribution to evaluate the impact of call holding time distribution.

Sessions requiring bandwidth guarantees can be either voice or video sessions. The requested bandwidth is uniformly distributed over [16Kb/s, 64Kb/s] for a voice session and over [1Mb/s, 5Mb/s] for a video session. The former allows us to simulate Internet voice and video phone with different quality, and the latter allows us to simulate video streams encoded by different algorithms. The ratio of voice sessions to video sessions is a simulation parameter.

For sessions requiring delay guarantees, the traffic source specifies a leaky bucket $\langle \sigma, b \rangle$, where σ is the token rate and b is the bucket size. The token rate determines the minimum amount of bandwidth that has to reserved and the bucket size restricts the burst size of the traffic allowed to enter the network. The token rate σ is uniformly distributed between $1 \sim 5$ Mb/second. The maximal bucket size b is uniformly distributed over one of the two intervals: [4KB, 8KB] and [16KB, 20KB], each may correspond to video streams compressed by different video encoders and with a different frame rate. Clearly, the larger the value of b, the burstier the guaranteed session.

Sessions arrive according to a Poisson distribution. Given a traffic load for guaranteed sessions, the mean call holding time, and the average rate of traffic sources (token rate or requested bandwidth), we can determine the mean session arrival rate.

The traffic source of a session requiring delay guarantees also needs to specify its delay bound. We assume that the end-to-end delay of a session is uniformly distributed either in the interval [80ms, 120ms] or in [200ms, 240ms]. The former is acceptable for most interactive user applications, while the latter may represent playback applications. Tighter end-to-end delay bounds require that more bandwidth is reserved.

In our simulation, we assume that a guaranteed session consumes the bandwidth it requests, although more bandwidth than the token rate may be reserved for a session with delay guarantees.

3.4 Performance Metrics

A routing algorithm performs better than others if it results in higher network throughput. For best-effort sessions, higher throughput means a higher average throughput for all the flows that have entered the network. For guaranteed sessions, higher throughput means that more flows are admitted into the network. In other words, fewer flows are blocked either by the routing algorithm or by admission control during the connection setup. We thus use two performance metrics in our evaluation: average throughput for best-effort sessions and call blocking rate for guaranteed sessions. In addition to these two metrics, we also examine the distribution of achieved bandwidth and the impact of inaccurate routing information.

3.4.1 Average Throughput

There are several alternative performance metrics that can be used to measure network throughput for best effort sessions. One of such alternatives is the *network throughput*, *i.e.*, the total number of bytes sent in a unit time interval. However, in routing studies we often assume that the network load is fixed, and the number of bytes entering the network during a given time interval is therefor also fixed. The network throughput averaged over a long time interval will be similar for different routing algorithms, although the average time for completing individual sessions can be significantly different.

An alternative is the *average per-session throughput*, *i.e.*, the sum of the throughputs achieved by all completed sessions divided by the number of completed sessions. The potential problem with this metric is that different numbers of bytes can be sent by different sessions and it encourages a routing algorithm to optimize the throughput for the sessions that send small messages.

In our study, we use the *average throughput* that takes the weighted harmonic mean of the average per-session throughput [51], using the message length as the weight:

$$\text{average throughput} = \frac{\sum_{i \in N} b_i}{\sum_{i \in N} t_i} \tag{3.1}$$

where N is the total number of sessions in the network, b_i represents the number of bytes sent over connection i, and t_i its duration. Since the total number of bytes $\sum_{i \in N} b_i$ sent over the network is fixed for a long time interval, the average throughput can also be viewed as a measure of the elapsed time experienced by all high-bandwidth connections.

However, the average throughput is only part of the performance picture. Another important parameter is how achieved throughput is distributed. In general, having all session throughputs fall in a small range is preferable over having a wide variance. This is especially true in a max-min fair share network that tries to balance the throughput of connections sharing links. For this reason, we will also discuss the actual throughput distribution for a few typical scenarios in more detail.

One frequently used performance index in the evaluation of scheduling, admission control, and congestion control algorithms is the *link utilization*. However, for the study of routing algorithms, higher link utilization does not imply that a routing algorithm is superior. Instead, it can encourage routing algorithms to select paths with more hops.

3.4.2 Blocking Rate

A guaranteed session can be rejected either because no path with sufficient resources can be found by the routing algorithm or because the resource availability on the selected path has changed since the time when the routing decision was made. The latter is caused either by the delay of hop-by-hop connection setup or by inaccurate routing information. Given a traffic load for guaranteed sessions, the network throughput is determined by the number of sessions admitted. Hence, the *call blocking rate* is a good performance metric for guaranteed sessions.

$$\text{call blocking rate} = \frac{\text{total number of rejected guaranteed sessions}}{\text{total number of arrivals of guaranteed sessions}}.$$

The call blocking rate has been widely used in routing studies for circuit-switched telephone networks [35, 45].

It is clear that, if all sessions are equal, a lower call blocking rate implies more guaranteed sessions admitted into the network. However, when sessions can request different amounts of bandwidth, a low call blocking rate does not necessarily reflect high efficiency. Thus, when evaluating algorithms for sessions requiring bandwidth guarantees, we introduce the *bandwidth blocking rate*, which takes into account the bandwidth of rejected sessions:

$$\text{bandwidth blocking rate} = \frac{\sum_{i \in \text{BG\_blk}} \text{bandwidth}(i)}{\sum_{i \in \text{BG}} \text{bandwidth}(i)}$$

where BG_blk is the set of blocked sessions and BG is the set of requests for guaranteed sessions. We also evaluate the fairness of the routing algorithm: how are sessions requesting different amounts of bandwidth treated?

Using the bandwidth blocking rate alone in evaluating a routing algorithm for guaranteed sessions can be biased or even misleading. Applying different routing algorithms for guaranteed sessions may have a different impact on the performance of best-effort sessions even when the call blocking rate for guaranteed sessions are the same. For example, when the call blocking rate for all routing algorithms is close to 0, the resource efficiency of the routing algorithm for guaranteed sessions is mainly determined by the performance of best-effort sessions.

3.4.3 Routing Inaccuracy

As a result of routing information distribution delay and connection setup delays, routing algorithms can generate an incorrect result. For best-effort sessions, inaccurate routing information may lead a routing algorithm to select paths that have low average throughput. We will evaluate this possible performance degradation by increasing the routing update interval. For guaranteed sessions, with inaccurate routing information, either a path with insufficient bandwidth may be selected or no feasible path may be found although one exists. To capture this aspect of routing algorithm performance, we define the *routing inaccuracy* metric:

$$\text{routing inaccuracy} = \frac{\text{number of incorrect route selections}}{\text{total number of session requests}}$$

Misrouted sessions include both those routed but rejected by the admission control, and those not routed while feasible paths actually exist. We will also evaluate whether inaccurate routing information can cause a performance degradation for guaranteed sessions.

3.5 Summary

In this section, we discussed the design of our simulation experiments. The simulator supports multiple classes of traffic, connection management, congestion control, routing information distribution, and many different routing algorithms. It works with different topologies, and supports various traffic load distributions. We defined performance metrics for the traffic classes that will be used in the rest of this dissertation.

Chapter 4

Routing Best-Effort Traffic

Best-effort traffic has been and will continue to be the most dominant traffic class in the Internet, although other traffic classes with QoS guarantees are being introduced. Traditional best-effort traffic, such as electronic mail, telnet, and RPC, has been mostly small messages with a payload typically less than a few tens of kilobytes of data. Users would like to have their messages arrive at their destination as quickly as possible. For this kind of low-latency traffic, it has been shown [101] that the minimum-hop routing, *i.e.*, packets are sent along the path with the minimum number of hops, may work well when the path is not congested. Alternative paths should be selected dynamically during periods of congestion.

Applications, such as I/O intensive scientific computation and the retrieval of large images, are moving increasingly larger data sets from a few megabytes to several gigabytes in a bursty fashion. In Chapter 2, we distinguished between this type of high-bandwidth traffic and traditional low-latency traffic and suggested that the network should provide explicit support for high-bandwidth traffic.

In this chapter, we study how to route high-bandwidth traffic in a max-min fair share network. We show how routing can be used to improve the performance of high bandwidth applications. While it is possible to devise priority or reservation-based schemes that give high-bandwidth traffic preferential treatment at the expense of lower performance for other sessions, we focus on designing routing algorithms that optimize the average application throughput, while treating all applications equally.

Our routing algorithms are closely coupled to the congestion control algorithm by taking advantage of the congestion state information, *i.e.*, the max-min fair share rate, to perform load-sensitive routing on a per-connection basis. Linking the two resource allocation mechanisms makes it possible to do effective load-sensitive routing. We propose an abstraction for the congestion state information to be used by the routing algorithm. Using an extensive set of simulation results, we identify a link cost for "shortest-path" routing that performs uniformly better than the traditional minimal-hop routing and shortest-widest path routing. To achieve even higher throughput without reducing the fair shares of single-path connections, we propose a novel prioritized multi-path routing scheme, in which each path is assigned a unique priority. This leads to a conservative extension of max-min fairness to multi-level prioritized max-min fairness, in which lower priority paths share the bandwidth left unused by higher priority paths. Simulation results confirm the validity of our multi-path routing scheme.

The remainder of this chapter is organized as follows. In Section 4.1 we discuss the characteristics of high-bandwidth traffic. Section 4.2 we describe our approach. We develop the single path and multi-path routing algorithms in Sections 4.3 and 4.4, respectively. Sections 4.5 through 4.7 present our simulation results. In Section 4.8 we discuss related work and we summarize in Section 4.9.

4.1 High-Bandwidth Traffic

Two traffic characteristics are associated with high-bandwidth traffic. One is its large message size compared with low-latency traffic. The other is a high source rate compared with constrained flows, such as voice and video streams, that have an inherent bound on their throughput. In other words, a source can send data as fast as the network can consume it. The key performance index for data traffic is the **elapsed time**, i.e. the period from the time when the application issues the transfer command to the time when the transfer completes. The elapsed time E_i for a flow F_i consists of the following terms:

$$E_i = P_i + D_i + \frac{b_i}{r_i} \tag{4.1}$$

where P_i is the connection establishment time, D_i is the end-to-end packet delay, b_i is the size of the data transfer, and r_i is the average rate.

For high-bandwidth applications, the message size b_i is very large, so the elapsed time E_i is dominated by the last term. Since b_i is a constant, we minimize E_i by maximizing the average rate r_i. One way of increasing r_i is to give session i a higher priority, larger service share, or reserved bandwidth. All these mechanisms give session i preferential treatment at the expense of other sessions, and they require external administrative or pricing policies to function properly. In contrast, we focus on max-min fair networks, where all traffic streams are treated equally by the traffic management algorithm. It is still possible to enhance the performance of high-bandwidth traffic streams by routing them along paths that will yield, on average, higher rates. This requires that the routing algorithm can estimate the rate that new flows will get if they are routed along a specific path.

4.2 Approach

To estimate the expected rate for a new flow along any path, the max-min fair rate information of the network should be made available to the routing algorithm. The goal of our routing algorithms is to select paths that achieve high average throughput. In this section, we first discuss how to obtain and represent the rate information inside the network. We then discuss our approach to path selection.

4.2.1 Max-min Fair Rates as Link State

In a max-min fair share network, the rate r_i for a flow F_i in Equation (4.1) is the average max-min fair share rate that F_i can obtain. The max-min fair rate is an indicator of link congestion conditions and has been used as an explicit rate in rate-based congestion control. By using the max-min rate information in our routing algorithms, it possible to do effective load-sensitive routing.

The highest rate that a flow can achieve in a max-min fair network is called its *max-min fair rate*, and it can be calculated step by step by increasing all flows rate equally and removing saturated flows and bottleneck links. A critical step to making max-min fair sharing practical in real networks is to develop efficient distributed asynchronous algorithms. We refer readers to [14] for more discussions on rate calculation algorithms and their implementations.

We now describe a simple centralized algorithm to calculate the max-min fair rates or the saturation rates of all connections [7]. A connection is called saturated if it has reached its desired source rate or a link on the path traversed by the connection is saturated. A link is called saturated if all of its bandwidth has been allocated to connections sharing the link. Let CN be the set of all connections in the network, CN_l the set of connections using link l, and SAT and UNSAT the set of saturated and unsaturated connections, respectively. Let SAT_l be the set $\mathsf{CN}_l \cap \mathsf{SAT}$, and UNSAT_l the set $\mathsf{CN}_l \cap \mathsf{UNSAT}$. Given a set S of connections, let $\mathsf{L}(\mathsf{S})$ be the set of all links in the network with at least one connection in S using them. Let C_l be the capacity of link l. The algorithm can be described as follows:

1. Initialization: $\mathsf{SAT} = \emptyset$, and $\mathsf{UNSAT} = \mathsf{CN}$.

2. Iteration: Repeat the following steps until UNSAT becomes $\emptyset$

   - For every link $l \in \mathsf{L} = \mathsf{L}(\mathsf{UNSAT})$, calculate

$$\mathrm{inc}_l = \frac{C_l - \sum_{i \in \mathsf{SAT}_l} r_i}{|\mathsf{UNSAT}_l|} \tag{4.2}$$

- Get the minimum: $\text{min\_inc} = \min\{\text{inc}_l \mid l \in \mathsf{L}\}$.
- Update rate r_i: $r_i = r_i + \text{min\_inc}$.
- Move new saturated connections from UNSAT to SAT.

Note that the max-min fair rate of a connection is a function of time, it can change when connections arrive or terminate.

Using the max-min fair rate as a cost function is unique. The routing algorithm used in most networks tries to minimize the number of hops (link cost is 1), or, for load sensitive routing, the end-to-end delay (link cost is packet delay). Neither cost function is necessarily a good predictor of available bandwidth. A common link cost function in reservation-based networks is the residual link bandwidth — unreserved bandwidth. We cannot use residual bandwidth since the nature of bandwidth sharing in reservation-based and max-min fair share networks is very different. In contrast, an estimate of max-min fair rate that accounts for the nature of bandwidth sharing is an accurate load-sensitive predictor of the bandwidth available to a new connection.

4.2.2 Link-state Representation

In order to consider congestion conditions, the routing algorithm needs to know the expected max-min rate r_i for a new connection at each link. The calculation of r_i requires access to rate information for all connections associated with this link. Since the number of connections can be very large, the volume of rate information can be large as well. To address this problem, we introduce a concise data structure to approximate this rate information, and propose an approximate algorithm to calculate r_i.

For each link, instead of having a separate rate entry for each connection, we use a fixed number of discrete rate intervals. The rate information is simply represented by the number of connections with a max-min fair rate in each interval. The size of this representation scales with the number of rate intervals, but it is independent of, and therefore scales well with, the number of connections sharing the link.

Let us look at an example. For each interval i, let `rate_scal[i]` be the middle value of the interval, and `num_conn[i]` the number of connections in the interval. For a link of 155 Mb/s, rate information of a link can be represented, for example, by a vector of 64 entries with a scale function defined by

$$\texttt{rate\_scal}(i) = \begin{cases} 0.5 * i & \text{if } 0 \leq i < 16 \\ 1.0 * i - 8 & \text{if } 16 \leq i < 32 \\ 1.5 * i - 24 & \text{if } 32 \leq i < 48 \\ 2.0 * i - 48 & \text{if } 48 \leq i < 64 \end{cases} \tag{4.3}$$

We used multiple scales in this example to ensure reasonable accuracy for connections with low rates while restricting the size of the vector. The max-min fair rate for a new connection can now be estimated by the following procedure

```
New_Rate(float rate_scale[], int num_conn[], int num_conn)
{
    int i, N, tmp_num_below_ave, num_below_ave = 0;
    float rate, rate_below_ave = 0.0;

    N = num_conn + 1;
    do {
        rate = (C - rate_below_ave) / (N - num_below_ave);
        tmp_num_below_ave = 0;
        while (rate_scale[i] < rate) {
            rate_below_ave += num_conn[i] * rate_scale[i];
            tmp_num_below_ave += num_conn[i];
            i++;
        }
        num_below_ave += tmp_num_below_ave;
    } while (tmp_num_below_ave > 0)
    return rate;
}
```

It is important to realize that both the rate representation and the algorithm that calculates the max-min fair rate of a link are approximations. Even if we know the precise max-min fair rate for all flows that use the link, the max-min fair rate calculated by the procedure `New_Rate()` is still an estimate. For the purpose of routing, this estimate should be sufficient, because the routing information available is usually not accurate and the max-min fair rate will change over time.

The procedure `New_Rate()` is invoked by network nodes to calculate the link state for all their output links. To avoid abrupt changes in the link state, which may potentially cause oscillation, the technique of exponential averaging has been suggested in the literature. That is, for some chosen value of α,

$$\text{rate} = \alpha * \text{rate}_{\text{old}} + (1 - \alpha) * \text{rate}_{\text{new}}$$

where rate_{old} is the link state advertised in the previous round and rate_{new} is the newly calculated rate by the procedure `New_Rate()`. However, we observed in our experiments that selecting a value for α other than 0 can slow down the routing algorithm's response to changes in the network state and can lower average throughput. In the rest of the discussion, we use $\alpha = 0$.

4.2.3 Achieve Efficiency: Balance Path Length and Width

There are two main approaches to achieving high resource utilization. One is to conserve network resources, the other is to balance the network load. To conserve the network resources, a path with the minimum hop count should be preferred over other paths. To balance the network load, a path with the least load should be preferred.

The max-min fair rate of a path is a good indication of the path congestion condition as well as the throughput achievable by a flow that uses the path. When trying to maximize bandwidth, it seems natural to pick the "widest path" algorithm, i.e. to select the path with the highest current max-min fair rate. This is however not necessarily the best link cost. The key observation is that the max-min fair rate changes over time, and the high rate available at connection establishment time may not be sustained throughout the lifetime of the connection. Since the fair rate of a connection is the minimum of the rates available on each link, the chance that the fair rate will go down increases with the number of hops. Moreover, longer paths consume more resources, which may reduce the rate available for future connections. When the network load is heavy, more traffic arrives in a fixed interval, and the "widest path" path may have to share resources with more future connections.

Simply using a minimal-hop path in order to conserve network resources is not always a good choice either. For example, when the traffic load is concentrated or when the network topology is less symmetric, minimal-hop paths can be very congested, and the network resources may not be efficiently used because other less congested paths may be available but unused. Thus, restricting the hop count of paths being selected does not balance the network load adequately.

Hence, there is a need to balance path length and width when selecting a path. If the network load is heavy, a path with fewer hops should be selected and if the network is light, a least loaded path should be selected. To achieve the goal, we must define the link cost function in such a way that a congested link has a high cost and at the same time there is a penalty for a using a long path.

4.3 Single Path Routing

For single path routing, the key is to select a link cost function that conserves network resources when the network load is heavy and balances network load when the network load is light. We want the path selection algorithm to do this automatically without requiring a manually specified threshold that distinguishes between heavy and light loads.

Towards this goal, we define a family of polynomial link costs, $(\frac{1}{r})^n$, where r is the current max-min rate for a new connection (see [61] for the use of polynomial costs in solving optimal graph cut problem). By changing n, we can cover the spectrum between shortest ($n = 0$, or minimum-hop) and widest ($n \to \infty$) path algorithms. In the remainder of this paper we will consider the following five algorithms:

- **Widest-shortest path**: a path with the minimum number of hops. If there are several such paths, the one with the maximum max-min fair rate is selected.
- **Shortest-widest path**: a path with the maximum max-min rate. If there are several such paths, the one with the fewest hops is selected.
- **Shortest-**dist(P, n): a path with the shortest distance

$$\text{dist}(P, n) = \sum_{i=1}^{k} \frac{1}{r_i^n}$$

  where $r_1, \cdots, r_k$ are the max-min fair rates of links on the path P with k hops. We will consider three cases corresponding to $n = 0.5$, 1, and 2.

An interesting point is that dist$(P, 1)$ can be interpreted as the bit transmission delay from the traffic source to the destination should the connection get the rate r_i at hop i. This delay is different from the measured delay used in traditional shortest delay paths in two ways. First, the focus is on bit transmission delay (i.e. bandwidth) instead of total packet delay. Second, the measure is for data belonging to a specific connection instead of a link average.

4.4 Multi-path Routing

It is often the case that resource on some links may remain unused. One technique to increase the average throughput of a high-bandwidth traffic session is to use multiple parallel paths to transfer the data. Each path is realized using a single network-level connection. However, simply having high bandwidth applications use multiple paths is not acceptable since "max-min fairness" is implemented on the basis of network-level connections and a session with multiple paths (i.e. network-level connections) will achieve a higher performance *at the expense of* sessions using only one path. Actually, applications could increase their bandwidth almost arbitrarily by using more paths.

4.4.1 Prioritized Multi-level Max-min Fairness

To take advantage of the higher throughput offered by multiple paths, without violating the fairness property, we propose a prioritized multi-level max-min fair share model. In this model, connections are assigned different priorities. If a session has N paths, the n^{th} path,

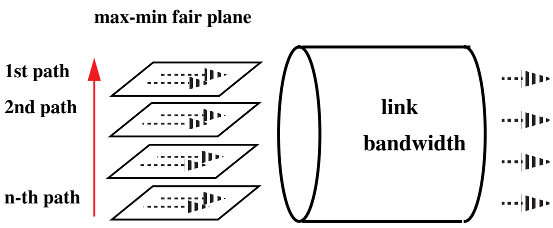


Figure 4.1: Prioritized multi-level max-min fair share

$n = 1, \cdots, N$, is assigned to the n^{th} priority level. The number of priority levels in the network determines the maximum number of parallel paths one session can have. For a network with N priority levels (Figure 4.1), N rounds of max-min fair share rate computations are performed. In the n^{th} round, the algorithm computes the max-min fair share rate for all the connections at level n using the residual link bandwidth left unused by the higher priority connections, that is,

$$\mathsf{inc}_l = \frac{(C_l - \sum_{k<n} R^k) - \sum_{i \in \mathsf{sat}_l^n} r_i^n}{|\mathsf{unsat}_l^n|} \tag{4.4}$$

where R_k is the sum of the rates of all connections with priority k.

This model has two interesting features. First, the multi-level fair share model is consistent with the single-level fair share model in the sense that the max-min fair rate for sessions using one path will not be reduced by the presence of multi-path sessions. Second, the priority levels are used in the max-min rate computation, but they do not necessarily have to be directly supported or even be visible by schedulers on switches and sources. For example, the rate-based congestion control adopted by ATM forum can easily be extended to prioritized multi-level fair sharing. The changes needed are the assignment of a priority to each connection and the use of a different algorithm for the fair rate calculation. The schedulers on the sources do not have to be changed: they continue to enforce the explicit rate assigned to them by the network. Similarly, switches can continue to use FIFO scheduling.

4.4.2 Prioritized Multi-path Routing

The multi-path routing algorithm based on prioritized multi-level fair sharing simply repeats a single-path routing algorithm at each level of max-min fair sharing, starting with the highest priority. At each level, the algorithm only uses bandwidth left unused by paths at the higher priority level. Note that this means that links that are saturated at a certain level will not be present in the network topology used at lower levels. The algorithm terminates when either paths with sufficient bandwidth have been found or no more new paths with nonzero bandwidth can be found.

Finally, striping data over parallel paths is likely to introduce some overhead on the sending and receiving host, for example to deal with out of order packet arrival. Multi-path routing should therefore only be used if the expect increase in bandwidth is above a certain threshold.

4.5 Simulation Results for Single-path Routing

We examine the performance of the five routing algorithms discussed in Section 4.3. Through a simulation-based evaluation, we answer the following questions: How do the different routing algorithms perform with different network load, different traffic distribution, different network topologies, and different ratios between high-bandwidth traffic and low-latency traffic? What is the distribution of achieved throughput? How sensitive is the performance of these algorithms to inaccurate routing information?

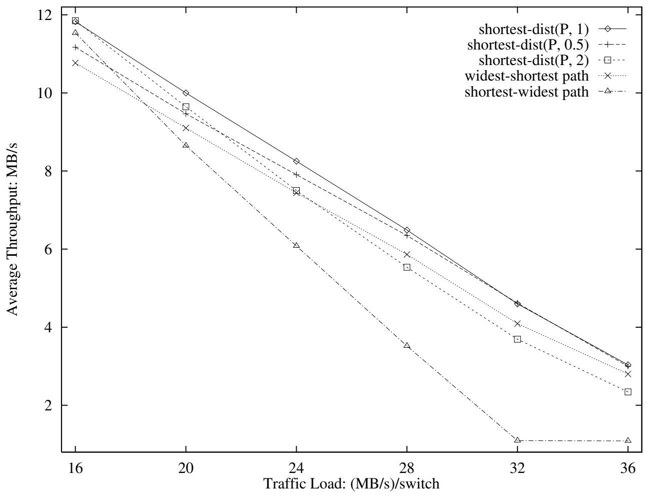
Initially our evaluation assumes that the accurate routing information is available. We then examine the impact of routing information distribution delay in Section 4.5.4.

4.5.1 Average Throughput as a Function of Traffic Load

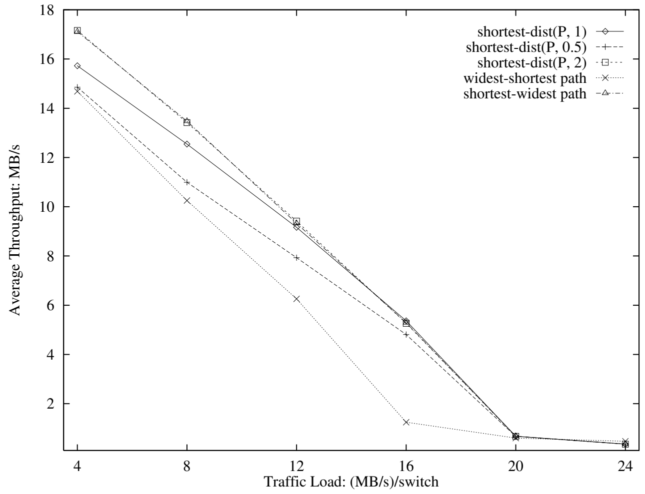
We examine the average throughput achieved by high-bandwidth connections as a function of the aggregate traffic arrival rate from all hosts connected to a switch. The traffic load is uniformly distributed with 90% of the bytes traveling over high-bandwidth connections. We assume that accurate routing information is available.

Figure 4.2 shows the average throughput for topologies G1 and G2. We can distinguish three phases corresponding to low, medium, and high traffic loads. We first focus on topology G1. When the load is low, all algorithms give fairly similar performance, although "greedy" users who use the Shortest-widest, Shortest-dist($P, 2$), or Shortest-dist($P, 1$) path algorithm achieve slightly higher throughput. This result matches our intuition: when the network is lightly loaded, we expect all algorithms to perform well, but greedy algorithms are likely to have an edge.

As the network load increases, the average per-connection throughput decreases. The greedy shortest-widest path algorithm has the biggest drop in performance, while the Shortest-dist($P, 0.5$) and widest-shortest path algorithms, which place more emphasis on finding a short path rather than a wide path, exhibit the slowest decrease in average throughput. The intuition is that with a higher network load, resources become more scarce, and algorithms that tend to pick longer paths (i.e. attach less value to a small hop count) perform more poorly. While



(a) Topology G1



(b) Topology G2

Figure 4.2: 90% bytes in high-bandwidth traffic

a greedy algorithm might be able to increase the throughput of an individual connection by picking a long path with higher bandwidth, this might reduce the throughput of many other connections, and thus the average throughput. Moreover, paths with more hops have a higher chance of having their throughput reduced as a result of fair sharing with connections that are added later. The Shortest-dist$(P, 1)$ outperforms all the other algorithms.

Further increases in network load reduce the difference in performance achieved by different algorithms. The reason is that under high load, all links are likely to be congested, so path selection becomes less sensitive to the obtainable rate and most algorithms tend to pick widest-shortest paths.

While the curves for the other topologies have a similar shape, there are some interesting differences. Topology $G2$ is less symmetric and has a lower degree of connectivity than topology $G1$. Figure 4.2(b) shows that as a result, greedy algorithms perform consistently better than algorithms that attach more weight to minimizing the number of hops. For example, the widest-shortest path, which performed well on topology $G1$, has very poor performance, and the shortest-widest and Shortest-dist$(P, 2)$ path algorithms, which performed poorly on topology $G1$, give the best performance. This difference is a result of the unbalanced nature of topology $G2$: to make good use of the links connected to switch node 5 it is important to attach a lot of weight to the width of the path so that the bottleneck link can be avoided (link 3-5). The Shortest-dist$(P, 1)$ path algorithm continues to perform well, e.g., it outperforms the widest-shortest path algorithm by as much as a factor of 4, while its throughput is only 9$\%$ or less lower than with the shortest-widest paths.

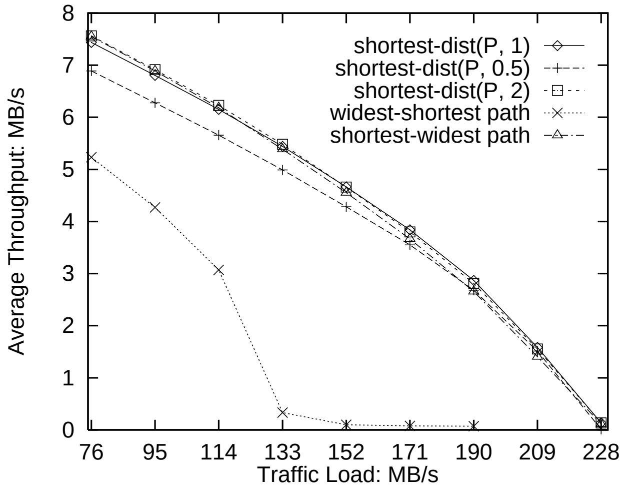
Both G1 and G2 are small topologies with 8 network nodes. We now examine the performance of the routing algorithms for larger topologies. Figure 4.3 shows the average throughput for the MCI and cluster topologies when the traffic load is uniformly distributed and we assume that all connections in the network are high-bandwidth sessions.

For the MCI topology, we see that all algorithms except the widest-shortest path perform very similarly. The poor performance of the widest-shortest path is caused by the asymmetric topology and the heterogeneous link capacity. Since the traffic load is evenly distributed, the widest-shortest paths for some source-destination pairs have very limited capacity, and these paths become congested easily. Note that a widest-shortest path is always a minimal hop path, so there is only limited flexibility to balance the network load and to avoid congested links.

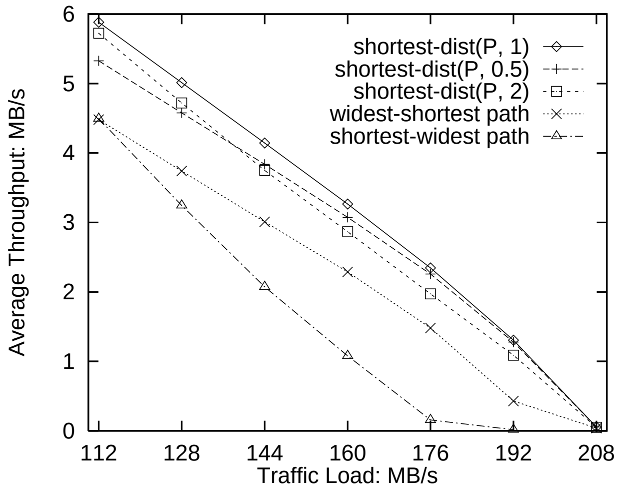
Compared with the MCI topology, the cluster topology is more symmetric. We observe that the shortest-widest path results in very poor performance. We see again that selecting a path with more hops but higher max-min fair rate at routing time can lead to consuming more network resources without necessarily offering much benefit. We also see that the widest-shortest path does not perform well because of its limited flexibility in balancing the network load. All three polynomial based distance functions exhibit similarly good performance, although the Shortest-dist$(P, 1)$ slightly outperforms the other two.

When the network load is unevenly distributed, load balancing becomes more important. Figure 4.4 shows that, for the MCI topology and unevenly distributed traffic load, the widest-shortest path performs poorly compared with the other routing algorithms. We also see that the Shortest-dist$(P, 0.5)$ path no longer performs as well as the Shortest-dist$(P, 1)$ path compared with the case when the load is evenly distributed, because it tends to select a path with fewer hops and does not avoid bottlenecks as well.

In summary, conserving network resources and balancing network load are the two important



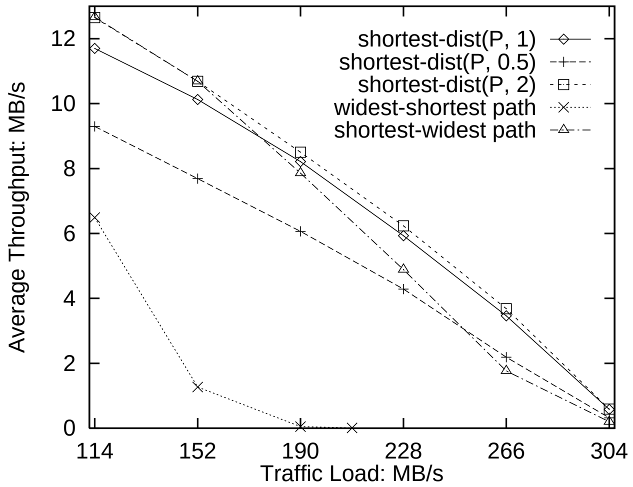
(a) MCI topology



(b) Cluster topology

Figure 4.3: 100% bytes in high-bandwidth traffic and evenly distributed load

mechanisms to achieve high resource utilization. For a max-min fair share network, an algorithm performs well if it pays attention to both conserving resources and balancing load. An algorithm that only pays attention to conserving resources by optimizing hop count can not avoid congested links. Using the widest path to ensure full load balancing consumes more network resources and can therefore reduce the throughput of the network. The shortest-widest path may work well when the network load is light, but performs poorly when the network load is heavy. The widest-shortest path, on the other hand can not avoid congested links and does not perform



(a) MCI topology

Figure 4.4: 100% bytes in high-bandwidth traffic and unevenly distributed load

well, especially when the network load is unevenly distributed. By using the polynomial link metrics, the path length and width can be balanced automatically according to the current load distribution: It tends to select a minimum-hop path when the load is heavy and a widest path when the load is light. This results in high performance regardless of the network topology and traffic load distribution. Among the three polynomial link metrics, the Shortest-dist$(P, 1)$ path algorithm performs consistently well across the different topologies when accurate routing information is available.

4.5.2 Impact of high-bandwidth traffic volume

We examine the effect of changing the distribution of traffic between high-bandwidth and low-latency connections. In Figure 4.5, the ratio of high-bandwidth traffic is reduced to 50%, compared to 90% in Figure 4.2(a). We observe that the results are similar, although the performance of the shortest-widest path is somewhat better. The Shortest-dist$(P, 1)$ path algorithm still outperforms the other algorithms.

Figure 4.6 shows the average throughput as a function of the percentage of high-bandwidth traffic, for a fixed traffic load of 24 MB/s per switch. We see that as the contribution of high bandwidth traffic increases, the choice of routing algorithm used for high-bandwidth traffic has more impact, although even with only 10% high-bandwidth traffic, the best algorithm (widest shortest) still gives a 20% higher throughput than the worst algorithm (shortest widest).

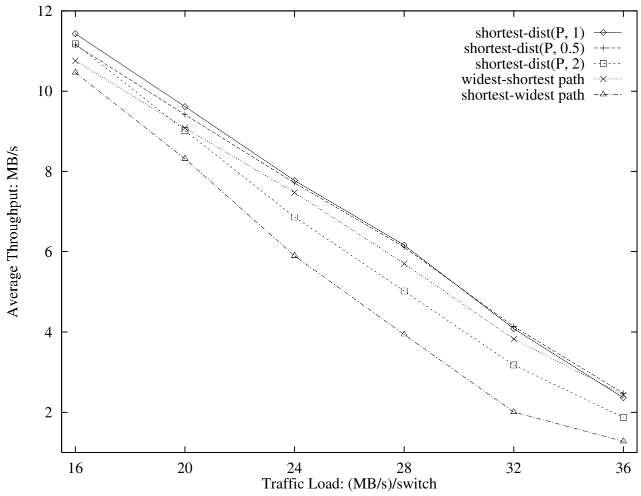


Figure 4.5: $G1$: 50% bytes in high-bandwidth traffic

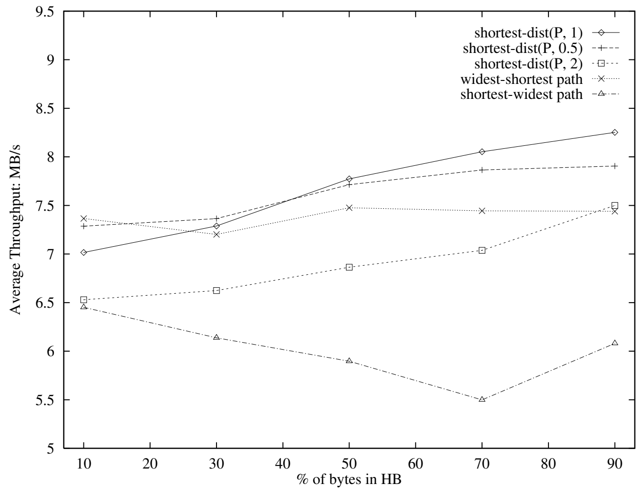
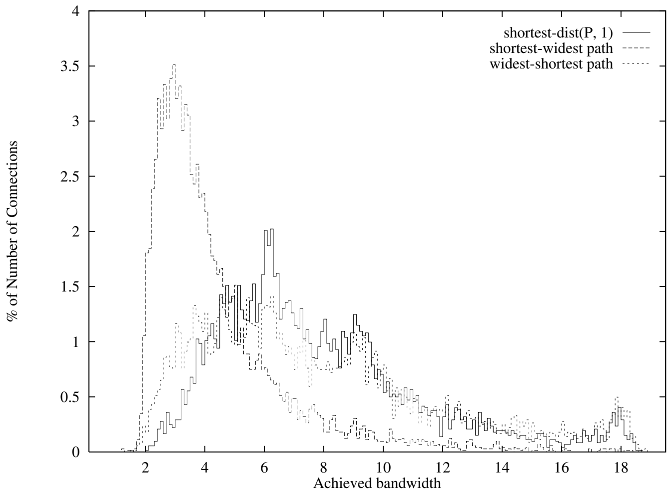


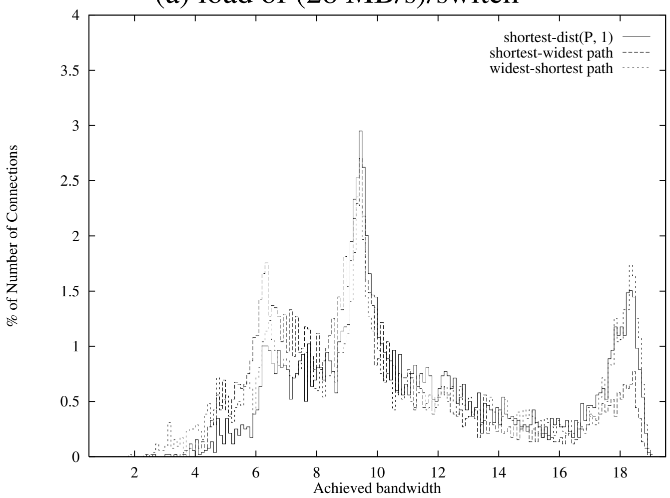
Figure 4.6: $G1$: arrival rate 24 MB/s per switch

4.5.3 Variability of per-connection throughput

So far we have focused on the average throughput obtained by high-bandwidth connections. In this section we look at how the routing algorithm influences the throughput variability. Note that since the shape of the throughput distribution is often uneven and influenced by the topology, measures such as variance are not meaningful, so we exam the actual distribution.



(a) load of (28 MB/s)/switch



(b) load of (20 MB/s)/switch

Figure 4.7: $G1$: 90% bytes in HB

In Figure 4.7, we present the throughput distribution of high-bandwidth connections for the shortest-widest path, widest-shortest path, and Shortest-dist$(P, 1)$ path algorithms. The results are for topology $G1$ with 90% high-bandwidth traffic and for two traffic loads: 28 MB/s and 20 MB/s per switch (compare with Figure 4.2(a)).

For higher loads, the throughput distribution for shortest-widest paths has a peak around 3 MB/s and a long tail corresponding to connections that achieve high throughput. With the the widest-shortest path and Shortest-dist$(P, 1)$ path algorithms, the throughput is more evenly distributed between 3 to 9 MB/s, with a tail of higher throughput. With the shortest-widest

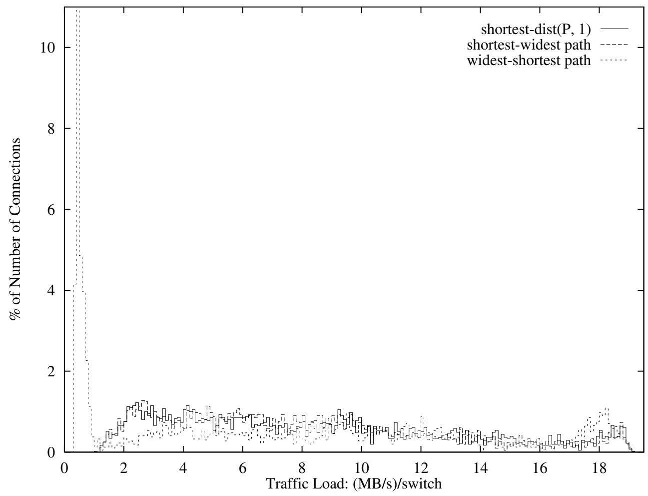


Figure 4.8: $G2$: 90% bytes in HB and (16 MB/s)/switch

path algorithm, few connections are able to get a high throughput because paths with more hops have a higher chance of having to share bandwidth with many other connections. This can be seen from Table 4.1, where we break down all connections according to the number of hops in their paths and show the average throughput, average initial rate, and distribution of connections with different hop counts. We see that the ratio of the average throughput over the average initial rate decreases as the path length increases.

hops	3	4	5	6	algorithms
throughput	4.4	3.3	2.9	2.7	shortest-widest
AveInitRate	4.2	3.3	3.1	3.6	
throughput/AveInitRate	1.05	1.00	0.94	0.75	
connections	30%	36%	23%	9%	
throughput	7.4	6.1	5.8	5.4	Shortest-dist(P,1)
AveInitRate	6.7	6.1	6.5	7	
throughput/AveInitRate	1.10	1.00	0.89	0.77	
connections	46%	41%	12%	1%	

Table 4.1: Average throughput, average initial rate, and % of connections with different hops

When the network load is lower (20 MB/s per-switch in Figure 4.7(b)), the throughput distributions for the different algorithms are more similar. All three loads have approximately a bimodal distribution, which is a result of the network topology. Using the Widest-shortest and Shortest-dist$(P, 1)$ path algorithms increases the chance of achieving very high throughput.

Figure 4.8 shows the throughput distribution for topology $G2$, with a traffic load of $(16\text{MB}/\text{s})$/switch and 90% high-bandwidth traffic (compare with Figure 4.2(middle)). It shows why the widest-shortest path algorithm performs poorly: many widest-shortest paths use the link between switches three and five, resulting in a bottleneck and low throughput $(0.5\text{MB}/\text{s})$. The other two algorithms can avoid the bottleneck and have more evenly distributed throughputs.

4.5.4 Impact of Inaccurate Routing Information

In the previous evaluation, we assumed that the routing algorithms have access to accurate network state information. However, because of delays in routing information distribution, the available routing information is often not accurate. We evaluate the impact of routing information update intervals on performance in this section.

In Figure 4.9, we show a scenario with the same traffic condition and topology as Figure 4.2 (a), but with a 100ms routing information update interval, i.e., routing information is usually somewhat dated. The two figures are very similar, although we observe slightly lower performance with a 100ms routing update interval, especially for the shortest-widest path algorithm. This suggests that greedy algorithms might be more sensitive to outdated information.

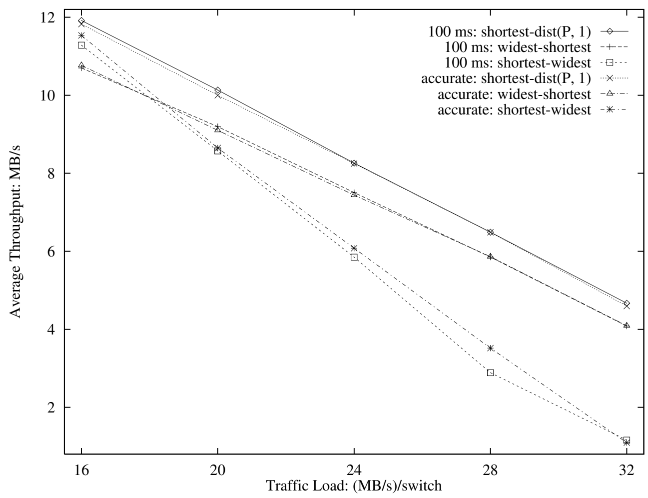
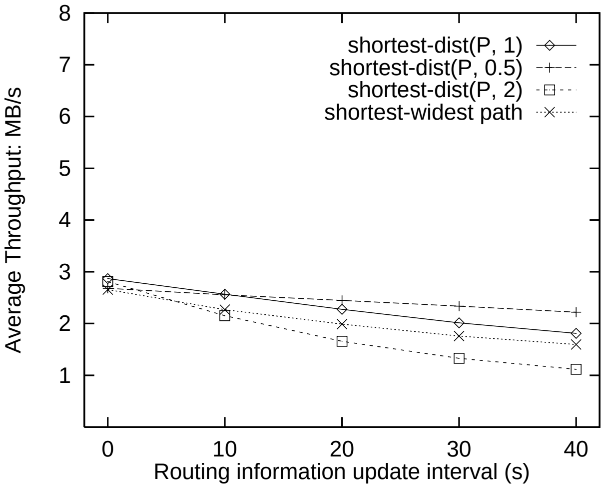
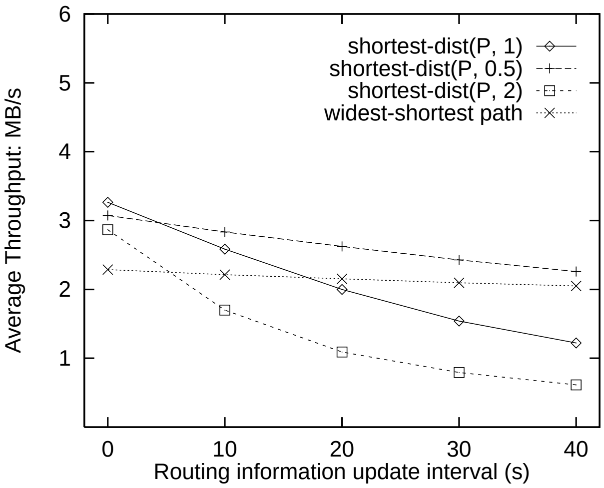


Figure 4.9: G1: 90% bytes in HB traffic and different routing update interval



(a) MCI topology (190 MB/s)

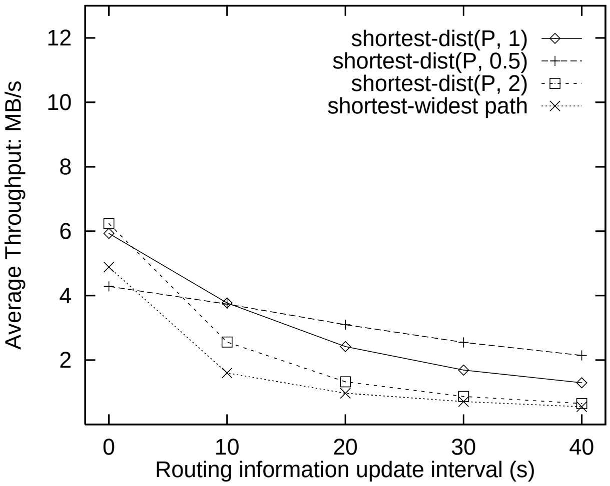


(b) Cluster topology (160 MB/s)

Figure 4.10: Average throughput as a function of the routing information update interval: 100% bytes in high-bandwidth traffic and even load

The routing information update interval used in Figure 4.9 is 100ms, which is small. In Figure 4.10 and 4.11, we use larger update intervals and show the impact on the average throughput as the routing information update interval changes. All curves have been plotted with the same scale for y-axis as in Figures 4.3 and 4.4 for the purpose of comparison. We focus on algorithms that proved to be at least somewhat competitive in Section 4.5.1.

Figures 4.10 and 4.11 show that the performance of algorithms that lead to paths with



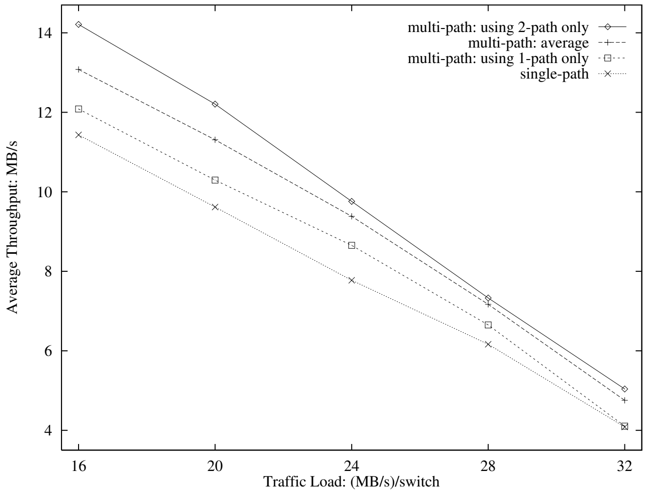
MCI topology (228 MB/s)

Figure 4.11: Average throughput as a function of routing information update interval: 100% bytes in high-bandwidth traffic and uneven load

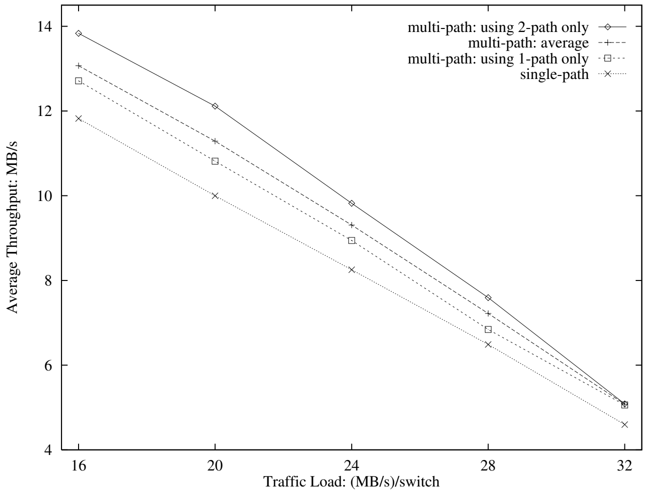
more hops is more sensitive to the increased routing update interval, and that the performance gap among different routing algorithms also increases with the update interval. The reason is that, with inaccurate routing information, a path with more hops has a higher risk of being based on bad information. When the routing update interval increases, this risk grows faster for longer paths. From the two figures, we observe that, the Shortest-dist$(P, 1)$ path gives up its performance advantage to the Shortest-dist$(P, 0.5)$ path as the routing update interval passes over 20 seconds, because the Shortest-dist$(P, 1)$ path will on average select paths with higher max-min rate and more hops than the Shortest-dist$(P, 0.5)$ path. The results show the importance of taking routing update interval into consideration when evaluating routing algorithms.

4.6 Simulation results for multi-path routing

In this section, we examine the performance of our multi-path routing algorithms. Since our evaluation of single path routing algorithms shows that the Shortest-dist$(P, 1)$ path algorithm has the best overall performance at least if routing information is frequently updated, we will only consider this algorithm at every priority level of multi-path routing. We will also limit our study to 2-path routing since our simulations show that the benefit of using a third and fourth path is limited (at most an additional 5% increase in throughput). Through simulation,



(a) 50% HB traffic



(b) 90% HB traffic

Figure 4.12: $G1$: average throughput as a function of traffic load for multipath routing

we answer the following questions: How much performance improvement can be achieved by using two paths and what is the performance impact on sessions that use a single path only?

Our main performance measure is the average throughput of high-bandwidth connections with multi-path routing, compared to that with single-path routing. Note that multi-path routing can improve throughput not only by adding a second path, but also by improving the throughput of the first path. The reason is that 2-path connections will often finish faster compared with single-path routing, thus freeing up bandwidth that is available for all paths. To show this effect, we will also present the average throughput for 1-path and 2-paths connections separately.

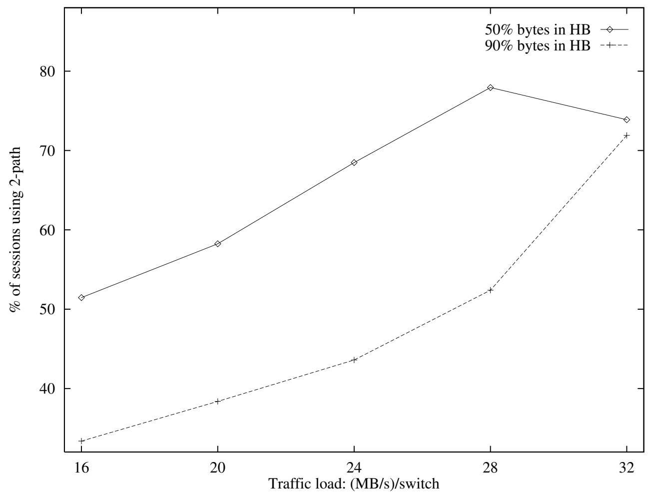
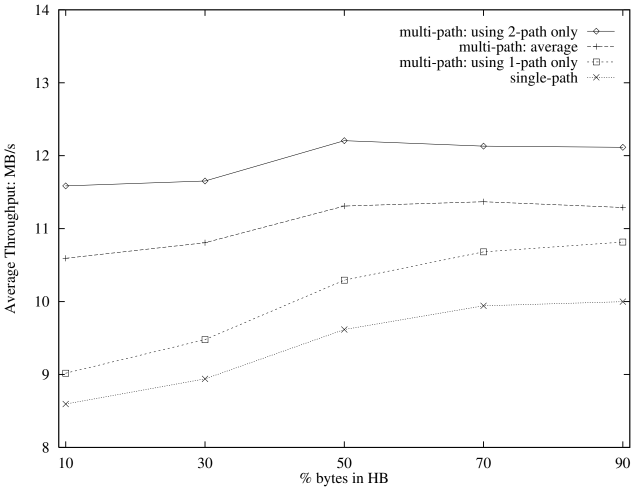


Figure 4.13: $G1$: percentage of sessions using two paths

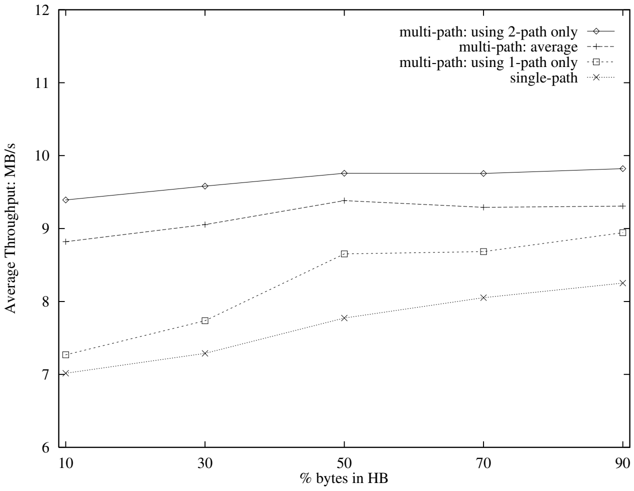
4.6.1 Average throughput as a function of traffic load

Figure 4.12 shows the average throughput as a function of the traffic load for two different percentages of high-bandwidth traffic, 50% and 90%. The results are for topology $G1$, but similar results were observed for the other topologies. We observe that 2-path routing increases the average throughput compared with single-path routing, not only for connections that use two paths but also for connections that use a single path. We also observe that, as the traffic load increases, the increase in throughput gets smaller, although the relative increase in throughput with multi-path routing compared with single-path routing remains relatively constant.

Figure 4.13 shows that the percentage of connections that use two paths increases with the traffic load. This is a result for the fact that, when the traffic load increases, the Shortest-dist($P, 1$) path algorithm tends to pick the shortest path more often, which leads to a relatively higher number of links with unused bandwidth, and these links can accommodate more secondary paths.



(a) 20 MB/s per switch



(b) 24 MB/s per switch

Figure 4.14: $G1$: average bandwidth as a function of the percentage of high-bandwidth traffic

4.6.2 Impact of high-bandwidth traffic volume

Figure 4.14 shows the average throughput using single-path and 2-path routing as a function of the percentage of high-bandwidth traffic for a traffic load of 20 MB/s and 24 MB/s per switch. Connections that use two paths achieve an average increase in throughput of 20% to 35%, while connections that use a single path have a increase of 2% to 8%. The overall improvement ranges from 13% to 26%. We also observe that the benefit of multi-path routing decreases as the contribution of high-bandwidth traffic increases. The reason is that more high-bandwidth traffic results in more competition among secondary paths.

Figure 4.15 shows the percentage of sessions that actually use two paths for the two scenarios in Figure 4.14. The percentage of sessions that use two paths decreases as the high-bandwidth traffic volume increases (although the number of 2-path connections goes up). The reason is that high-bandwidth connections can use any available network bandwidth, i.e. they can by themselves saturate links, making them unavailable for secondary paths. As a result, more high-bandwidth traffic means fewer links available for secondary paths and a lower percentage of 2-path connections.

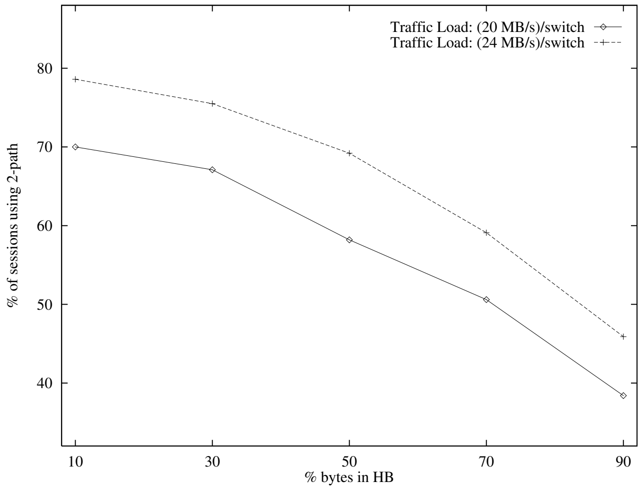


Figure 4.15: $G1$: arrival rate 24 MB/s per switch

Our results suggest that multi-path routing is an effective technique to make use of unused network resources in a max-min fair share network. While the performance improvement for multi-path routing is relatively constant across different traffic loads (previous section), it is sensitive to the volume of high-bandwidth traffic. When the percentage of high-bandwidth traffic increases the performance improvement from multi-path routing goes down, both because it is harder to find secondary paths and because less bandwidth is available once they are established.

4.6.3 Variance of Increased Throughput

Figure 4.16 shows the distribution of the throughput increase over single-path routing for sessions that use two paths; the graph includes results for 30% and 70% high-bandwidth traffic (compare to Figure 4.14). The two distributions are fairly symmetric, with an average throughput increase around 3 MB/s. A few sessions increased their throughput by as much as 6 to 10 MB/s. A few sessions suffer a throughput reduction. The reason is that the early completion of some sessions changes the routes of later sessions, and in some cases that results in routes with a slightly lower average rate.

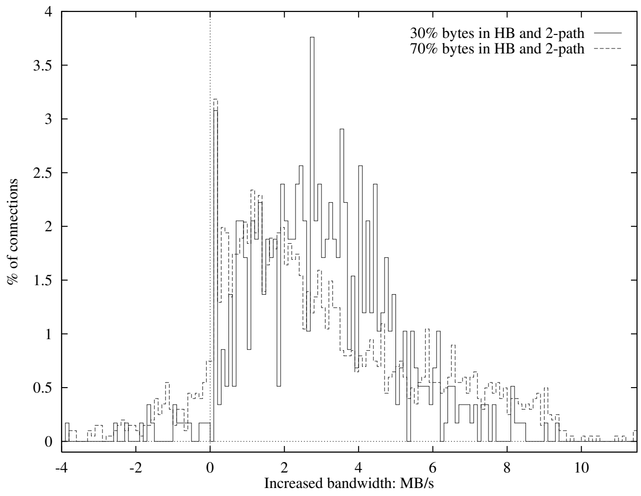


Figure 4.16: $G1$: throughput increase for two-path connections with an arrival rate 20 MB/s per switch

Figure 4.17 shows the distribution of the throughput increase over single-path routing for sessions that could not find a second path; the peak (off scale) corresponds to 54% of the connections observing an increase of about 0.1 MB/s. On average, single-path connections benefit slightly from multipath routing. We also observed that the distribution is more spread out when the ratio of high-bandwidth traffic is higher. This indicates that there is more interference among high-bandwidth connections.

4.7 Sensitivity analysis

In the previous discussions, we used a fixed Peak Cell Rate (PCR) (3-5 MB/s depending on message size) for low-latency traffic, and a fixed routing cost of 10 ms for routing algorithms other than the widest-shortest path. In this section, we show the performance impact of changing these parameters.

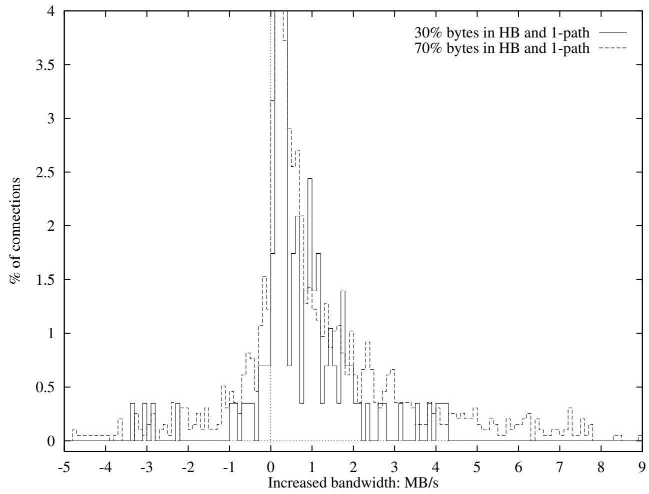


Figure 4.17: $G1$: throughput increase for single-path connections with an arrival rate 20 MB/s per switch

4.7.1 Impact of PCR of low-latency traffic

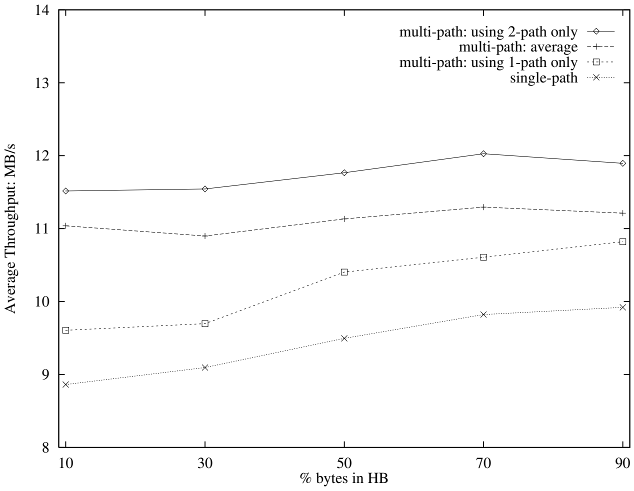
In Figure 4.18, we show the performance impact on single path and multi-path routing of (a) increasing and (b) decreasing the PCR rate for low-latency traffic by 1 MB/s. The topology is G1 and the traffic load is 24 MB/s per switch. We observe that the performance of single-path routing is fairly insensitive to the PCR. For multi-path routing, while the overall performance is very close to what we showed earlier (Figure 4.14(a)), the performance improvement is slightly higher when the PCR is lower. The reason is that a lower PCR leaves more unused bandwidth for lower priority paths.

4.7.2 Impact of routing cost

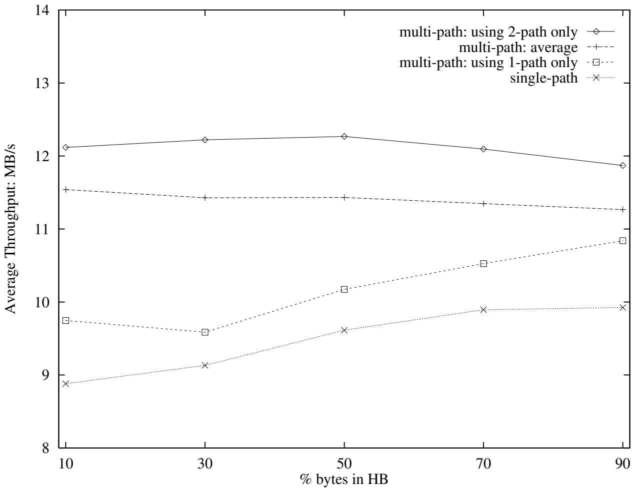
In Table 4.2 we list average throughputs for two different routing costs (1ms instead of 10ms); the results are for topology $G1$, a ratio of high-bandwidth traffic of 90%, and a traffic load of 24 MB/s per switch. We see that the change in routing cost has little impact on the results. This is no surprise since high-bandwidth flows have a long life time.

4.7.3 Impact of LLvsBB

In Table 4.3, we list average throughputs for two different cutoff **LLvsBB**'s between high-bandwidth and low-latency traffic. The results are for topology $G1$, a **HBFraction** of 90%,



(a) PCR 4-6 MB/s



(b) PCR 2-4 MB/s

Figure 4.18: $G1$: arrival rate 24 MB/s per switch

and a traffic load of 28 MB/s per switch. The performance becomes slightly worse when **LLvsBB** increases from 1 to 10 MByte, due to the decreased number of connections and consequently increased traffic concentration. The impact on the results is small, although it affects shortest-widest paths more than Shortest-dist(P, 1) paths.

	1 ms	10 ms
shortest-widest	5.86	6.08
widest-shortest	7.45	7.44
Shortest-dist$(P, 1)$	8.29	8.25
Shortest-dist$(P, 2)$	7.45	7.50
Shortest-dist$(P, 0.5)$	7.91	7.91

Table 4.2: Average throughput in MB/s for two routing costs

	1 MB	10 MB
shortest-widest	3.52	3.18
widest-shortest	5.86	5.59
Shortest-dist$(P, 1)$	6.49	6.42
Shortest-dist$(P, 2)$	5.54	5.50
Shortest-dist$(P, 0.5)$	6.35	6.12

Table 4.3: Average throughput in MB/s for two **LLvsHB**'s

4.8 Related Work

To the best of our knowledge, this is the first study on routing algorithms in networks with max-min fair sharing. In this section, we review related work in the areas of service definition and routing.

It has long been recognized that data communication applications can be divided into several classes, including bulk data transfer and interactive applications. However, traditionally networks do not distinguish between these classes. With the advent of integrated service networks, several proposals [18, 92, 64] have been made to divide the traditional best-effort service into multiple service classes. An alternative to defining service classes is to implement queueing policies that optimize the performance of interactive application without sacrificing bulk data transfer applications; this is for example done in the DataKit network [31]. These approaches rely on traffic management support to optimize the performance of different classes of applications. In contrast, we assume that the traffic management algorithms treats all data transfers in the same way, and we pursue the use of routing to optimize performance.

Packet-switched networks have traditionally used shortest-path routing. While different measured “link-cost” measures can be used (see [91, 73]), earlier networks typically selected minimal-hop paths. The problem with using measured link costs is that it does not always accurately account for how resources are shared among connections, so they can be inaccurate and even misleading. The rate information we use is an accurate measure of available bandwidth since we model the sharing algorithm that is used in the max-min fair share network.

Routing in circuit-switched networks has focused on finding paths with certain QoS guarantees while minimizing the blocking rate of future requests. Trunk-reservation [2], adaptive routing [35], shortest-widest path [102], and min-max routing have been well-studied and are very relevant to today's QoS routing in data networks. However, these algorithms are based on a residual bandwidth model, which is representative for reservation-based networks but not for max-min fair share networks.

Multipath routing algorithms have been used to optimize network performance [75, 101, 5, 13, 94, 99]. Multiple paths are selected in advance. When data traffic arrives, a path with the lowest traffic load is used. None of these studies addresses the issue of fairness. In contrast, we studied the use of multiple paths simultaneously to maximize throughput in a max-min fair share network.

4.9 Summary

In this chapter, we have studied routing support for high-bandwidth traffic in max-min fair share networks. A fundamental feature of our routing algorithms is that they make use of rate information provided by the fair share congestion control mechanism. By giving the routing algorithm access to rate information, we couple the coarse grain (routing) and fine grain (congestion control) resource allocation mechanisms, allowing us to achieve efficient and fair allocation of resources.

To achieve high network utilization, a routing algorithm should not only conserve network resources but also balance network load. Neither selecting a path with minimum hop count nor selecting a path with widest available rate achieves this goal.

Our evaluation of single-path routing algorithms for high-bandwidth traffic shows that, when routing information is accurate, the Shortest-dist($P, 1$) path algorithm performs best in most of the situations we simulated. When the routing information becomes less accurate because of an increase in the routing update interval, the Shortest-dist($P, 0.5$) path algorithm can outperform the Shortest-dist($P, 1$) path algorithm. While the Shortest-widest path algorithm can give slightly better performance when the network load is very light, it can have very poor performance for medium and high traffic loads, because it tends to pick paths that are resource-intensive. The Shortest-dist($P, 1$) path and the Shortest-dist($P, 0.5$) path algorithms are able to route around bottlenecks, thus avoiding the clusters of connections with very low throughput that are sometimes the result of using the Widest-shortest path algorithm. Overall, the Shortest-dist($P, 1$) path algorithm balances the weight given to the "shortest" and "widest" metrics in an appropriate way, and the Shortest-dist($P, 0.5$) path algorithm is more robust with respect to delayed routing information.

Finally, we introduces a prioritized multi-level max-min fairness model, in which multiple paths are assigned different priority. This approach prevents a multi-path connection from grabbing an unlimited amount of bandwidth by using a large number of paths, i.e. additional paths only use unused bandwidth and do not affect the bandwidth available to primary paths. Our simulations show that 2-path routing increases the average bandwidth compared with single-path routing by 25% overall and 35% for those connections using two paths.

Chapter 5

Routing Traffic with Bandwidth Guarantees

Transmission of multimedia streams imposes a minimum-bandwidth requirement on the path being used to ensure end-to-end QoS guarantees. In this chapter, we study how to route traffic requiring bandwidth guarantees. The only QoS parameter is **bandwidth**. The goal of routing traffic with bandwidth guarantees is to find a feasible path if one exists, and to select one that achieves efficient resource utilization if more than one such path is available.

While finding a path with bandwidth guarantees has been studied extensively for telecommunication circuit-switched networks, but QoS routing with bandwidth guarantees in packet-switched data network is very different because of the distributed control and lack of fully connected network topology in data networks.

Although any shortest-path algorithm can be used to select a feasible path with a requested bandwidth, the resource efficiency of different "shortest path" criteria is not well understood. Several path selection algorithms have been proposed in the literature, including widest-shortest path [44], shortest-widest path [102], and utilization-based path selection algorithms [70]. However, a systematic evaluation of these algorithms is missing.

In this chapter, we present a systematic evaluation of four routing algorithms using the following "optimality" criteria: minimum hop count, maximum residual bandwidth, minimum path cost based on link utilization, and a variant of the dynamic-alternate path algorithm used in telecommunication networks. These algorithms make different tradeoffs between minimizing resource consumption and balancing network load. Our evaluation considers not only the call blocking rate but also the fairness to requests for different bandwidths, robustness to inaccurate routing information, and sensitivity to the routing information update frequency. Finally, the choice of routing algorithm for traffic with bandwidth guarantees can have a significant impact on the performance of best-effort sessions, so we also evaluate that performance metric.

The rest of the chapter is organized as follows. After a brief discussion on finding feasible paths in Section 5.1, we present the four routing algorithms in Section 5.2. The next three sections examine the performance of different routing approaches: dynamic on-demand routing in Section 5.3, static routing in Section 5.4, and class-based routing in Section 5.5. The performance impact on best-effort traffic is examined in Section 5.6. We discuss related work in Section 5.7 and conclude in Section 5.8

5.1 Selecting Feasible Paths

A path is *feasible* if the unserved bandwidth of all links on the path is higher than the requested bandwidth. Given a path $\mathbf{p} = \{i_1, \ldots, i_k\}$, the maximal reservable bandwidth (**mrb**) on the path **p** is the minimum of the reservable bandwidth of all links on the path:

$$\mathbf{mrb_p} = \min\{R_{i_j} | i_j \in \mathbf{p}\}.$$

The path **p** is feasible if the $\mathbf{mrb_p}$ is no less than the requested bandwidth **b**: $\mathbf{mrb_p} \geq \mathbf{b}$.

To select a feasible path, either Dijkstra's shortest-path algorithm or the Bellman-Ford shortest-path algorithm (see [20]) can be used. For example, we can define the cost of a link i_j as R_{i_j}—the link residual bandwidth—and the cost of a path **p** the $\mathbf{mrb_p}$. Using either shortest-path algorithm, a path with the maximum **mrb** can be selected. If the **mrb** is no less than the request bandwidth **b**, a feasible path is found. An alternative is to prune all links whose residual bandwidth is less than the request bandwidth and then to select a shortest-path in the remaining graph using any cost function.

5.2 Selecting Efficient Paths

While a feasible path can be selected using any shortest-path algorithm, additional optimality constraints need to be imposed to achieve efficient resource utilization. The most common way for a routing algorithm to achieve resource efficiency is to limit resource consumption and to keep the network load balanced.

For traffic with bandwidth guarantees, resource consumption can be reduced by restricting the hop count of the path being selected, while the network load can be balanced by selecting the least loaded path. However, these two "optimality" constraints can conflict, since a path with the fewest hops may contain heavily loaded links. Routing algorithms with different optimality criteria can be obtained by attaching different weights to the two constraints. Understanding the performance tradeoffs between these routing algorithms is essential to the successful deployment of QoS in future networks.

5.2.1 Selection Criteria

Several path selection algorithms that put different weight on limiting hop count and on balancing the network load have been proposed in the literature. They include the widest-shortest path [44], the shortest-widest path [102], and a utilization-based shortest path algorithm [70]. In the literature on telecommunication routing, where the network is often fully connected, it has been shown [35] that dynamic alternate path routing combined with trunk reservation performs well. In such a scheme, a one-hop path is always used if it is available; a two-hop alternate path is chosen randomly when a call is blocked on the one-hop path. Trunk reservation reduces the number of calls using alternate paths by only permitting a certain number of trunks on a link to be used by alternate paths. Since data networks are rarely fully connected, we introduce a simple variant of dynamic alternate path routing that does not use trunk reservation. Overall, we selected the following four candidate paths for evaluation:

- **Widest-shortest path**: a path with the minimum hop count among all feasible paths. If there are several such paths, the one with the maximum reservable bandwidth is selected. If there are several such paths with the same bandwidth, one is randomly selected.
- **Shortest-widest path**: a path with the maximum bandwidth among all feasible paths. If there are several such paths, the one with the minimum hop count is selected. If there are several such paths with the same hop count, one is randomly selected.
- **Shortest-distance path**: a feasible path with the shortest distance. The distance function is defined by
$$\text{dist}(\mathbf{p}) = \sum_{j=1}^{k} \frac{1}{R_{i_j}}$$
where R_{i_j} is the bandwidth available on link i_j. It has been shown that this algorithm performs consistently well when routing best-effort sessions 4.
- **Dynamic-alternate path**: Let n be the hop count of a minimum-hop path when the network is idle. A dynamic-alternate path is a widest-shortest path with no more than $n+1$ hops.

The widest-shortest path gives high priority to limiting the hop count, while the shortest-widest path gives high priority to balancing the network load. The shortest-distance path uses the distance function to dynamically balance the impact of the hop count and the path load. For the dynamic-alternate path, we do not consider trunk-reservation, since the minimal-hop path may not be unique and the performance may be sensitive to the trunk-reservation rate. Compared with the shortest-widest path, the dynamic-alternate path puts an upper bound $(n+1)$

on the widest-shortest path. The four algorithms are points in a spectrum that corresponds to different tradeoffs between conserving resources and balancing network load:

dynamic-alternate	widest-shortest	shortest-distance	shortest-widest
$\longleftarrow$ *resource conserving*			*load balancing* $\longrightarrow$

5.2.2 Algorithms

In this section, we describe algorithms for each of the four path selection criteria.

A widest-shortest path algorithm based on the Bellman-Ford algorithm is described in [44]. It selects a path with maximal residual bandwidth for different hop count. The one with minimum hop count is a widest-shortest path. We can also modify Dijkstra's algorithm to find the widest-shortest paths by defining two distance functions: the hop count is the primary function and the **mbr** is the secondary function. When selecting the next node to mark, one selects the node with the minimal hop count. If several nodes have the same minimal hop count, one selects the node with the largest **mbr**. The algorithm terminates when the destination is reached.

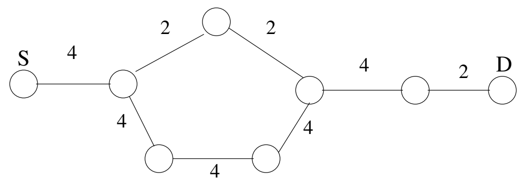
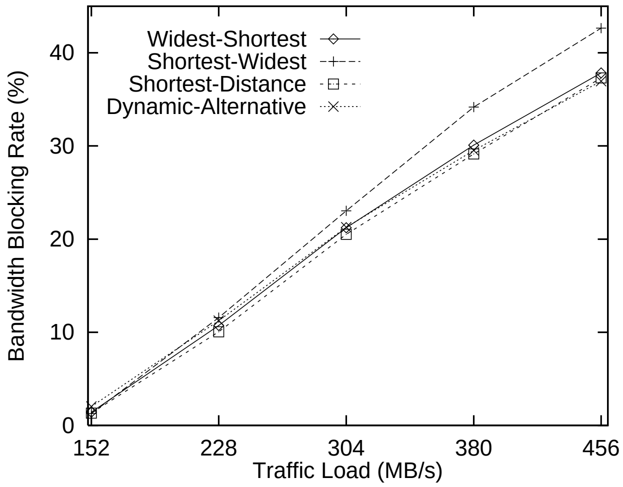


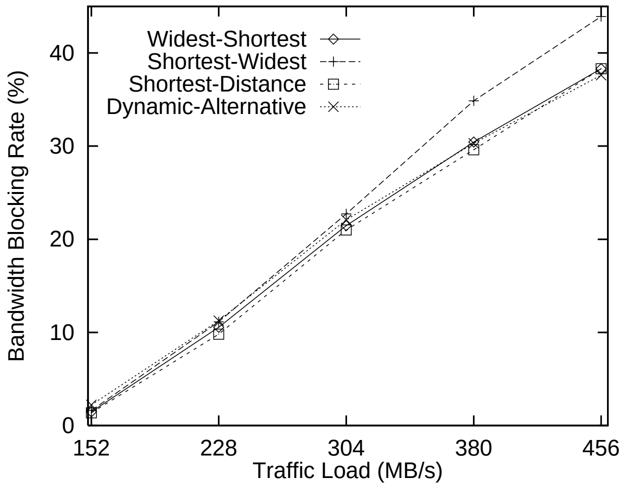
Figure 5.1: Finding a path from S to D with bandwidth 2

A simple shortest-widest path algorithm applies Dijkstra's algorithm twice. First we find a widest path; assume its bandwidth is B. Then we find a shortest path with bandwidth B by using Dijkstra's algorithm on a network that only includes links of bandwidth B or higher. Note that the single-pass link-state shortest path algorithm given in [102] does not always find the shortest-widest path. For example for the topology in Figure 5.1, the algorithm will select the lower path in the graph using links with bandwidth 4. The reason is that, when a link (e.g., the link connected to the node D) with low bandwidth has to be added to the path, the earlier shortest-widest segment may no longer be the shortest-widest one.

The shortest-distance path can be selected by any shortest-path algorithm using the distance function as the cost function. The dynamic-alternate path can be selected by using the widest-shortest path algorithm while imposing a hop count restriction on the nodes being selected.



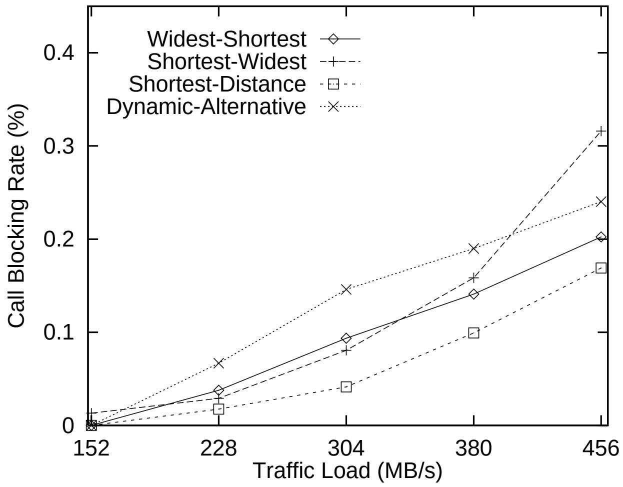
(a) 60% voice and 40% video sessions



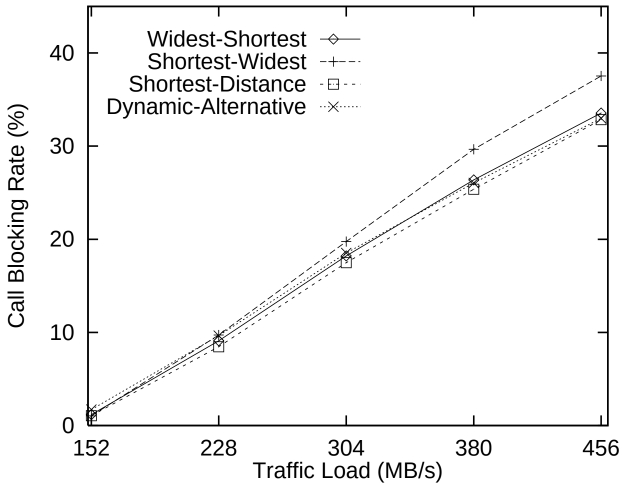
(b) 100% video sessions

Figure 5.2: Bandwidth blocking rate as a function of network load: MCI topology, evenly distributed load, and path selection on demand

Each of these algorithms may select a path that is not feasible, either because of stale routing information or because of changes in the network state while the connection is being established. If that happens, the request is rejected. Similarly, it is possible that the algorithms do not find a feasible path, although one exists.



(a) Audio sessions



(b) Video sessions

Figure 5.3: Blocking rate as a function of network load: MCI Topology, 60% voice sessions, evenly distributed load, and path selection on demand

5.3 Dynamic On-demand Routing

In this section, we examine the performance of the four routing algorithms described in Section 5.2.2. We assume that guaranteed sessions are the only traffic class in the network. Paths are selected on-demand using dynamic load information, which is updated asynchronously every 30 seconds.

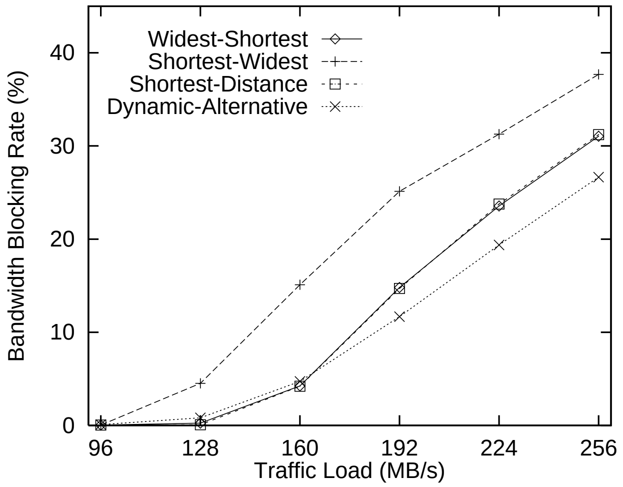


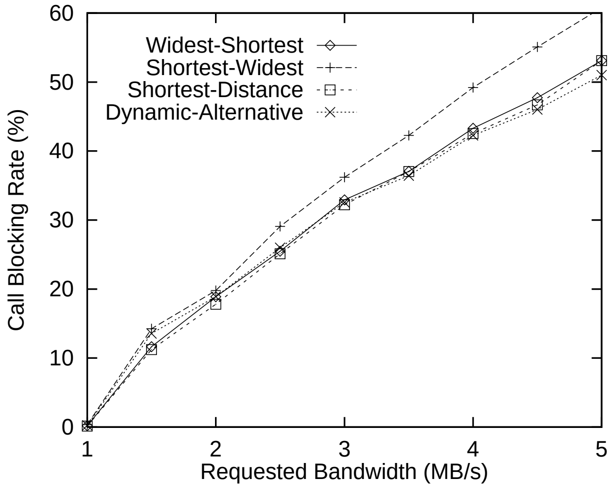
Figure 5.4: Bandwidth blocking rate as a function of network load: cluster topology, 100% video sessions, Evenly distributed load, and path selection on demand

5.3.1 Blocking Rate

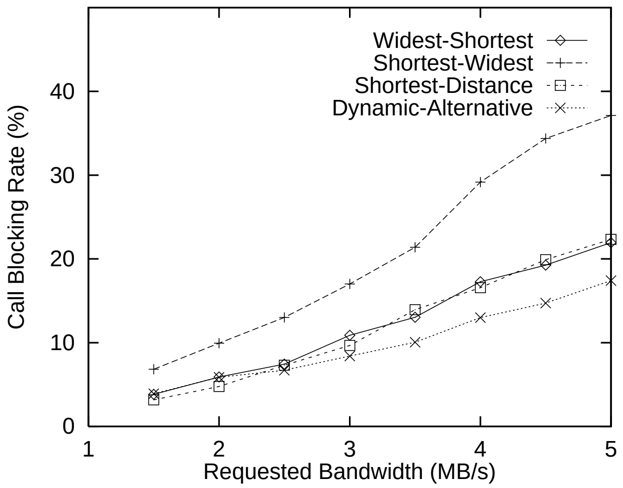
We examine the blocking rate of the four routing algorithms for different loads, topologies, and call holding time distributions. We first consider evenly distributed traffic loads and then unevenly distributed loads. The details of the traffic loads are described in Section 3.3.2.

5.3.1.1 Evenly Distributed Load

Figure 5.2 shows the bandwidth blocking rate as a function of the traffic load. Traffic is evenly distributed and results are shown for 60% audio and 40% video sessions (a), and for 100% video sessions (b). Overall, we see that the bandwidth blocking rate is linearly proportional to the network load. When the load is heavy, the shortest-widest path performs poorly because it tends to allocate long expensive paths, which penalizes later arrivals. As the network load decreases, the performance difference between the four algorithms becomes smaller; the shortest-distance path performs slightly better and the shortest-widest path slightly worse than the others. When the network load is very light, all four algorithms have similar performance, although the shortest-widest path now performs best and the shortest-distance path algorithms is a close second. The following table lists the bandwidth blocking rate (in %) for the same simulation



(a) MCI topology, 40% video sessions, and 456MB/s



(b) Cluster topology, 100% video sessions, and 192MB/s

Figure 5.5: Call blocking rate as a function of requested bandwidth: evenly distributed load and path selection on demand

configuration as in Figure 5.2 (a) when the the network load is 114 MB/s.

	WS	SW	SD	DA
40% video sessions	0.2388	0.1665	0.1836	0.6860
100% video sessions	0.1505	0.0604	0.0678	0.4806

This result is different from the result obtained for best effort traffic in Chapter 4, where we found that the impact of the routing algorithm on performance was often significant. This difference is caused by the different resource sharing rules in the two traffic classes. For best effort traffic, link capacity is shared among all sessions, and all paths are feasible, even those that use links that are heavily congested compared with other parts of the network. Compared with the widest-shortest path, which is strictly a minimum-hop path, the shortest distance path algorithm is able to route around congested links. For traffic with bandwidth guarantees, the heavily congested links are no longer feasible and all algorithms will route around them. For example, a widest-shortest path is not necessarily a minimum-hop path, but the shortest one among all feasible paths.

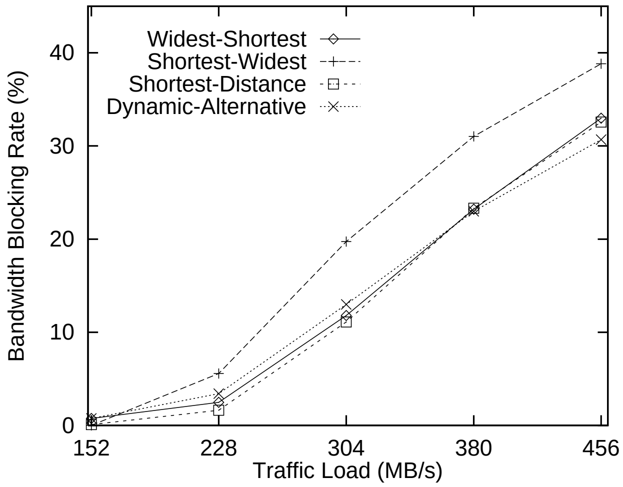
In Figure 5.3 we show the call blocking rate for audio and video traffic separately. We see that audio traffic has a much lower blocking rate (under 0.4%) than video traffic, as one would expect given its lower bandwidth requirements.

Figure 5.4 shows the bandwidth blocking rate as a function of traffic load for the cluster topology. We observe that the shortest-widest path performs much worse than the other three, and the dynamic alternate path performs much better when the load is heavy. This suggests that with a more symmetric topology, restricting resource consumption becomes more important when the network load is heavy. The dynamic-alternate path does not consider "expensive" paths that are two or more hops longer than the minimal hop path, i.e. it rejects requests that would require a relatively expensive path, favoring later "cheaper" requests. The shortest-distance path performs slightly better than the dynamic-alternate path when the load is light. The reason is that the restriction on the hop count of a path limits the degree to which the algorithms can route around congested links.

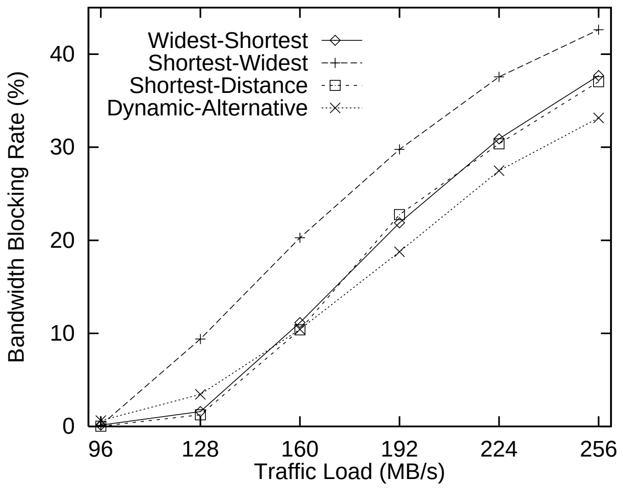
In Figure 5.5 we show the call blocking rate as a function of the requested bandwidth; all requests for less than 1 MB/s (i.e. audio) are combined in a single data point (MCI topology). We see that the call blocking rate is almost a linearly function of requested bandwidth, confirming that the algorithms favor sessions that ask for less bandwidth.

5.3.1.2 Unevenly Distributed Load

Figure 5.6 shows bandwidth blocking rates for an uneven load distribution. 50% of the traffic is between the west coast and east coast in the MCI topology, and between the left bottom corner building and other buildings in the cluster topology. For both topologies, our earlier observations hold: the shortest-widest path performs worse than the other three algorithms, the dynamic-alternate path performs better when the load is heavy, and the shortest distance path performs better than dynamic alternate path when the load is light.



(a) MCI topology

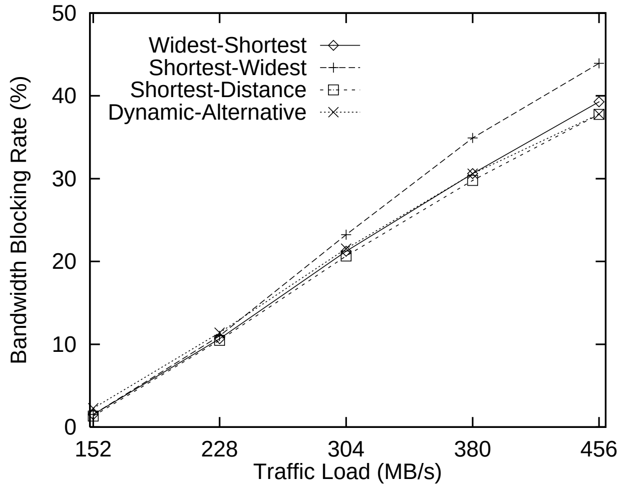


(b) Cluster topology

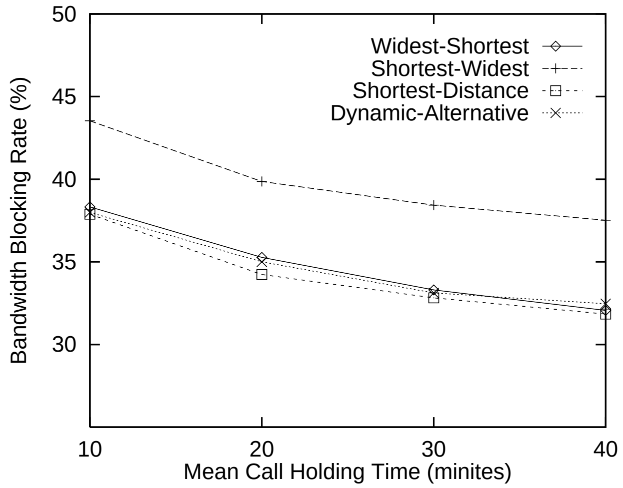
Figure 5.6: Bandwidth blocking rate as a function of network load: 100% video sessions, unevenly distributed load, and path selection on demand

5.3.1.3 Impact of Call Holding Time Distribution

Figure 5.7 (a) shows the bandwidth blocking rate for the configuration used in Figure 5.2, but using an exponential call holding time distribution with the same mean as the long tail distribution used in Figure 5.2. We observe a slightly higher bandwidth blocking rate when using an exponential call holding time distribution. The reason is that with a long-tail distribution, there are more sessions with a short call holding time. These shorter sessions can more easily



(a) Bandwidth blocking rate as a function of network load

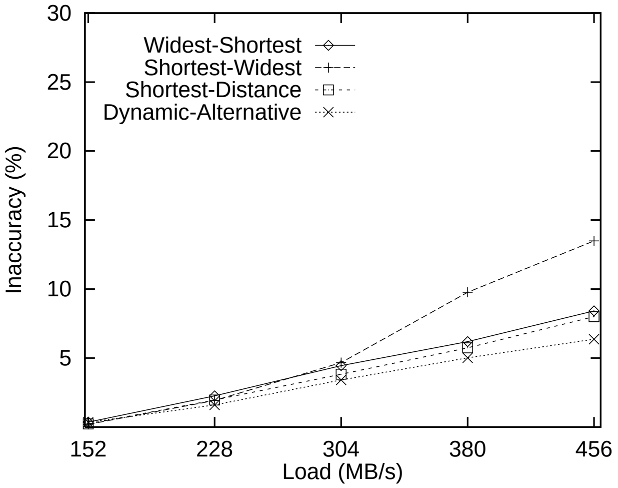


(b) Bandwidth blocking rate as a function of mean call holding time: 380 MB/s

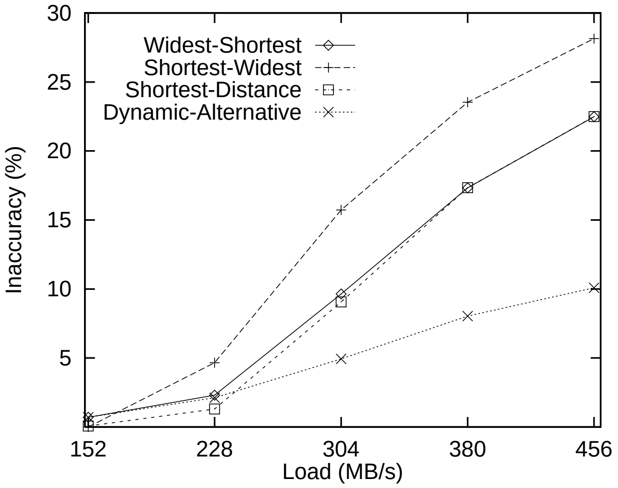
Figure 5.7: Bandwidth blocking rate: MCI topology, 60% voice sessions, evenly distributed load, exponential call holding time, and path selection on demand

use the bandwidth left unused by long sessions, minimizing the impact of a poor routing decision.

Figure 5.7(b) shows the call blocking rate as a function of the mean call holding time. We see that the bandwidth blocking rate is higher for shorter mean call holding times. The reason is that, with the same traffic load but lower mean call holding times, more sessions arrive during a routing update interval. This reduces the accuracy of the routing information, which results



(a) Evenly distributed load



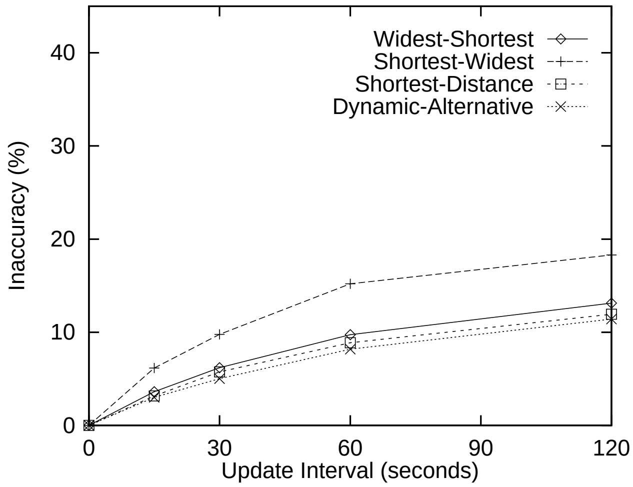
(b) Unevenly distributed load

Figure 5.8: Routing inaccuracy as a function of network load: MCI topology, and 100% video sessions

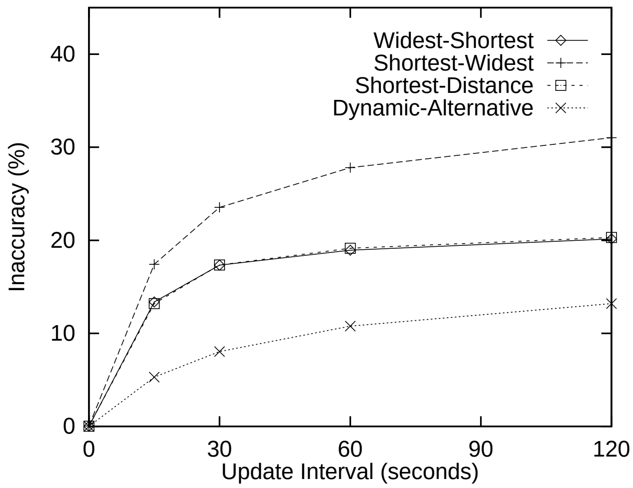
in more rejected sessions.

5.3.2 Routing Inaccuracy

In Figure 5.8 we show the routing inaccuracy as a function of the network load for both evenly and unevenly distributed traffic loads.



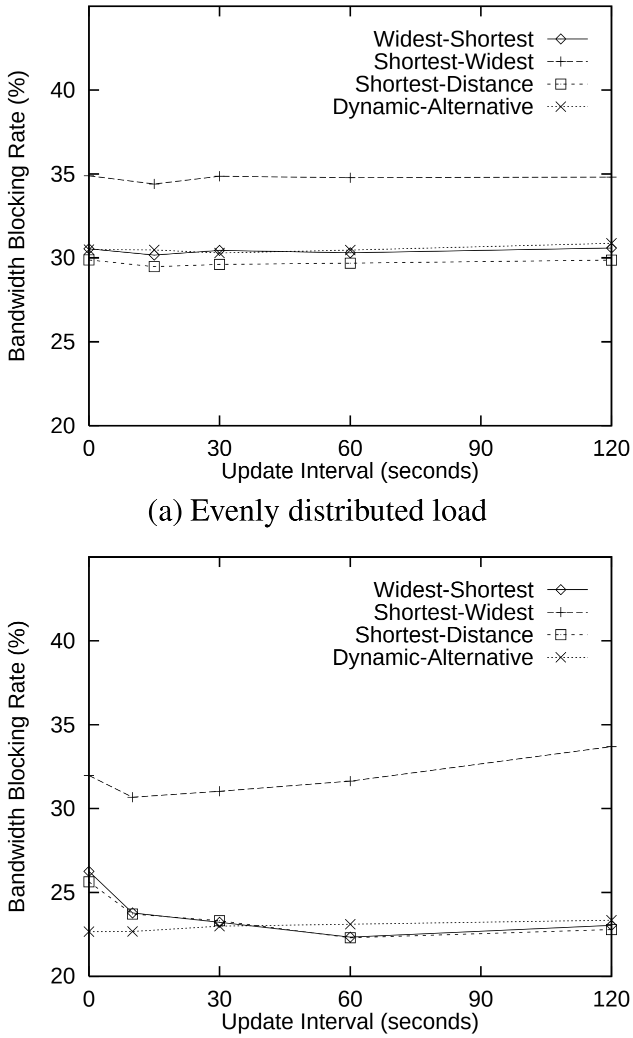
(a) Evenly distributed load



(b) Unevenly distributed load

Figure 5.9: Inaccuracy as a function of routing update time: MCI topology, 100% video sessions, and 380MB/s

We observe that the routing inaccuracy increases with the network load for all algorithms, and that the shortest-widest path algorithm is the most sensitive to inaccurate routing information. The widest-shortest path and the shortest-distance path have similar performance, with the shortest-distance path doing slightly better. The dynamic-alternate path algorithm is the most robust one when the load is heavy but it loses ground to the shortest-distance path algorithm when the load is uneven and light. We also see a significant difference between evenly and



(a) Evenly distributed load

(b) Unevenly distributed load

Figure 5.10: Bandwidth blocking rate as a function of routing information update interval: MCI topology, 100% Video sessions, and 380MB/s

unevenly distributed loads in how sensitive performance is to routing inaccuracies. The reason is that when the traffic is concentrated (unevenly distributed), more sessions will be routed to the area where traffic is concentrated, which results in a faster rate of changes in the network state and more incorrectly routed sessions. We can see this more clearly by comparing the routing inaccuracy metric (Figure 5.8) with the blocking rate (Figure 5.2 (b) and 5.6 (a)). We observed that for evenly distributed traffic most of the blocked sessions are rejected by the routing algorithm. For uneven traffic, except for the dynamic-alternate path, blocked sessions have typically been routed, but they are rejected by the admissions control module on one of the switches.

The routing inaccuracy metric covers both sessions that are routed but later rejected and sessions that are not routed when a paths does exist. Our simulations show that the sessions that are routed but are later rejected dominate routing inaccuracy. Among the sessions for which no paths are found, 6-10% percent for even load and 10-40% for uneven load would have found a path if accurate routing information had been available.

Figure 5.9 explores the sensitivity of the routing accuracy to the routing update interval, which was 30 seconds in the earlier simulations. We see that the routing inaccuracy increases with the update interval. However, the pace of increasing slows down above a threshold of about 30 seconds. The shortest-widest path is most sensitive and the dynamic-alternate path is least sensitive to increases in the routing update interval.

Figure 5.10 shows the bandwidth blocking rate as a function of the routing information update interval. It is interesting to see that, while the routing inaccuracy increases with the routing update interval, the bandwidth blocking rate remains quite stable. In some cases, the bandwidth blocking rate is even slightly higher when the routing information is more accurate. The reason is that, with more accurate information, the network allocates resource more "conservatively" in the sense that it discourages sessions from trying whether there is a path available. With less accurate information, it can "over allocate" resources: even though no path can be found at routing time with accurate routing information, sessions can successfully set up a connection, using resources that are being relinquished by connections that are terminating.

We conclude that increasing the routing information update interval does not affect the overall blocking rate much, even though more sessions are routed and rejected by the admission control algorithm, or are not routed while a path exists.

5.4 Static Routing

Dynamic routing can be expensive both in terms of operational costs and implementation complexity. Static routing uses static link capacity as link-state and is much simpler to implement, since it, for example, eliminates the need to distribute dynamic traffic information. Our simulation results show that in a network with evenly distributed traffic load, the performance difference between static and dynamic routing can be very small. However, when the load is unevenly distributed, the performance difference between dynamic and static routing can be significant (see Figure 5.11). The reason for the significant difference is that loaded links cannot be avoided with static routing, and selecting a loaded path leads to a session being rejected.

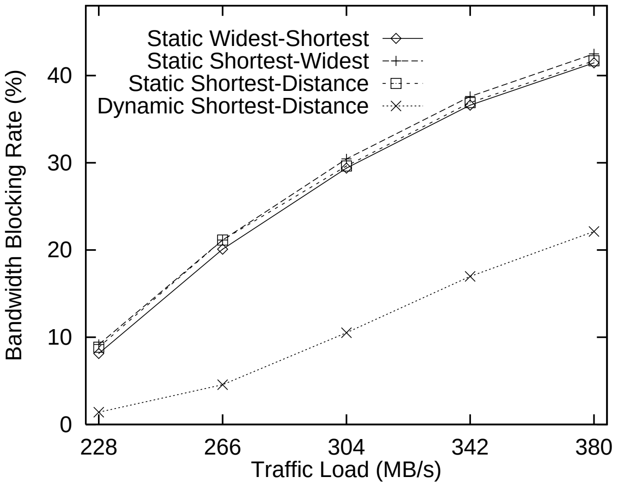


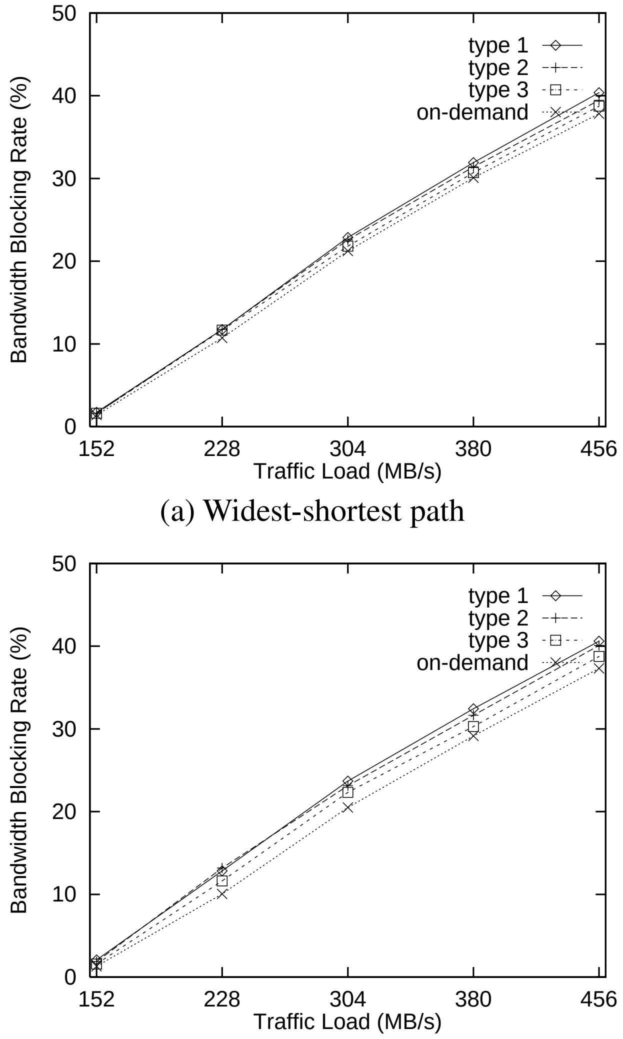
Figure 5.11: Bandwidth blocking rate as a function of network load: MCI topology, 100% video sessions, and unevenly distributed load

An evenly distributed load matches the traffic pattern that the network was designed for. In such cases, static routing may work reasonably well. However, as we mentioned in Chapter 1, network design often relies on long-term measurements of user traffic and network load, but the actual traffic pattern in the network changes over time. Dynamic routing can track these changes in the traffic pattern, allowing it to make more efficient use of the network resources.

5.5 Class-based Routing

An alternative to on-demand dynamic routing is to use pre-computed paths: each router updates its paths regularly when new link-state information is received. The scheme is still dynamic but it is cheaper. Since the requested bandwidths can be very diverse, using a path that meets all bandwidth requests will result in using the widest path, which does not always perform well, as we have shown above. An alternative is to use class-based routing: several paths, each for a different bandwidth range, are precomputed. A new session selects the path with the lowest bandwidth satisfying the request.

Figures 5.12 and 5.13 compare the performance of three ways of selecting pre-computed paths and compares them with on-demand path selection; we consider the widest-shortest and shortest-distance path algorithms, since they have the best overall performance in Section 5.3. The "type 3" algorithm uses three classes serving bandwidth requests falling in the ranges $(0, 1]$, $(1, 3]$, and $(3, 5]$. The "type 2" algorithm uses two classes, serving audio and video



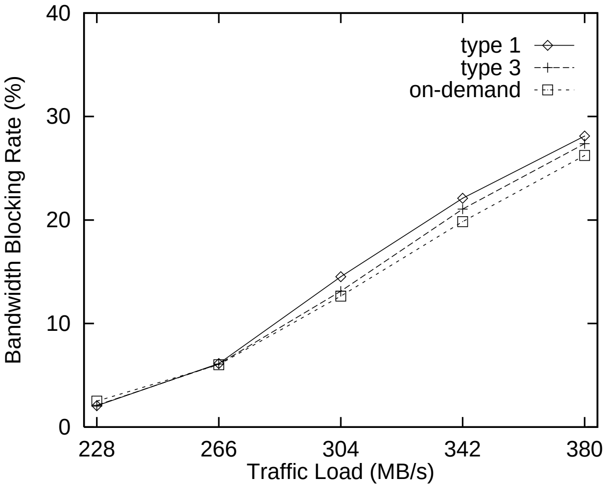
(a) Widest-shortest path

(b) Shortest-distance path

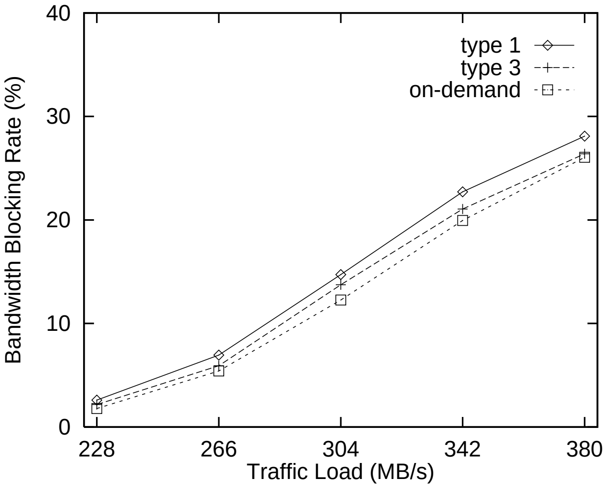
Figure 5.12: Bandwidth blocking rate as a function of network load: MCI Topology, 60% voice sessions, evenly distributed load

traffic, and the "type 1" algorithm uses only one class. We observe that the efficiency increases with the number of classes. However, the difference is not very high, which suggests that class-based routing is feasible. We also see that the use of class-based routing has less effect on the widest-shortest path algorithm, while the shortest-distance path gives up the small performance advantage it had over the widest-shortest path when paths are selected on demand.

Figure 5.14 shows the call blocking rate of class-based routing and on-demand routing as a



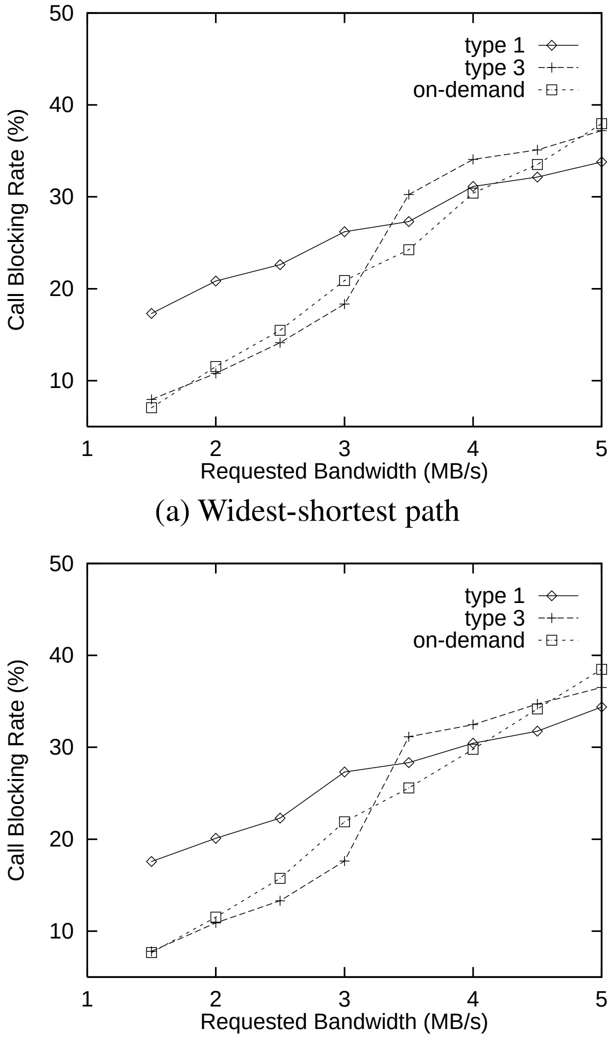
(a) Widest-shortest path



(b) Shortest-distance path

Figure 5.13: Bandwidth blocking rate as a function of network load: MCI Topology, 100% video sessions, unevenly distributed load

function of requested bandwidth. Since only video sessions are considered, there is no difference between the "type 1" and "type 2" schemes. The data point for a bandwidth x represents the call blocking rate for sessions with bandwidth requests in the range $(x - 0.5, x]$. We see that for all algorithms, the call blocking rate increases slowly with the requested bandwidth. We also see that class-based routing treats sessions with different bandwidth demands more evenly than on-demand routing. The reason is that while pre-computing paths may be less efficient, it

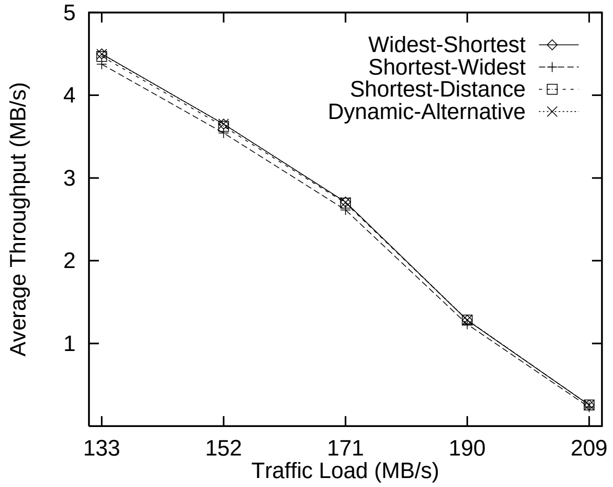


(a) Widest-shortest path

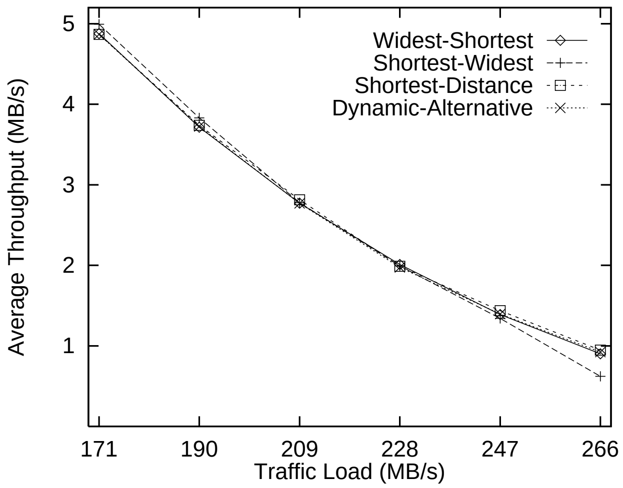
(b) Shortest-distance path

Figure 5.14: Call blocking rate as a function of requested bandwidth: MCI Topology, 100% video sessions, unevenly distributed load (380MB/s)

reduces bandwidth fragmentation, which benefits larger requests. We also notice that with the "type 3" algorithm, there is a jump in the call blocking rate between the two classes. The reason is that while the bandwidth demands are uniformly distributed, the pre-computed path for the high-bandwidth class can accommodate fewer sessions, increasing the blocking rate. Moreover, the high-bandwidth path is likely to have more hops, increasing the risk of interference from other sessions and thus the routing inaccuracy.



(a) Evenly distributed load



(b) Unevenly distributed load

Figure 5.15: Average throughput as a function of network load: MCI Topology, 60% high-bandwidth best-effort traffic, and 40% video traffic

5.6 Performance Impact on Best-effort Sessions

In a network with multiple classes of traffic, what paths are used for high priority sessions can affect the performance of lower priority traffic. In this section, we consider networks with two classes of traffic: guaranteed sessions and best-effort sessions. Best-effort traffic uses resources left unused by the guaranteed sessions. The routing algorithm used for best effort sessions is the shortest-distance path algorithm with the distance for a path defined as (Chapter 4)

$$\mathrm{dist}(\mathbf{p}) = \sum_{i=1}^{k} \frac{1}{r_i}$$

where r_i is the max-min fair rate of link i for a new best-effort session.

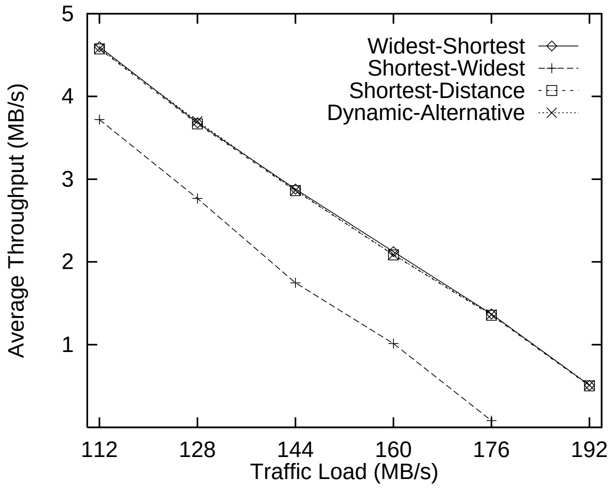


Figure 5.16: Average throughput as a function of network load: cluster topology, evenly distributed load, 60% high-bandwidth best-effort traffic, and 40% video traffic

Figure 5.15 shows the average throughput of all best-effort sessions as a function of the total network load. 60% of traffic is generated by best-effort sessions and we assume that up to 90% of the capacity of each link can be reserved by guaranteed sessions. Both with evenly and unevenly distributed load, very small call blocking rates are observed for the widest-shortest path and the dynamic-alternate path, and no sessions are blocked when the shortest-distance path and the shortest-widest path are used for the guaranteed bandwidth sessions.

We observe that the impact of the routing algorithms for guaranteed sessions on the performance of best-effort sessions is small. However, when the traffic load is heavy and unevenly distributed, we see much lower average throughput for the shortest-widest path. This is because the shortest-widest path consumes more resources for guaranteed sessions, which leaves fewer resources for best-effort sessions.

Figure 5.16 shows the average throughput of best-effort sessions for the Cluster topology. We observe much worse performance for best-effort sessions when the shortest-widest path is used for guaranteed sessions, while the other three algorithms result in similar performance for the best-effort traffic.

We conclude that the use of the shortest-distance path, the widest-shortest path, and the dynamic alternate path for guaranteed sessions results in better performance for best-effort sessions than the shortest-widest path.

5.7 Related Work

Many papers in the literature have studied QoS routing and path selection algorithms. We list the results that are most relevant to our study. A good introduction to QoS routing and routing in general can be found in [63, 95]. QoS routing in telecommunication networks has been an active area of research for a long time. Trunk Reservation and Dynamic Alternate Routing [35], and Least Load Routing [45] are some of the algorithms that have been studied.

In [102], Wang and Crowcroft propose the shortest-widest path algorithm as a way of minimizing the call blocking rate, but no performance evaluation is given. In [13], an adaptive load-based source routing algorithm for traffic requiring bandwidth guarantees is suggested by Breslau, Estrin, and Zhang. Its performance is compared with the static minimal-hop path and the alternate path algorithms. In their study [34], Gawlick, Kalmanek, and Ramakrishnan present an evaluation study of several routing algorithms, including minimal-hop path, exponential path, and max-min path, for traffic with bandwidth guarantees. However, requests are assumed to have infinite call holding times. In his thesis [87], S. Rampal evaluates the performance of several path selection algorithms, including static minimal-hop path, dynamic shortest-distance paths based on link utilization and residual bandwidth, and shortest-delay path. The study also assumes that the session holding time is infinite. Matt and Shankar study delay and throughput based type-of-service routing [72] and dynamic routing of real-time virtual circuits [71] .

Guerin, Orda, and Williams [44] propose to use the widest-shortest path to extend OSPF for QoS routing. In another paper [43], Guerin and Orda study routing with inaccurate information and show that, using a shortest-path algorithm, they can find the feasible path that is most likely to accommodate the requested bandwidth. Neither of these studies address resource utilization efficiency and no performance evaluation is given.

Compared with these studies, we evaluate a wider range of algorithms under more realistic traffic loads: audio and video traffic, long-tail distribution for the call holding times, and even and uneven traffic load distributions. We also consider the effect of routing information propagation and the use of pre-computed paths on performance. Through an extensive set of simulations, we are able to characterize how a routing algorithm can achieve high network throughput: it is important to distribute the load evenly when the network load is light and to conserve resources when the network load is heavy.

5.8 Summary

This chapter presents a simulation study of QoS routing for traffic requiring bandwidth guarantees. While selecting a feasible path, i.e. a path that meets the bandwidth requirement, can be done using any shortest path algorithm, selecting paths that also optimize overall network performance by utilizing resources efficiently is not well understood.

Conserving resources and balancing the network load are two mechanism that can achieve network resource utilization efficiency. Our evaluation considers four routing algorithms that attach different weight to conserving resources and balancing load: the widest-shortest path, shortest-widest path, shortest-distance path, and dynamic-alternate path. Since requests can specify different bandwidth requirements, we use the bandwidth blocking rate as our main performance metric.

Our results show that for dynamic on-demand routing, restricting the hop count (e.g., the dynamic-alternate path) gives better performance when the network load is heavy, while giving preference to balancing load (e.g., the shortest-distance path and shortest-widest path) pays off when the load is light. This result is different from what we observed for best-effort traffic in Chapter 4, where the shortest-distance path algorithm has a clear performance edge over the other algorithms. This difference is caused by the different resource sharing policies of the two traffic classes. Heavily loaded links become automatically ineligible for guaranteed sessions, causing any dynamic algorithm to route around them, but these links remain eligible for best effort traffic so the routing algorithm has to explicitly avoid them. Our results also show that algorithms that minimize the hop count result in a more even blocking rate across sessions with diverse bandwidth requirements. When comparing static and dynamic routing, we observed that using dynamic information can reduce the blocking rate significantly in cases of unevenly distributed load, because dynamic information makes it possible to route around (infeasible) bottleneck links. We also examined a class-based routing algorithm and found that its performance is comparable to that of computing paths on-demand.

An evaluation of the impact of inaccurate routing information and connection setup delay showed that the overall blocking rate is fairly insensitive to the increase in routing information update interval although more sessions are misrouted. Algorithms that minimize the hop count are more robust with respect to increases in the routing information update interval.

To best of our knowledge, our study is the first to consider the performance impact on best-effort traffic when evaluating routing algorithms for guaranteed sessions. We showed that the shortest-widest path algorithm takes away more bandwidth from best-effort sessions and results in reduced performance for best effort sessions.

Overall, we recommend that the shortest-distance path should be used, since it performs consistently well for both evenly and unevely distributed load and for both light and heavy load. Moreover, this algorithm can effectively support dynamic inter-class resource sharing as we will discuss in Chapter 7.

Chapter 6

Routing Traffic with Delay Guarantees

Interactive real-time applications, such as Internet telephony and video conferencing, have stringent QoS requirements on delay, delay jitter, and packet loss. These QoS requirements impose strict resource constraints, e.g., bandwidth and buffer space, on paths being selected to ensure guaranteed end-to-end QoS guarantees. In this chapter, we study routing algorithms for the class of traffic that requires delay guarantees. The QoS constraints include delay, delay-jitter, bandwidth, and buffer space. Similar to routing traffic that requires bandwidth guarantees, the goal of routing traffic with delay guarantees is to find a feasible path if one exists, and to select one that achieves high network throughput if more than one such path is available. However, since satisfying delay guarantees involves meeting multiple QoS constraints, routing traffic with delay guarantees is much harder than routing traffic with bandwidth guarantees.

The general routing problem with multiple QoS constraints is NP-complete. We focus on a network whose service disciplines are rate proportional. The results in this chapter have been published in [68] for finding feasible path and in [69] for finding efficient paths.

The rest of the chapter is organized as follows. Section 6.1 motivates our study. Section 6.2 discusses rate-proportional service disciplines. Polynomial routing algorithms are developed in Sections 6.3 and 6.4. We examine the performance of these algorithms for guaranteed sessions in Section 6.5, and their performance impact on best-effort sessions in Section 6.6. In Section 6.7 we present and evaluate our approximation algorithms. Related work is discussed in Section 6.9. We summarize in Section 6.10.

6.1 Motivation

It is shown in [33, 102] that the general problem of finding a path with multiple QoS constraints, including delay and delay jitter, is NP-complete. However, there are two limitations in existing studies of QoS routing algorithms. First, these studies assume that multiple QoS constraints are unrelated and, therefore, need to be constrained independently. Second, the delay and delay jitter of a link are assumed to be known *a priori*. As a result of these two assumptions, QoS routing becomes a path-finding problem with multiple constraints and is computationally intractable.

We take a rather different approach in our study. First, we observe that the QoS constraints are not independent. For example, the worst case delay jitter of a flow is bounded by the maximal queueing delay of all packets of the flow. Also, both delay and delay jitter are closely related to the amount of reserved bandwidth. The exact relationship is determined by the packet scheduling algorithms deployed in the network. Second, the delay and delay jitter of a link is not known *a priori*. We know that the link delay includes propagation delay, transmission delay, and queueing delay. While the propagation delay and the transmission delay are related to link distance and capacity, the queueing delay is determined not only by the amount of bandwidth reserved for the flow but also the burstiness of the traffic sources. However, the burstiness is specified in the user's traffic specification and can be different for different flows. Thus, both scheduling algorithms deployed in the network and the traffic source specification must be taken into consideration while making a routing decision.

The question now is what scheduling algorithms should be considered for QoS routing. To meet the challenges of achieving delay guarantees in packet-switched network, a substantial amount of research has focused on defining new packet scheduling disciplines. An important class of new packet service disciplines called rate-proportional service disciplines has been developed and extensively studied [77, 17, 105, 98, 39]. Under these service disciplines, packets of different flows are sent in an order to ensure a weighted fair sharing of the link capacity. As a result, a rate is guaranteed for each flow and mathematically provable bounds exist for delay, jitter, and buffer space, if the traffic source conforms to its traffic specification given by a leaky bucket. Based on these new service disciplines, an architecture and mechanism to support real-time applications has been proposed [17]. New service models providing QoS guarantees are being developed both by the IETF [93] and by the ATM Forum [4].

With the rate-proportional service disciplines deployed in the network, we show that QoS routing is polynomial if one takes into consideration the relationship between the multiple QoS constraints, the queueing delay determined by the service disciplines, and the traffic specifications. Using an iterative Bellman-Ford (IBF) algorithm with hop-count constraints, a feasible path with bandwidth, delay, delay-jitter, and/or buffer space constraints can be found if it exists.

To address resource efficiency, we identify among all feasible paths four "optimal" paths based on four different optimality criteria: bandwidth to reserve, hop count, path load, and end-to-end delay. All four optimal paths can be found by IBF based routing algorithms. Their performance is compared through an extensive set of simulation. Furthermore, we introduce approximation algorithms that have execution times within a constant factor of the Bellman-Ford algorithm, and that achieve resource utilization efficiency similar to that of the original algorithms.

6.2 Rate-proportional Service Disciplines

One of the most important issues in providing guaranteed performance services is the choice of the packet service discipline at the switch. Recently, a number of new service disciplines aimed at providing per-connection performance guarantees have been proposed. Examples include Virtual Clock (VC) [106], Weighted Fair Queueing (WFQ) [26, 77], Worst-case Weighted Fair Queueing (WF^{2}Q) [6], Self Clocked Fair Queueing (SCFQ) [38], and Frame-Based Fair Queueing [97]. Under these service disciplines, packets from different connections that share the same output link are sent in an order to ensure a weighted fair sharing of the link capacity. As a result, a guaranteed flow is isolated from others. The end-to-end queueing delay of the flow is determined by the bandwidth being reserved and the burstiness of the traffic source.

To provide QoS guarantees, a flow specifies its traffic characteristics. The network, on the other hand, reserves a certain amount of resources at each switch node on its path. Thus, the traffic specification and the QoS guarantees constitute a "contract" between the network and the applications: The network guarantees that as long as the traffic source conforms to its traffic specification, its QoS requirements will be met. A commonly used traffic specification is a leaky-bucket $\langle\sigma, b\rangle$, where σ is the average token rate and b is maximal number of tokens the session can have. In other words, the total number of bytes allowed to enter the network during any time interval $(t_0, t]$ is bounded by

$$A(t_0, t) \leq b + \sigma(t - t_0).$$

Given a path $\mathbf{p}$ of n hops with the link capacity C_i at hop i. For the broad class of packet service disciplines listed above, if the traffic source is constrained by the leaky bucket $\langle\sigma, b\rangle$, the provable end-to-end delay bound is given by ([105, 98])

$$D(\mathbf{p}, r, b) = \frac{b}{r} + \frac{n \cdot L_{\text{max}}}{r} + \sum_{i=1}^{n} \frac{L_{\text{max}}}{C_i} + \sum_{i=1}^{n} \text{prop}_i \tag{6.1}$$

where r ($r \geq \sigma$) is the amount of bandwidth to be reserved and L_{max} is the maximal packet size in the network, and prop_i is the propagation delay. The end-to-end delay-jitter is bounded by

$$J(\mathbf{p}, r, b) = \frac{b}{r} + \frac{n \cdot L_{\text{max}}}{r}. \tag{6.2}$$

The buffer space requirement at the h-th hop is

$$B(\mathbf{p}, b, j) = b + h \cdot L_{\max}. \tag{6.3}$$

We see that the delay-jitter bound is the maximal queueing delay a packet may encounter. The buffer space only depends on the burstiness of the traffic source and the depth of the node in the path.

6.3 Selecting Feasible Paths

A path is *feasible* if it meets the delay, delay-jitter, and buffer space requirements given in Equations 6.1, 6.2, and 6.3, respectively. The challenge in routing traffic with delay guarantees is that the delay depends not only on the path being selected but also on the bandwidth being reserved, so the traditional routing algorithms (e.g Dijkstra and Bellman-Ford) do not apply. Note that these algorithms are sufficient for traffic with bandwidth guarantees [67]. Moreover, the amount of bandwidth needed to ensure the end-to-end delay is usually not known *a priori*, but has to be determined based on the final path being selected, the delay bound requested by the application, and the burstness of the traffic source. Thus, there are two cases to consider. First, the bandwidth r to be reserved is known *a priori*. Second, r is not known and has to be calculated by the routing algorithm. The main results of this section are summarized in the following theorem:

Theorem 1 *The QoS routing problem of finding a path with delay, delay-jitter, and/or buffer space constraints is solvable in polynomial time. If the bandwidth to be reserved is known* a priori*, a slightly modified version of the Bellman-Ford algorithm can solve it in $S = O(m \cdot L)$, where m is the number of nodes and L the number of links in the network. If the bandwidth to be reserved is unknown, an algorithm that iterates the modified version of the Bellman-Ford algorithm can solve it in $E \cdot S$, where E is the number of different link residual bandwidth values in the network.*

Note that the maximal buffer space requirement of a path is determined by the path hop count. Thus, selecting a path with the minimum hop count reduces the maximal buffer space consumption. The following lemma will be frequently referenced in our discussion of path selection algorithms.

Lemma 1 *Given a path-length function $l(P) = \sum_{i \in P} l(i)$ with $l(i) > 0$ for all links i, a bound* **d***, and a hop bound N. Finding a path P from a source s to a destination d with $l(P) \leq$* **d** *and no more than N hops can be solved by the Bellman-Ford Algorithm in $O(N \cdot E)$, where E is the number links in G.*

Proof. The Bellman-Ford algorithm [7] finds a shortest path step by step with increasing hop count: At the i-th step, a shortest path with at most i hops is found. The total number of steps is restricted to min$\{m, N\}$, where m is the number of nodes in the network. The first feasible path found is the one with the minimum hop count. □

6.3.1 Delay

The problem of finding a path that meets a given end-to-end delay bound is formulated as follows.

Path Finding Problem (D-r): *Given a delay bound* **d**, *a leaky bucket* $\langle\sigma, b\rangle$, *and a bandwidth* r *(*$r \geq \sigma$*) to reserve, find a path* **p** *with* $r \leq R_j$ $(\forall j \in \mathbf{p})$, *such that* $D(\mathbf{p}, r, b) \leq \mathbf{d}$, *where* R_j *is the residual bandwidth of link* j.

Path Finding Problem (D): *Given a delay bound* **d** *and a leaky bucket* $\langle\sigma, b\rangle$, *find a path* **p** *and the amount of bandwidth* $r \geq \sigma$ *to reserve, such that* $\sigma \leq r \leq R_j$ *for all* $j \in$ **p**, *and*

$$D(\mathbf{p}, r, b) \leq \mathbf{d}$$

where R_j *is the residual bandwidth of link j and* $D(\mathbf{p}, r, b)$ *is as defined in Equation (6.1).*

Proposition 1 *QoS routing problem* **(D-r)** *is solvable by both the Dijkstra and Bellman-Ford shortest-path algorithms.*

Proof. Since the amount of bandwidth r to reserve is known, we can easily define an additive distance function by using the link cosk $l(i)$ and length function $l(P)$

$$l(i) = \frac{L_{\max}}{r} + \frac{L_{\max}}{C_i} + \text{prop}_i \quad \& \quad l(P) = \frac{b}{r} + \sum_{j \in P} l(j) \tag{6.4}$$

Find a shortest path P using either Dijkstra or Bellman-Ford algorithm (considering only those links with $R_i \geq \sigma$).

If $l(P) > \mathbf{d}$, there is no path that meets the delay bound **d**. Otherwise, P is a path with the minimum delay. If the Bellman-Ford algorithm is used, the first feasible path found is the one with the minimum hop count. □

Proposition 2 *QoS routing problem* **(D)** *of finding a path with delay constraint is solvable by iterating any single-pair shortest-path algorithm over all values of link residual bandwidth in* $e \cdot S$ *time, where* e *is the number of different values of link residual bandwidth and* S *the time for the shortest-path algorithm. At the iteration of bandwidth value* v_k, *links whose residual bandwidth less than* v_k *are pruned from consideration.*

Proof. The difference with Proposition 1 is that the bandwidth r to be reserved is unknown. For a given path P, the delay can be reduced by increasing r. The maximal reservable bandwidth on path P is $\min\{R_j \mid j \in P\}$. It is clear that a path with the shortest delay must be a feasible path if there is one. Such a path requires reserving the maximum reservable bandwidth

$$\mathbf{mrb_p} = \min\{R_j \mid j \in \mathbf{p}\},$$

to achieve the minimum delay, where **p** is the path and R_j the residual bandwidth, i.e., the amount of reservable bandwidth, of the link j. Note that the $\mathbf{mrb_p}$ must be equal to R_j for some $j \in \mathbf{p}$,

One would expect to use a shortest path algorithm using a length function as in Equation 6.4, setting r to the maximal reservable bandwidth on the partial path. The problem is that the maximal reservable bandwidth changes during the search. An earlier short path may turn into a long path when a link with small residual bandwidth is added to the path. To overcome this, we iterate the shortest-path algorithm over possible choices of link residual bandwidth. At each iteration, a fixed r is used in the length Equation 6.4, and only those links whose residual bandwidth are equal to or higher than r are considered. That is, for every $r = R_k$ of some link k in the network, we define a length function l_r as follows:

$$l_r(i) = \frac{L_{\max}}{r} + \frac{L_{\max}}{C_i} + \text{prop}_i \quad \& \quad l_r(P) = \frac{b}{r} + \sum_{j \in P} l_r(j) \tag{6.5}$$

Using any shortest-path algorithm, we find a shortest path P_r from the source s to the destination d, such that there is no link j in P_r whose residual bandwidth R_j is less than r. We store r, P_r and $l_r(P_r)$ in a vector with E entries. After iterating r over all possible R_k, we search the whole vector to find a path $P_{\min}$ whose length $l_r(P_{\min})$ is minimal. We claim that the path $P_{\min}$ is the shortest delay path if $r = \min\{R_j \mid j \in P_{\min}\}$ is reserved. If it is not, there must be a path P' with a shorter delay than $P_{\min}$. Let r' be $\min\{R'_j \mid j \in P\}$, the maximal reservable bandwidth of path P'. This r' must be the residual bandwidth of some links along path P', and the residual bandwidth of all other links on P' must be no less than r'. This is impossible since $P_{r'}$ stored in the vector is a shorest delay path with $r' \leq R_j$. □

The proof of the propositions is based on finding a path with the shortest delay $D(\mathbf{p}, \mathbf{mrb}, b)$, assuming that **mrb** is reserved. If this delay is larger than the given delay bound **d**, there is no feasible path available. The need to iterate the shortest path algorithm over all different values of link residual bandwidth is caused by the decreasing of the **mrb** during the search. That is, the r in equation 6.1 decreases while the hop count i increases. Hence the delay for the initial segment of the path may no longer be the shortest one.

The algorithm can stop at any time when a feasible path has been found. Having picked a path $\mathbf{p}$, the minimum amount of bandwidth to be reserved to achieve the given delay bound is

$$r = \max\{\sigma,\ (b + n \cdot L_{\max})/(\mathbf{d} - \sum_{i=1}^{n} \frac{L_{\max}}{C_i} - \sum_{i=1}^{n} \text{prop}_i)\}.$$

When the Bellman-Ford shortest path algorithm is used in each iteration, the resulting algorithm is called the Iterative Bellman-Ford algorithm, or IBF. The pseudocode for the IBF algorithm is shown in Figure 6.1, where the procedure Relax() updates existing paths if a shorter one is found (see [20] for details).

```
a. for k ← 1 to e do              /* over different values of link residual bandwidth */
b.     for h ← 1 to m − 1 do      /* over hop count, m is the number of nodes */
c.         for (x, y) ∈ L do      /* over all links (x, y) ∈ G */
d.             Relax(x, y, v_k, ...)
e.     ......                     /* the rest of the program */
f. return
```

Figure 6.1: A sketch of the pseudo-code for the IBF algorithm

6.3.2 Delay and Delay Jitter

The problem of finding a path satisfying both end-to-end delay and delay-jitter bounds can be formulated as:

Path Finding Problem (D-J-r): Given a delay bound $\mathbf{d}$, a delay-jitter bound $\mathbf{j}$, a leaky bucket $\langle\sigma, b\rangle$, and a bandwidth r ($r \geq \sigma$) to reserve, find a path $\mathbf{p}$ with $r \leq R_j (\forall j \in \mathbf{p})$, $D(\mathbf{p}, r, b) \leq \mathbf{d}$, and $J(\mathbf{p}, r, b) \leq \mathbf{j}$, where R_j is the residual bandwidth of link j.

Path Finding Problem (D-J): Given a delay bound $\mathbf{d}$, a delay-jitter bound $\mathbf{j}$, and a leaky bucket $\langle\sigma, b\rangle$, find a path $\mathbf{p}$, such that $D(\mathbf{p}, r, b) \leq \mathbf{d}$ and $J(\mathbf{p}, r, b) \leq \mathbf{j}$ for some r with $\sigma \leq r \leq R_j$ ($\forall j \in \mathbf{p}$), where R_j is the residual bandwidth of link j.

Proposition 3 *QoS routing problem* **(D-J-r)** *is solvable by the Bellman-Ford algorithm in time $m \cdot L$, where m is the number of nodes and L the number of links in the network.*

Proof. We learn from Equation 6.2 that the hop count (n) is the only parameter that determines whether a delay-jitter bound $\mathbf{j}$ can be met:

$$\frac{b}{r} + \frac{n \cdot L_{\max}}{r} \leq \mathbf{j} \quad \text{iff} \quad n \leq \lfloor \frac{r \cdot \mathbf{j} - b}{L_{\max}} \rfloor$$

Thus, for any path, as long as its hop count is no more than $N = \lfloor (r \cdot \mathbf{j} - b)/L_{\max} \rfloor$, it will meet the delay-jitter bound $\mathbf{j}$. We apply Lemma 1 using the distance function defined in Equation 6.4 with delay bound $\mathbf{d}$, and hop count restriction N, to get our result. □

Proposition 4 *QoS routing problem* **(D-J)** *is solvable in $E \cdot S$, where E is the number of possible link residual bandwidths in the network and S is the time for the Bellman-Ford algorithm.*

Proof. Similar to the proof of Proposition 2, we iterate the Bellman-Ford shortest-path algorithm over all possible values of the link residual bandwidth R_k. At each iteration of $r = R_k$, the length function l_r of Equation 6.5 is used. The only difference is that, as in the proof of Proposition 3, we use the hop count $N_r = \lfloor (r \cdot \mathbf{j} - b)/L_{\text{max}} \rfloor$ to control the number of steps. Also, only those links with $R_i \geq r$ are considered in each step. □

6.3.3 Delay and Buffer Space Constraints

The problem of finding a path satisfying both an end-to-end delay bound and buffer space constraints can be formulated as follows.

Path Finding Problem (D-B-r): Given a delay bound $\mathbf{d}$, a buffer space constraint $\mathbf{b}_u$ for each node u, a leaky bucket $\langle \sigma, b \rangle$, and a bandwidth r ($r \geq \sigma$) to reserve. Find a path $\mathbf{p}$, such that $D(\mathbf{p}, r, b) \leq d$, $B(\mathbf{p}, b, h) \leq \mathbf{b}_h$, and $r \leq R_j (\forall j \in \mathbf{p})$, where R_j is the residual bandwidth of link j and $\mathbf{b}_h$ is the buffer space constraint for the node h hops from the source.

Path Finding Problem (D-B): Given a delay bound $\mathbf{d}$, a buffer space constraint $\mathbf{b}_u$ for each node u, and a leaky bucket $\langle \sigma, b \rangle$. Find a path $\mathbf{p}$ such that $D(\mathbf{p}, r, b) \leq \mathbf{d}$, and $B(\mathbf{p}, b, h) \leq \mathbf{b}_h$, for some r with $\sigma \leq r \leq R_j$ ($\forall j \in \mathbf{p}$), where R_i is the residual bandwidth of link j and $\mathbf{b}_h$ is the buffer space constraint for the node h hops away from the source.

Proposition 5 *QoS routing problem* **(D-B-r)** *is solvable by a modified version of the Bellman-Ford algorithm in $O(m \cdot L)$, where m is the number of nodes and L the number of links in the network.*

Proof. For each node u in the network, the buffer space constraint $\mathbf{d}_u$ defines a bound on the hop count $N_u = \lfloor (\mathbf{d}_u - b)/L_{\text{max}} \rfloor$, such that node u cannot appear in a path more than N_u hops away from the source. We define the same length function l as in Equation 6.4 and apply the Bellman-Ford shortest-path algorithm. During step h, we consider only those nodes with $N_u \geq h$. Finally, we select a path from the result paths of all steps. To conclude the proof, we need to show that if there exists a path P' from s to d that satisfies the delay $\mathbf{d}$ and hop count constraints N_j for all nodes j on the path, the modified Bellman-Ford algorithm must find a path P that has no more hops than P', $l(P) \leq l(P')$, and P satisfies the hop count constraints for all nodes on the path. We use induction on h, the hop count of P'. The results clearly applies for $h = 1$. Assuming the result applies for h, we have to show that it applies for $h + 1$. Let w be the last node on the path P' prior to the destination of P', and Q' the path obtained by removing the last hop from P'. Using the induction hypothesis, there exists a path Q such that Q has no more hops than Q', $l(Q) \le l(Q')$, Q satisfies the hop count constraints, and Q is the shortest path with no more than the h hops found by the Bellman-Ford algorithm. Since the path concatenating Q and the link from w to d is a possible choice, the Bellman-Ford algorithm will find a path P with at most $(h+1)$ hops at the $(h+1)$-th iteration, such that P satisfies hop count constraints and

$$l(P) \le l(Q) + l(w,d) \le l(Q') + l(w,d) = l(P'). \qquad \square$$

Proposition 6 *QoS routing problem* **(D-B)** *is solvable in $O(m \cdot E^2)$, where m is the number of nodes and E the number of links in the network.*

Proof. Similar to the proof of Proposition 2, we iterate the modified version of the Bellman-Ford shortest-path algorithm in the proof of Proposition 5 over all possible values of the link residual bandwidth R_k. At each iteration of $r = R_k$ for some link k, the same length function l_r as in the proof of Proposition 2 is used. As in the proof of Proposition 5, for every node u, we use the hop count constraint N_u to achieve the buffer space constraint $\mathbf{b}_u$, and let the Bellman-Ford shortest-path algorithm search only those nodes whose hop count bound are larger than the length of the path currently being considered and only those links with $R_i \ge r$. $\square$

6.3.4 Delay, Delay Jitter, and Buffer-Space Constraints

The problem of finding a path satisfying delay, delay-jitter, and buffer space constraints can be formulated as follows.

Path Finding Problem (D-J-B-r): Given a delay bound $\mathbf{d}$, a delay-jitter bound **j**, and buffer space constraints $\mathbf{b}_u$ for all the node u, a leaky bucket $\langle \sigma, b \rangle$, and a bandwidth r ($r \ge \sigma$) to reserve, find a path **P**, such that $D(\mathbf{p}, r, b) \le d$, $J(\underline{\mathrm{p}}, r, b) \le \mathbf{j}$, $B(\mathbf{p}, b, h) \le \mathbf{b}_h$ for all nodes on the path, and $r \le R_j$ ($\forall j \in \mathbf{p}$), where R_j is the link residual bandwidth and $\mathbf{b}_h$ is the buffer space constraint for the node with h hops from the source.

Path Finding Problem (D-J-B): Given a delay bound $\mathbf{d}$, a delay-jitter bound **j**, and buffer space constraints $\mathbf{b}_i$ for all the node i, and a leaky bucket $\langle \sigma, b \rangle$. Find a path **p** with the amount of bandwidth r to reserve, such that $\sigma \le r \le R_j$ for all $j \in \mathbf{p}$, and

$$\begin{aligned} D(\mathbf{p}, r, b) &\le \mathbf{d}, \\ J(\mathbf{p}, r, b) &\le \mathbf{j}, \\ B(\mathbf{p}, b, h) &\le \mathbf{b}_h, \end{aligned}$$

where the third inequality holds for all nodes on the path.

Proposition 7 *QoS routing problem* **(D-J-B-r)** *is solvable by a modified version of the Bellman-Ford algorithm in $O(m \cdot L)$, where m is the number of nodes and L the number of links in the network.*

Proof. To meet the delay-jitter bound, we only need to control the number of steps in the algorithm in the proof of Proposition 5. □

Proposition 8 *QoS routing problem* **(D-J-B)** *is solvable in $E \cdot S$, where E is the number of all possible residual bandwidth of links in the network and S the time for the Bellman-Ford algorithm.*

Proof. The proof is similar to the proof of Proposition 6, except that the delay-jitter bound is used to limit the number of iterations. □

6.4 Selecting Efficient Paths

The IBF algorithm discussed above can find a feasible path. In practice, more than one feasible path is often available. This raises the question of what path to use to achieve higher network throughput (or lower blocking rate). We propose four different optimality criteria that can be used in path selection and present polynomial routing algorithms that select paths with these different optimality properties.

6.4.1 Alternatives Optimal candidates

To select an "optimal" path, one must first decide what optimality criterion should be used. The goal is to reduce the probability that a future arrival is blocked because no feasible path can be found and, therefore, to achieve high network throughput. There are two common ways to achieve low call blocking rate, one is to conserve per-flow resource utilization and the other is to distribute the load evenly through the network. Based on this observation, we identify four candidate criteria of optimality:

- **Minimum bandwidth consumption.** The traffic specification gives only the minimal bandwidth requirement (the token rate σ). More bandwidth may have to be reserved in order to achieve the requested end-to-end delay. How much more to reserve depends on the path being selected. An optimal path is the one requiring the reservation of the minimum amount of bandwidth.

- **Minimum hop count.** The network is a resource pool shared by many users. New traffic can enter the network from any switch node at any time. A path with fewer hops consumes less network resources than a path with more hops. An optimal path is the one with the minimum hop count.

- **Minimum path load.** Selecting a path with high **mrb** can avoid using congested links. An optimal path is the one with the maximum **mrb**.

- **Minimum end-to-end delay.** Selecting a path with the minimum end-to-end delay if the **mrb** of the path is reserved is not an obvious optimality criterion. This choice is motivated by the form of Equation (6.1). It suggests that a path with the minimum delay is likely to have few hops and a relatively high **mrb**. In other words, the minimum delay path is unlikely to be long (high hop count) or congested (low **mrb**). Note that selecting a path with the shortest delay does not require that the **mrb** is reserved.

For each of these optimality criteria, or some combination with different priority, we can select a path in the hope of achieving high resource utilization efficiency. We select the following candidates:

- **Minimum-bandwidth path**—a feasible path that requires the reservation of the minimum amount of bandwidth. If there is more than one choice, the one with the minimum hop count is selected.

- **Widest-shortest path**—a feasible path with the minimum hop count. If there is more than one choice, the one with the maximum reservable bandwidth is selected.

- **Shortest-widest path**—a feasible path with the maximum reservable bandwidth. If there are several such paths, the one with the minimum hop count is selected.

- **Shortest-delay path**—a feasible path giving the minimal end-to-end delay if the maximal reservable bandwidth is reserved. If there are several such paths, the one with the minimum hop count is selected.

Note that other optimality criteria are possible. For example, selecting a path with minimum delay-jitter or buffer space. However, a widest-shortest path is likely to give the minimum delay-jitter and buffer space. We selected these four algorithms because they represent a broad spectrum of tradeoffs between resource conservation and network load distribution, as is shown below:

minimum-bandwidth	widest-shortest	shortest-delay	shortest-widest
$\longleftarrow$ *resource conserving*			*load balancing* $\longrightarrow$

6.4.2 Algorithms

We present algorithms that select "optimal" paths based on each of the above optimality criteria. Recall that the IBF algorithm has two levels of iterations: the outer loop (line (a) in Figure 6.1) over different values of link residual bandwidth and the inner loop (line (b) in Figure 6.1) over the hop count. For every value of link residual bandwidth and every hop count, a feasible path may be found. We store all these paths in an array as follows:

bandwidth value	hop 1	$\cdots$	hop $m-1$
v_1	$\mathbf{p}_{1,v_1}$		$\mathbf{p}_{m-1,v_1}$
$\cdots$			
v_e	$\mathbf{p}_{1,v_e}$		$\mathbf{p}_{m-1,v_e}$

We show that, all four paths with the different optimality criteria can be found from the feasible paths recorded in the table. In other words, we can have polynomial algorithms to select any of the four optimal paths.

Clearly, the IBF algorithm can find the shortest-delay path by remembering the delay of every path found in every iteration for different bandwidth values and hop counts. After all iterations have been completed, the one with the shortest delay can be selected by comparing all paths in the table.

To find a minimum-bandwidth path, we compare all feasible paths in the table, and select the one requiring the reservation of the minimum bandwidth. We claim that there is no other path requiring less bandwidth that meets the delay bound. If there is one, let v_i be the **mrb** of that path and let h be the hop count. Remember that the path in the entry with hop count h and the bandwidth value v_i is the one with the minimum end-to-end delay if the **mrb** v_i is reserved. Since both paths have the same **mrb** and the same hop count, the one requiring the reservation of the least bandwidth introduces the shortest end-to-end delay when the **mrb** is reserved according to Equation (6.1). This contradicts the fact that the path in the entry is the one with the shortest delay if the **mrb** is reserved.

To find a widest-shortest path, we can search the table column by column. In each column we search the entries according to the decreasing order of bandwidth values. The first feasible path found is a widest-shortest path. It is clear that there is no other feasible path not listed in the table but with fewer hops or with the same hop count but higher **mrb**. Similarly, a shortest-widest path can be found by searching the table row by row.

The shortest-delay path and minimum-bandwidth path algorithms require that the entire table is constructed, *i.e.*, all iterations have to be executed. In contrast, the shortest-widest path and the widest-shortest path algorithms can be implement more efficiently by ordering the different bandwidth values in decreasing order. To select a shortest-widest path, the algorithm can terminate when the first feasible path is found. To select a widest-shortest path, in addition

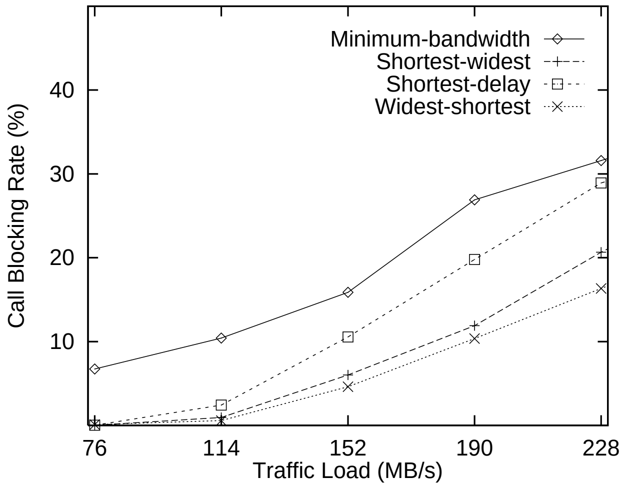
```
Widest_Shortest_Path(G : GRAPH, s : NODE, d : NODE, d : real, j : real, b : NODE → int)
r_vec ← Sorting(L)                          /* sort R_i (i ∈ L) in decreasing order, delete duplicates */
E ← lenght of r_vec                         /* the number of different link residual bandwidth */
for v ∈ V do                                /* for all nodes in G */
      N_v ← ⌊(b(v) − b)/L_max⌋               /* hop count bound for node v by buffer space */
      max_hop = max(max_hop, N_v)
m ← min(|V|, max_hop)                       /* |V| is the number of nodes in G */
Initialize-Single-Source(G, s, E, r_vec)    /* initialization*/
for h ← 1 to m − 1 do                       /* iteration over hop count */
      for k ← 1 to E do                     /* iteration over different values of link residual bandwidth */
            if (n_k ≥ h) then               /* if h meets the hop count bound set by j */
                  for (u, v) ∈ L do         /* for all links in G */
                        Relax(u, v, k, l_k, h, N_v)
            if l_k(d) ≤ d then
                  return π_k                /* path found */
return FALSE                                /* no feasible path */

Initialize-Single-Source(G, s, E, r_vec)
for k ← 1 to E do                           /* iteration over different values of link residual bandwidth */
      n_k ← ⌊(r_vec[k] · j − b)/L_max⌋       /* bound on hop count determined by j */
      for v ∈ V do                          /* for all nodes in G */
            l_k[v] ← ∞                      /* for r = r_vec[k] */
            π_k[v] ← NIL                    /* initialize path */
return

Relax(u, v, k, l_k, h, N_v)
if N_v ≤ h and R_(u,v) ≥ r_vec[k] then      /* check hop count bound and link residual bandwidth */
      if l_k[v] > l_k[u] + l_k(u, v) then
            l_k[v] ← l_k[u] + l_k(u, v)
            π_k[v] ← u                      /* update the path */
return
```

Figure 6.2: Pseudo-code for the widest-shortest path algorithm

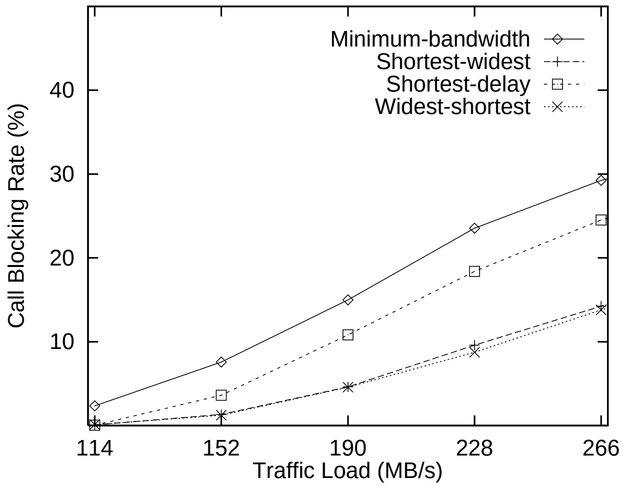
to ordering the residual bandwidth values, we can swap the loop order in lines (a) and (b) in Figure 6.1. With this slight modification, the first feasible path found is a widest-shortest path. Figure 6.2 gives the pseudocode for the widest-shortest path algorithm.

6.5 Performance of Guaranteed Sessions

In this section, we examine the performance of the four candidate "optimal" paths. The focus is on understanding the performance of different routing algorithms under a heavy load of guaranteed sessions. Under a light load, all four algorithms achieve a zero call blocking rate.



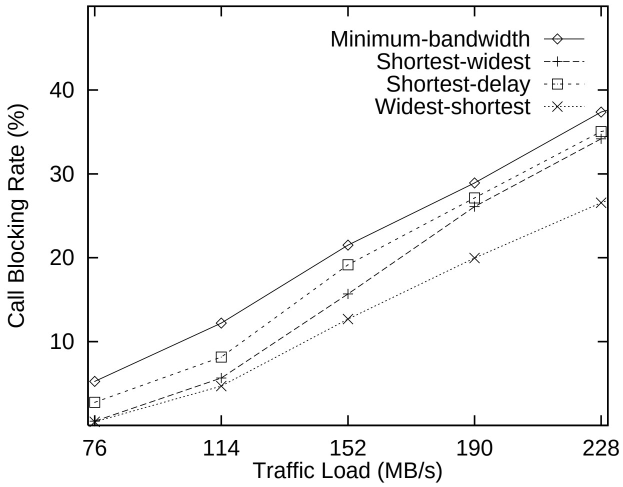
(a) b = [4KB, 8KB] and **d** = [80ms, 120ms]



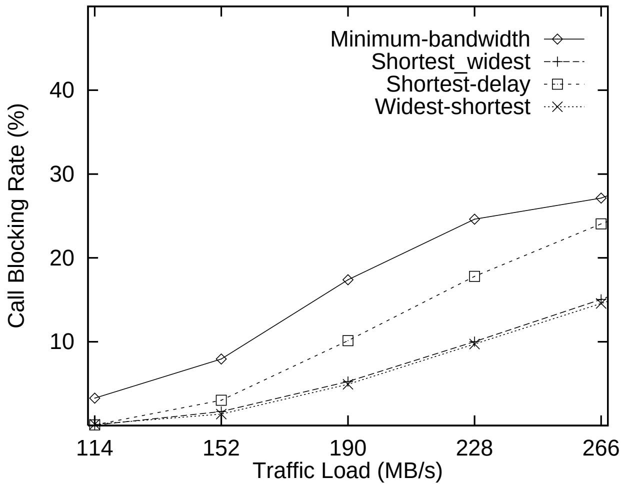
(b) b = [4KB, 8KB] and **d** = [200ms, 240ms]

Figure 6.3: Call blocking rate as a function of network load: MCI topology, 100% guaranteed sessions, and evenly distributed load

Only guaranteed sessions are considered in this section. Our evaluation in this chapter uses a default routing update interval of 30 seconds.



(a) b = [16KB, 20KB] and **d** =[80ms, 120ms]



(b) b = [16KB, 20KB] and **d** =[200ms, 240ms]

Figure 6.4: Call blocking rate as a function of network load: MCI topology, 100% guaranteed sessions, and evenly distributed load

6.5.1 Evenly Distributed Load

Figures 6.3 and 6.4 show the call blocking rate as a function of the network load for the MCI topology for four different combinations of traffic burstiness and delay bounds.

Overall, we see that the performance difference between the routing algorithms can be large: up to a factor of two. This implies that selecting a good routing algorithms is very important.

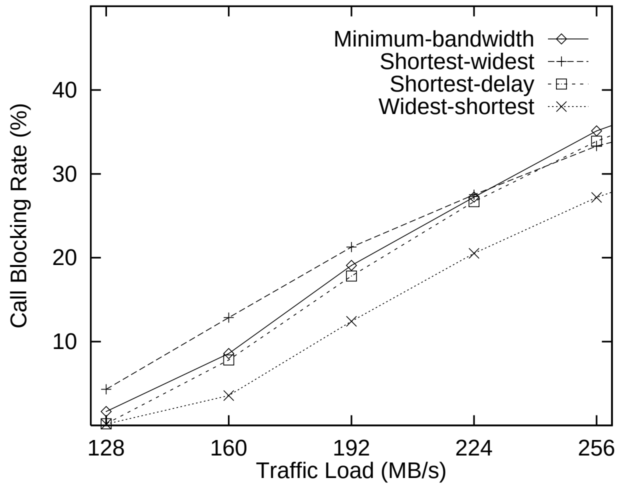
We also see (comparing the top two figures and the bottom two figures) that the call blocking rate is higher when the delay bound is tighter and when the traffic is more bursty. This is to be expected: more bandwidth has to be reserved to limit the queueing delay. When the load is light, all algorithms except the minimum-bandwidth path converge quickly and result in a zero call blocking rate.

In comparison, the widest-shortest path performs better than the other three algorithms and the minimum-bandwidth path performs the worst. The reason is that the widest-shortest path algorithm selects paths with as few hops as possible to conserve resources. In the mean time, it does load balancing by selecting the widest one among all paths with the same hop count. On the other hand, the minimum-bandwidth path, while it also tries to conserve resources, does not take the current load of the path being selected into account, and may pick a loaded path, therefore, blocking future arrivals. It is interesting to see that the shortest-widest path performs reasonably well and has a performance similar to the widest-shortest path when the delay bounds are between 200ms and 240ms. This suggests that the load is an important factor when selecting a path. The performance for the shortest-delay path is interesting. When the load is light, its performance converges to that of the widest-shortest path. When the load is heavy, its performance moves closer to that of the minimum-bandwidth path. This can be explained using Equation (6.1). If the load is light, r is high, and the second two terms will dominate, so minimizing hop count is important. When the load is heavy, r is small and n carries less weight than r in determining the final end-to-end delay. This results in selecting a path with higher r but with potentially more hops.

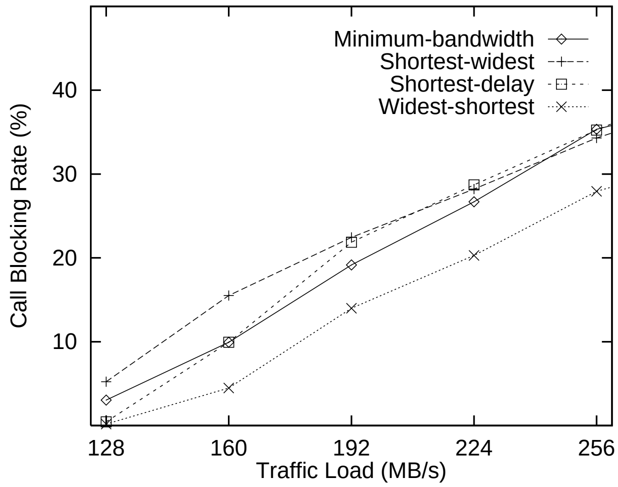
Figure 6.5 shows the call blocking rate for the switched cluster topology. We observe similar behavior for the four routing algorithms as for the MCI topology. The main difference is that the shortest-widest path does not perform as well as for the MCI topology. In some scenarios, it is worse than the minimum-bandwidth path. Thus, the shortest-widest path is sensitive to the topology change as we already observed for best-effort traffic [70]. The reason is that, for the MCI topology, a shortest-widest path is often a shortest path that consists of only OC3 links. For the switched cluster topology, the shortest-widest path can use a diverse combination of of links, depending on the current load of the links, and as a result, it will often have more hops than a shortest path.

6.5.2 Unevenly Distributed Load

In Figure 6.6, the traffic load is unevenly distributed: 50% of the traffic is between the West coast and East coast and the other 50% is evenly distributed. We continue to see a large performance difference between the four routing algorithms. The widest-shortest path continues to perform well, although in some scenarios, when the load is very light, the shortest-delay path can outperform the widest-shortest path. This suggests that using a path with the minimum hop-count may make it hard to avoid some congested links. It also indicates the importance of



(a) b = [4KB, 8KB] and **d** = [200ms, 240ms]

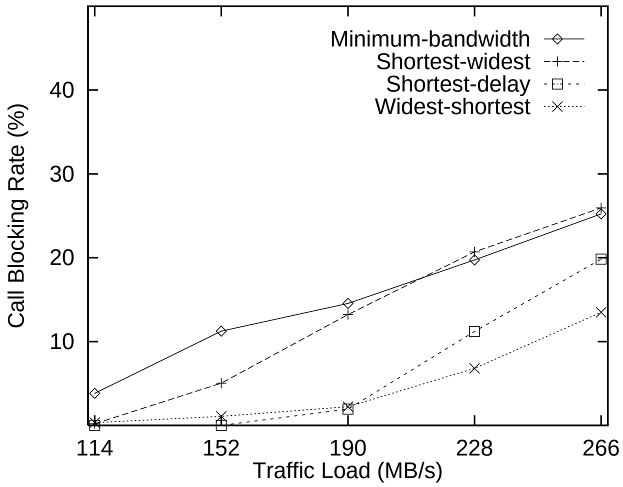


(b) b = [16KB, 20KB] and **d** =[200ms, 240ms]

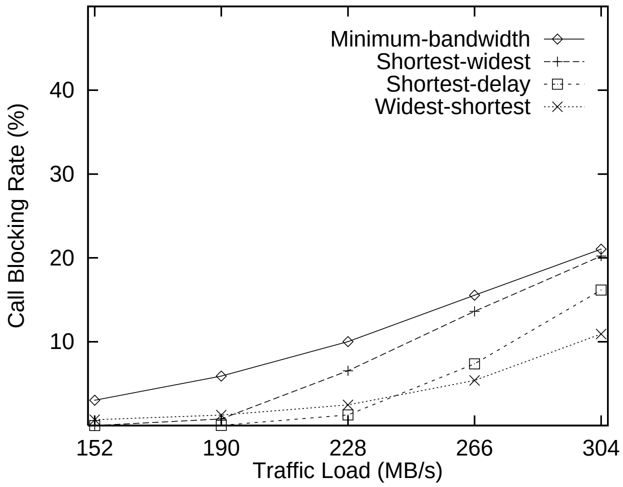
Figure 6.5: Call blocking rate as a function of network load: Switched cluster, 100% guaranteed sessions, and evenly distributed load

selecting an unloaded path rather than a minimum hop path when the load is relative light. The minimum-bandwidth path still performs badly, as was the case with evenly distributed traffic. The shortest-widest path keeps performing inconsistently.

In summary, for evenly distributed loads, the widest-shortest path algorithm outperforms the other three algorithms because it places more emphasis on conserving resources. For unevenly distributed loads, balancing the traffic load becomes more important. The shortest-delay path



(a) b = [4KB, 8KB] and d = [80ms, 120ms]

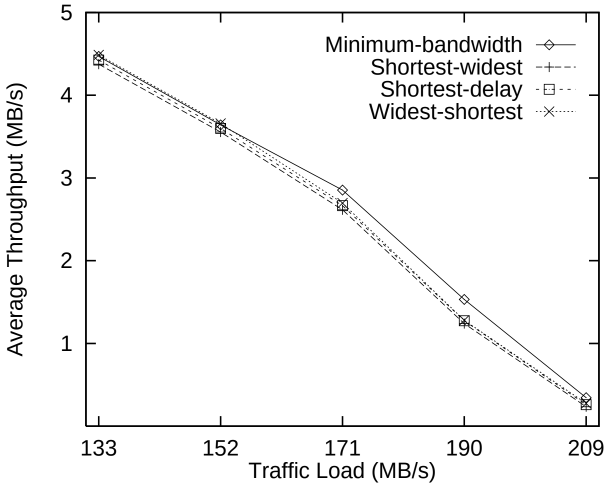


(b) b = [16KB, 20KB] and **d** =[200ms, 240ms]

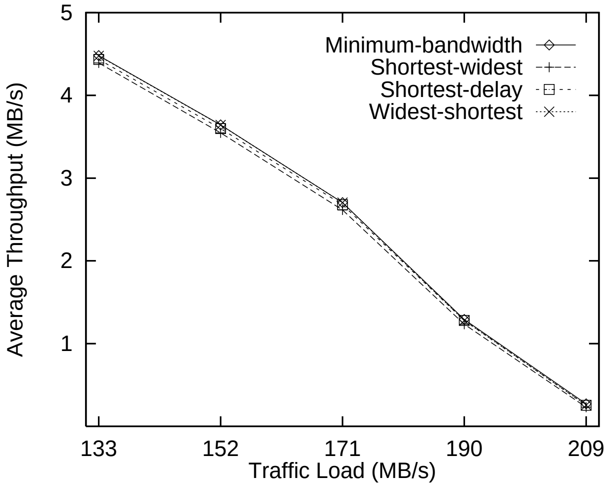
Figure 6.6: Call blocking rate as a function of network load: MCI topology, 100% guaranteed sessions, and unevenly distributed load

algorithm outperforms other algorithms when the load is light. However, when the load is heavy, conserving resources becomes more important, the widest-shortest path outperforms others. The minimum-bandwidth path algorithm performs poorly because it may select heavily loaded paths, while the performance of the shortest-widest path algorithm is very sensitive to the topology and load distribution.

This result for the widest-shortest path is different from what we observed for best-effort



(a) b = [4KB, 8KB] and **d** = [80ms, 120ms]



(b) b = [16KB, 20KB] and **d** =[200ms, 240ms]

Figure 6.7: Average throughput as a function of network load: MCI topology, 40% guaranteed sessions, evenly distributed load, and 90% reservation rate

traffic in Chapter 4, where the shortest-distance path algorithm has a clear performance edge over the other algorithms. This difference is caused by the different resource sharing policies of the two traffic classes. Heavily loaded links become automatically ineligible for guaranteed sessions, causing any dynamic algorithm to route around them. But these links remain eligible for best effort traffic so the algorithm has to explicitly avoid them.

6.6 Performance Impact on Best-effort Traffic

In a network with both guaranteed traffic and best-effort traffic, what path is used for guaranteed traffic can have a significant impact on the performance of best-effort traffic. Thus, it is important to understand this performance impact on best effort traffic before adopting a routing algorithm for guaranteed traffic. For example, when the guaranteed traffic load is relatively low compared to the network capacity, the call blocking rate for guaranteed traffic is likely to be zero, regardless of what routing algorithm is used. In such cases, the throughput for best-effort traffic is the main performance metric distinguishing the routing algorithms for guaranteed traffic.

In this section, we examine the throughput of best-effort sessions in networks supporting both guaranteed traffic and best-effort traffic, for different path selection algorithms for guaranteed traffic. We assume that the best-effort sessions share the unreserved link capacity according to the max-min fair sharing policy defined in [50]. The algorithm used for best-effort sessions is the shortest distance path with the distance for a path defined as

$$\text{dist}(P) = \sum_{i=1}^{k} \frac{1}{r_i}$$

where r_i is the max-min fair share rate a new best-effort session will obtain (see Chapter 4 for details).

6.6.1 Evenly Distributed Load

We first consider the case that the traffic load is evenly distributed. The total traffic load is split between guaranteed sessions and best-effort sessions, with 40% of the traffic for guaranteed sessions. Figure 6.7 shows the average throughput of the best-effort sessions as a function of the total network load. We assume that up to 90% of the link capacity can be reserved for the guaranteed sessions. Overall, all four routing algorithms have a similar performance impact on best-effort sessions, although the minimum-bandwidth path performs slightly better than others. However, this slight performance advantage is achieved at the cost of more guaranteed sessions being blocked. Table 6.1 shows the call blocking rate for guaranteed sessions. A much higher call blocking rate is observed when the minimum-bandwidth path is used, which is not surprising given the results presented in the previous section.

Figure 6.8 shows the performance impact for the switched cluster topology. For all traffic loads in the figure, no guaranteed sessions are blocked. What is surprising is that the shortest-widest path performs much worse than the three other algorithms. This suggests that, for a more symmetric network topology, applying the shortest-widest path algorithms for guaranteed sessions can cause much higher resource consumption, therefore leaving fewer resources for

Traffic load (MB/s)	133	152	171	190	209
Minimum-bandwidth	0.06	0.13	0.08	1.44	3.68
Shortest-widest	0.00	0.00	0.00	0.01	0.07
Shortest-delay	0.00	0.00	0.00	0.01	0.22
Widest-shortest	0.02	0.03	0.03	0.04	0.13

(a) b = [4KB, 8KB] and **d** =[80ms, 120ms]

Traffic load (MB/s)	133	152	171	190	209
Minimum-bandwidth	0.00	0.00	0.01	0.08	0.78
Shortest-widest	0.00	0.00	0.00	0.00	0.01
Shortest-delay	0.00	0.00	0.00	0.00	0.01
Widest-shortest	0.00	0.00	0.00	0.00	0.04

(b) b = [16KB, 20KB] and **d** =[200ms, 240ms]

Table 6.1: Call blocking rate in percentage: MCI topology, 40% guaranteed sessions, evenly distributed load, and 90% reservation rate

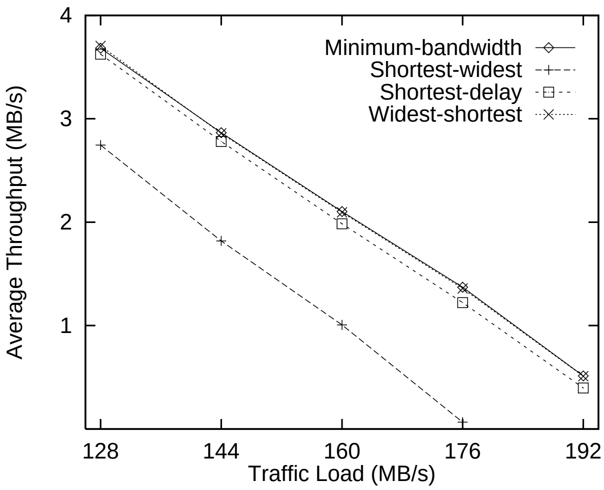


Figure 6.8: Average throughput as a function of network load: Switched cluster, 60% best-effort, 90% link-capacity reservation, evenly distributed load, b = [16KB, 20KB], and d =[200ms, 240ms]

best-effort sessions. This again demonstrates that the performance of the shortest-widest path is inconsistent.

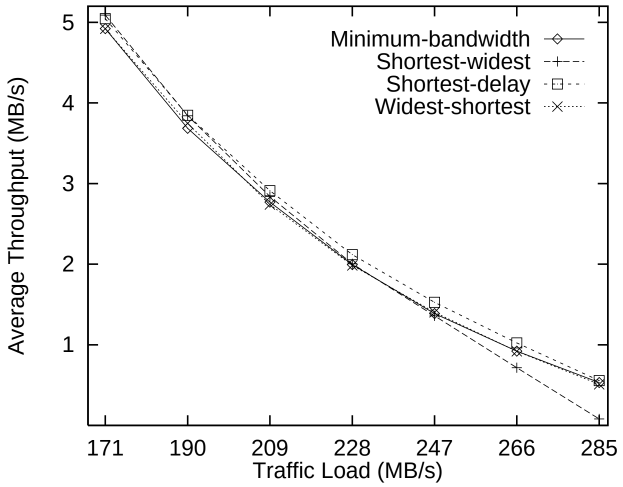


Figure 6.9: Average throughput as a function of network load: MCI topology, 40% guaranteed sessions, 90% link-capacity reservation, unevenly distributed load, b = [16KB, 20KB], and d =[200ms, 240ms]

Traffic load (MB/s)	209	228	247	266	285
Minimum-bandwidth	0.00	0.00	0.07	0.23	1.17
Shortest-widest	0.00	0.00	0.00	0.00	0.00
Shortest-delay	0.00	0.00	0.00	0.00	0.00
Widest-shortest	0.00	0.00	0.06	0.10	0.19

b = [16KB, 20KB] and **d** =[200ms, 240ms]

Table 6.2: Call blocking rate: MCI topology, 40% guaranteed sessions, unevenly distributed load, and 90% reservation rate

6.6.2 Unevenly Distributed Load

We next look at unevenly distributed loads. Figure 6.9 shows the impact of different routing algorithms on the performance of the best-effort sessions for the MCI topology, and Table 6.2 shows the corresponding call blocking rates for the guaranteed sessions. We see again that the shortest-widest path algorithm takes away more resources from best effort sessions, especially when the load is heavy. The minimum-bandwidth path blocks more guaranteed sessions although its performance impact on best-effort sessions is close to the shortest-delay path and the widest-shortest path algorithms. Nonzero call blocking rates are observed for the widest-shortest path because of its limited ability to balance the network load. The shortest-delay path has overall the best performance: no guaranteed sessions are blocked and best-effort sessions achieve a high average throughput.

In summary, the shortest-widest path algorithm tends to select resource-intensive paths for guaranteed sessions, which reduces the throughput of best-effort traffic. The other three routing algorithms result in better and similar performance for best-effort traffic.

6.7 Approximation Algorithms

In this section, we present approximation algorithms for the four candidate "optimal" routing algorithms and evaluate their performance both in terms of running time and in terms of network throughput.

6.7.1 Approximation Scheme

In the previous sections, we evaluated the performance of four different routing algorithms both in terms of the call blocking rate and in terms of the average throughput of the best-effort sessions. These algorithms have a running time $O(m \cdot e \cdot E)$, where m is the number of nodes, e the number of different values of link residual bandwidth, and E the number links in the network. In the worst case, e is equal to E, and the running time is $O(m \cdot E^2)$. The worst case happens in fact frequently. The reserved bandwidths for different sessions are very diverse, so the residual bandwidth of two links will often be different, although the difference may be small. The algorithms have to iterate over all these very close but different bandwidth values. We recall that the original motivation for iterating the Bellman-Ford algorithm over all possible values of the link residual bandwidth was to ensure that no path with the shortest delay is missed, regardless of the amount of bandwidth that has to be reserved.

A first optimization is to use a small set of residual bandwidths, and to approximate the residual bandwidth of each link by rounding down to one of these values. We can then iterate the Bellman-Ford algorithm over this smaller set of values, which enables us to eliminate iterations over those close but different residual bandwidth values.

A second optimization is to have an upperbound on the amount of bandwidth that a flow can reserve. In practice, there must a limit to how much bandwidth can be allocated for a single session without having a negative impact on overall network utilization. For example, if the token rate of a flow is 5 Mb/s, the network should not reserve more than 25 Mb/s of the link bandwidth in order to achieve a delay bound.

Our approximation algorithms select a set of bandwidth values $V = \{v_1, \ldots, v_n\}$ such that $v_1 > \ldots > v_n$. The value v_1 is the maximal bandwidth that can be reserved. For every link, we round its residual capacity v to the largest v_i such that $v_i \leq v$. The Bellman-Ford algorithm is iterated only over those v_i. At each iteration, it assumes that the value of v_i is the reservable bandwidth.

The selection of the set V depends on the link capacity and the distribution of bandwidth requests. In order to select as few values as possible without eliminating many potential feasible paths from consideration, we reduce the distance between two consecutive values as the value decrease. In our simulation, we use the following 14 values:

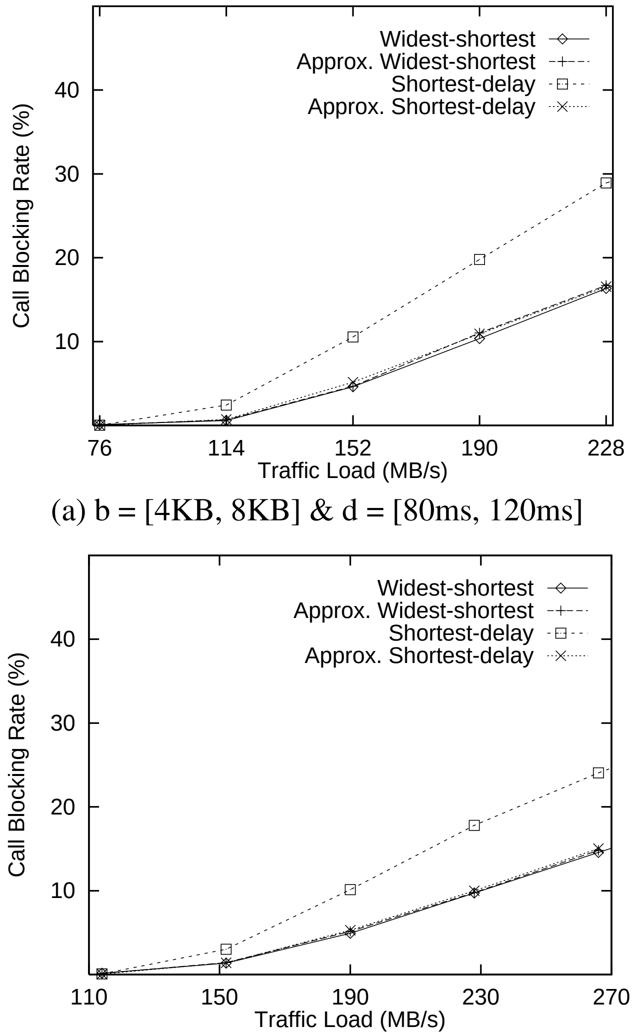
$$V = \{24, 20, 16, 14, 12, 10, 8, 7, 6, 5, 4, 3, 2, 1\}$$

Once the values of V have been selected, the worst case running time of the iterative Bellman-Ford algorithm is $O(|V| \cdot m \cdot E)$, where $|V|$ is a constant. In other words, the approximation algorithm has running times that are within a constant factor of the Bellman-Ford algorithm.

6.7.2 Evaluation of Network Throughput

The simulation results in Sections 6.5 and 6.6 show that the shortest-widest path and the minimum-bandwidth path perform inconsistently for different topologies because they put either too much weight on balancing the network load or on bandwidth consumption. Our evaluation of the approximation algorithms focuses on the shortest-delay path and the widest-shortest path algorithms.

Figures 6.10 and 6.11 show for the MCI topology the performance difference between the widest-shortest path and the shortest-delay path algorithms, and their approximation. Figure 6.10 is for evenly distributed loads, and Figure 6.10 is for unevenly distributed loads. We see that the call blocking rates for the widest-shortest path and its approximation are very similar, while the approximation algorithm for the shortest-delay path performs better than the original "correct" algorithm. This result is somewhat unexpected. Further analysis shows that the improvement is not caused by the limit on the reserved bandwidth, since the exact algorithms rarely allocate high bandwidth anyway. Instead the performance improvement is a result of the use of bandwidth intervals. Recall that the end-to-end delay (see Equation 6.1) is a function of the hop count and reservable bandwidth. By using bandwidth intervals, we reduce the reservable bandwidth to its closest discrete interval value and equalize the link bandwidths with small differences, which leads the shortest-delay path algorithm to select paths with fewer hops, and therefore, to place more emphasis on conserving network resources. When the load is heavy, the reservable bandwidth is small, and reducing the bandwidth is likely to have a bigger impact. However, the approximation scheme has little impact on the widest-shortest path algorithm. The reason is that it is already very biased towards minimizing resource utilization.

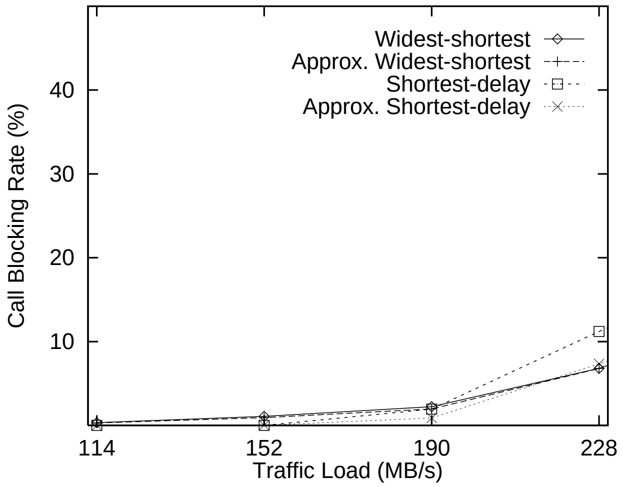


(a) b = [4KB, 8KB] & d = [80ms, 120ms]

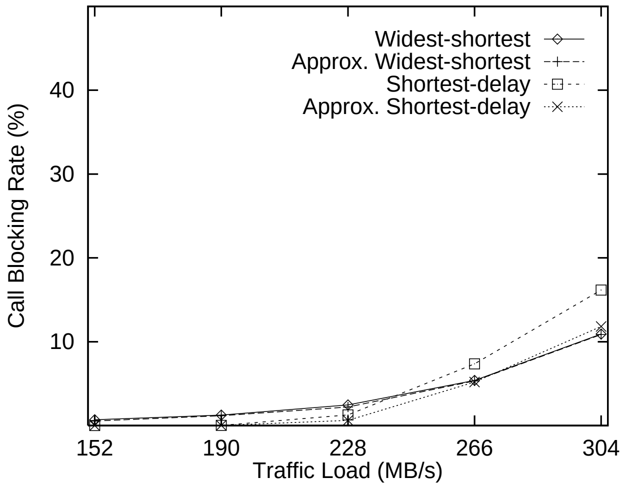
(b) b = [16KB, 20KB] & d = [200ms, 240ms]

Figure 6.10: Call blocking rate as a function of network load: MCI topology, 100% guaranteed sessions, and evenly distributed load

Figure 6.12 shows similar results for the switched cluster topology. Again, the heuristic for the shortest-delay path algorithm performs better than the original algorithm, although it does not perform as well as the widest-shortest path algorithm for this scenario. The reason is that in this fairly symmetric topology conserving resources is more important than balancing the load, since the load is already fairly evenly distributed.



(a) b = [4KB, 8KB] & d = [80ms, 120ms]



(b) b = [16KB, 20KB] & d = [200ms, 240ms]

Figure 6.11: Call blocking rate as a function of network load: MCI topology, 100% guaranteed sessions, and unevenly distributed load

6.7.3 Running Time

An important performance issue for routing is the running time of the path selection algorithm. Our performance metric is the average algorithm running time in our simulation:

$$\text{Average running time} = \frac{\text{time used by routing algorithm}}{\text{number of flows}}.$$

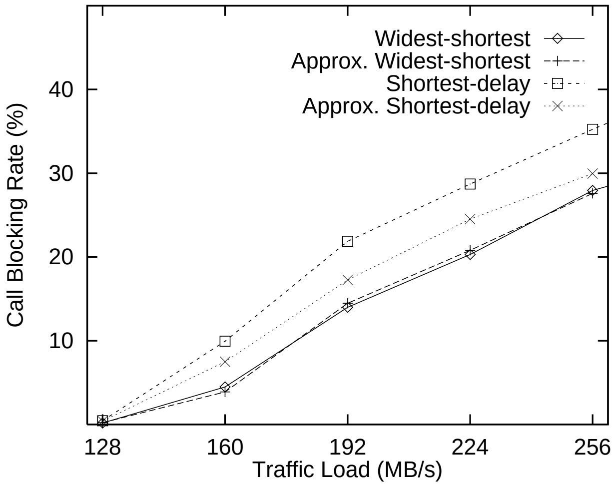


Figure 6.12: Call blocking rate as a function of network load: Switched cluster, 100% video traffic, evenly distributed load, b = [16KB, 20KB], and d = [200ms, 240ms]

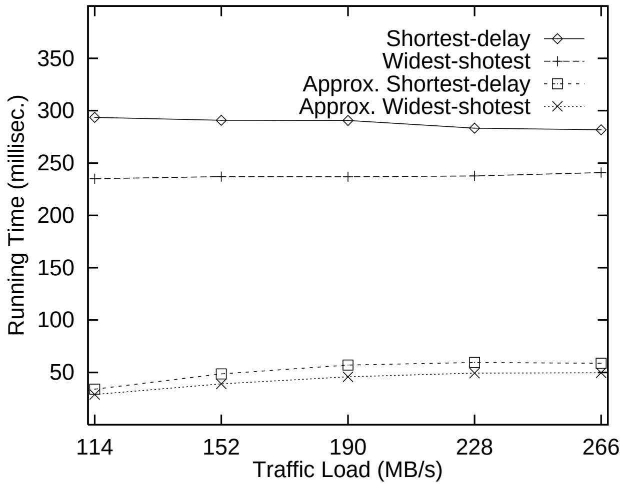


Figure 6.13: Average running time as a function of network load: MCI topology, 100% guaranteed sessions, 90% link-capacity reservation, unevenly distributed load, b = [16KB, 20KB], and d =[200ms, 240ms]

We compare the average running time of the approximation algorithms and the original iterative Bellman-Ford algorithms. The time is measured on a DEC Alpha Station 255. Figure 6.13 shows the time (in milliseconds) needed for the widest-shortest and shortest-delay path algorithms, and their approximations. We see that the approximation algorithms run significantly faster than the original algorithms by up to a factor of 8. Moreover, we see that the widest-shortest path algorithm and its approximation run 18% faster than the shortest-delay path and its approximation, respectively. The significant improvement on the running time of our approximation algorithms comes from eliminating iterations over similar but different values of the link residual bandwidth.

6.8 An NP-Complete Result

In Section 6.2, we assume that the same amount of bandwidth r is reserved on all the the links of a path. A more general result defines delay and delay-jitter bounds when different amounts of bandwidth can be reserved on different links (see, e.g., [22, 39]):

$$D(\mathbf{p}, \sigma, b) = \frac{b}{\min\{r_i \mid 1 \le i \le n\}} + \sum_{i=1}^{n} \frac{L_{\max}}{r_i} + \sum_{i=1}^{n} \frac{L_{\max}}{C_i} + \sum_{i=1}^{n} \mathrm{prop}_i \tag{6.6}$$

where r_i ($r_i \ge \sigma$) is the bandwidth to be reserved on link i and $L_{\max}$ and prop_i are the same as in Section 6.2. The end-to-end delay-jitter is bounded by

$$J(\mathbf{p}, \sigma, b) = \frac{b}{\min\{r_i \mid 1 \le i \le n\}} + \sum_{i=1}^{n} \frac{L_{\max}}{r_i}. \tag{6.7}$$

We show that requiring the same amount of bandwidth to be reserved over all links is essential to the proof of Proposition 8: the path finding problem **(D-J-B)** becomes computationally intractable otherwise, even if rate-proportional scheduling algorithms are used. We first prove two lemmas. A distance function l is said to be *additive*, if $l(P) = \sum_{(i,j)\in P} l(i,j)$.

Lemma 2 *Given a graph G and two non-negative additive distance functions d_1 and d_2 with $d_1 \le d_2$ and two bounds $\mathbf{d}_1$ and $\mathbf{d}_2$ with $\mathbf{d}_1 \le \mathbf{d}_2$, finding a path P from a source s to a destination d, such that $d_1(P) \le \mathbf{d}_1$ and $d_1(P) \le \mathbf{d}_1$ is NP-complete.*

Proof. This problem is a special case of the NP-complete two Additive Metrics Problem [102]. To show the problem is NP-complete, we only need to show that the Additive Metrics Problem can be reduced to this problem. Given two non-negative distance functions $\bar{d}_1$ and $\bar{d}_2$ and two bounds $\bar{\mathbf{d}}_1$ and $\bar{\mathbf{d}}_2$, let C be $1 + \bar{\mathbf{d}}_2 + \max\{\bar{d}_2(i,j) \mid (i,j) \in G\}$ and

$$d_1 = \bar{d}_1/C, \quad d_2 = \bar{d}_2, \quad \mathbf{d}_1 = \bar{\mathbf{d}}_1/C, \quad \text{and} \quad \mathbf{d}_2 = \bar{\mathbf{d}}_2.$$

Clearly, we have $d_1 \le d_2$ and $\mathbf{d}_1 \le \mathbf{d}_2$. A path P satisfies $d_1(P) \le \mathbf{d}_1$ and $d_1(P) \le \mathbf{d}_2$ if and only if $\bar{d}_1(P) \le \bar{\mathbf{d}}_1$ and $\bar{d}_1(P) \le \bar{\mathbf{d}}_2$. □

Lemma 3 *Given a graph G and two non-negative additive distance functions d_1 and d_2, and two bounds $\mathbf{d}_1$ and $\mathbf{d}_2$ with $\mathbf{d}_1 \geq \mathbf{d}_2$. Finding a path P from a source s to a destination d, such that $d_1(P) + d_2(P) \leq \mathbf{d}_1$ and $d_2(P) \leq \mathbf{d}_2$ is NP-complete.*

Proof. To show this problem belongs to NP, we reduce it to the problem in Lemma 2 using the following reduction

$$\bar{d}_1 = d_1 + d_2, \quad \bar{d}_2 = d_2, \quad \bar{\mathbf{d}}_1 = \mathbf{d}_1, \quad \text{and} \quad \bar{\mathbf{d}}_2 = \mathbf{d}_2.$$

To show that the problem is NP-complete, we reduce the problem in Lemma 2 to this problem by using the reduction

$$\bar{d}_1 = d_1 - d_2, \quad \bar{d}_2 = d_2, \quad \bar{\mathbf{d}}_1 = \mathbf{d}_1, \quad \text{and} \quad \bar{\mathbf{d}}_2 = \mathbf{d}_2.$$

It is clear that, for any path P,

$$\bar{d}_1(P) + \bar{d}_2 \leq \bar{\mathbf{d}}_1 \;\text{ and }\; \bar{d}_2(P) \leq \bar{\mathbf{d}}_2 \;\text{ if and only if }\; d_1(P) \leq \mathbf{d}_1 \;\text{ and }\; d_2(P) \leq \mathbf{d}_2$$

□

Theorem 2 *The QoS routing problem of finding a path with delay and delay-jitter bounds* **d** *and* **j** *is NP-complete if the bandwidth to be reserved on different links can be different.*

Proof. To show that the problem is NP-hard, we can reduce the problem in Lemma 3 to this problem by using the following reduction

$$b = 0, \quad r_i = \frac{L_{\max}}{d_2(i)}, \quad C_i = \frac{L_{\max}}{d_1(i) + d_2(i)}, \quad \text{prop}_i = 0 \quad \mathbf{d} = \mathbf{d}_1, \quad \text{and} \quad \mathbf{j} = \mathbf{d}_2.$$

To show that the problem is NP-complete, we iterate any algorithm for the problem in Lemma 3 over all link residual bandwidth values, with the first term in Equations 6.6 and 6.7 replaced by b/r, where r is the current residual bandwidth. □

6.9 Related Work

Many studies in the literature have addressed different aspects of QoS routing. A good introduction to QoS routing and existing routing techniques can be found in [63, 95]. In [21], a QoS based routing framework is defined and many challenging questions are raised.

Several studies have investigated algorithmic problems associated with QoS routing. In [33, 102], it is shown that the QoS routing problem of finding a path satisfying both delay and delay-jitter constraints is NP-complete. Jaffe and Salama studied [49, 90] heuristics to tackle the

NP-problem of routing with two additive constraints. Przygienda suggested the use of cost vectors to solve the NP-complete problem by associating a priority with each constraint [81]. Rampal [87] evaluated the performance of several path selection algorithms, assuming three different scheduling algorithms. The IBF algorithm that is used in this paper was first described in [68]. Zhao and Tripathi in [108] proposed a similar algorithm, but issues related to selecting efficient paths and resource utilization efficiency are not addressed. Several people have presented algorithms that address a subset of the problems associated with routing guaranteed traffic. In [89], routing algorithms for achieving the shortest delay when moving a burst of bytes are proposed. In [79], QoS routing algorithms are studied for finding the shortest-delay path when the network employs the weighted fair queueing service discipline. In [43], QoS routing algorithms for a network with inaccurate link-state information are explored. Guerin, Orda, and Williams outlined in [44] how OSPF can be extended to QoS routing, using the widest-shortest feasible paths, but no performance evaluation is provided.

In contrast to the previous studies, we present not only a complete set of QoS routing algorithms for traffic with delay, delay jitter, and buffer space constraints with a variety of optimality criteria, but also approximation algorithms that run significantly faster than the original algorithms without sacrificing resource utilization efficiency. We also present a detailed performance evaluation of four different algorithms, considering both call blocking rate and impact on best-effort traffic as performance metrics.

6.10 Summary

In this chapter, we present a study of routing for traffic requiring delay, delay jitter, and packet loss guarantees. Our simulation study evaluates the proposed routing algorithms for guaranteed sessions while also considering their impact on best-effort sessions.

While the general routing problem of finding a path with multiple constraints is NP-complete, we show that, for a broad class of rate-proportional packet service disciplines employed by the network, QoS routing can be turned into a simple polynomial iterative Bellman-Ford algorithm if the relationship among QoS constraints defined by the service disciplines is incorporate into the algorithm design. Our algorithms makes no assumption about the link queueing delay and do not need to know the bandwidth to reserve in advance, but derive it during routing.

Conserving per-flow resources and balancing the network load are two important ways achieving resource utilization efficiency. To achieve low blocking rates, we identified four optimization criteria that place a different emphasis on minimizing resource utilization and balancing the load in the network. Based on the IBF algorithm, we developed path selection algorithms that select paths with these different criteria. The performance of these routing algorithms is evaluated through an extensive simulation study, using realistic topologies, traffic models, and periodic routing information distribution. Unlike telecommunication networks, multiple classes of service will be supported in data networks. Our performance evaluation considers not only the blocking rate for guaranteed traffic but also the influence on the throughput of best-effort traffic. Our results show that the performance gap for these different algorithms is large and both conserving resources and balancing the network load are necessary to achieve high network throughput. The widest-shortest path algorithm performs well for heavy load because it conserves resources, but the shortest-delay path has a performance edge for light load because it balances the load better. Surprisingly, the minimum-bandwidth path algorithm has the worst performance because it is not very sensitive to load, and can easily saturate some links while other parts of the network are underutilized.

Even though the IBF algorithms are polynomial, their computational complexity is significantly higher than that of the Bellman-Ford algorithm. We develop approximation algorithms that runs almost ten times faster than the exact iterative Bellman-Ford algorithm for an MCI backbone topology. The simulation results show that there is no loss in routing accuracy and network throughput. Overall, the approximation algorithm for the shortest-delay path performs consistently well.

Chapter 7

An Integrated Routing Framework

In the previous chapters, we defined and evaluated routing algorithms for individual traffic classes, and while we did evaluate how routing decisions for reserved sessions can impact the performance of best effort traffic, routing decisions for traffic in one class did not consider the traffic conditions in, or even the presence of, other traffic classes. The assumption of a homogeneous traffic model simplified both the analysis as well as the interpretation of the results. However, in an intergrated services network that provides multiple classes of service, flows of different classes coexist in the network, and ignoring the other traffic classes when routing a session in a particular class could result in poor performance. In this chapter, we present a routing framework that integrates the routing algorithms for individual traffic classes into a multi-class routing algorithm. The goal is to improve inter-class resource sharing and to achieve high resource utilization efficiency. To simplify our discussion, we assume that all guaranteed sessions require only bandwidth guarantees. The proposed framework can also be applied to networks that support sessions with delay guarantees.

The rest of the chapter is organized as follows. We begin with a brief discussion in Section 7.1 on why routing for an integrated services network is more complicated than just putting together the routing algorithms for the different traffic classes. We then present our integrated routing framework in Section 7.2 and evaluate its performance in Section 7.3. Section 7.4 discusses the applicability and possible extension of the proposed framework. Related work is discussed in Section 7.5 and we conclude in Section 7.6.

7.1 The problem

This section motivates the study of inter-class resource sharing in routing multiple classes of traffic.

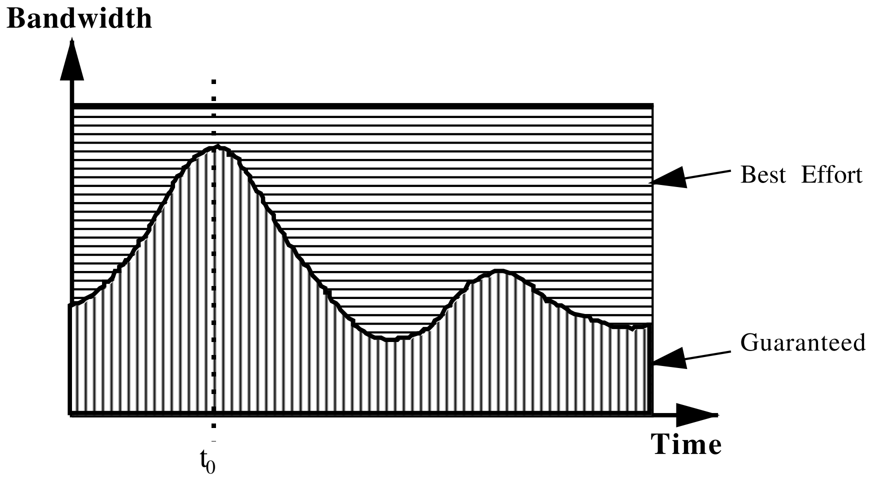


Figure 7.1: Link resources are shared across guaranteed traffic and best effort traffic

7.1.1 Multi-Class Routing: Challenges

When both guaranteed and best-effort flows are present in the network, the network resources are shared by the two service classes. As a result of the different resource sharing models (reservation for guaranteed sessions and fair sharing for best-effort sessions), guaranteed sessions have priority over best-effort sessions: once resources are reserved, they are no longer available to best-effort sessions, unless they are unused by guaranteed sessions. The packet scheduler in a router or switch ensures that the guaranteed traffic will get at least the amount of bandwidth that it reserved. The portion of the link capacity available to best-effort traffic is allocated according to max-min fair sharing policy. Given a fixed traffic load of guaranteed sessions, the resources available to best effort traffic will be influenced by the routing of guaranteed sessions.

If the routing algorithm for guaranteed sessions ignores the load of low priority traffic, as we did in earlier chapters, it in fact optimizes its throughput while ignoring the performance of best-effort sessions. Considering that best-effort traffic is likely to continue to be the dominant traffic class in the network for quite a while, optimizing the throughput for guaranteed sessions while ignoring the performance of best effort traffic is the wrong tradeoff.

Generally speaking, since the network is typically designed to match a long-term average of the network load, the short-term traffic load of the network can often be quite different. As a result, how much bandwidth is left over for best effort sessions can be highly variable (Figure 7.1). Let us look at an example scenario. Links close to a powerful data server or Web server can get heavily loaded with best-effort traffic, even when the unreserved link bandwidth on these links is high. Simply giving high priority to guaranteed sessions while ignoring low

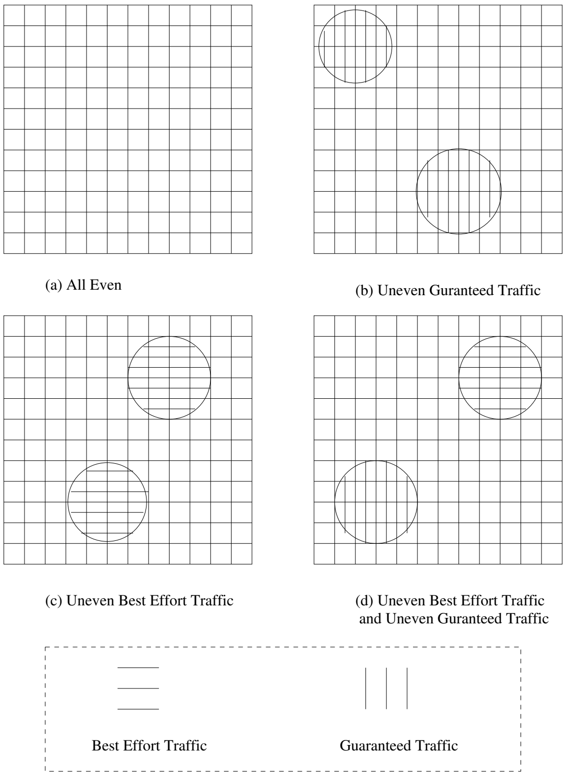


Figure 7.2: Four possible scenarios of load distribution

priority traffic will direct guaranteed sessions to these congested links, which will worsen the congestion conditions on these links.

Because both guaranteed traffic and best-effort traffic can be either evenly or unevenly distributed, there are four combinations of load distribution, as shown in Figure 7.2. The interesting scenarios are when the best-effort traffic is unevenly distributed. In such scenarios, if guaranteed sessions can avoid the best-effort hot spots, the throughput for best effort traffic can potentially be improved significantly.

7.1.2 Alternative Approaches

There are several possible ways to integrate routing algorithms employed for individual service classes.

The most naive approach is just simply put algorithms for different traffic classes together. As we discussed above, this approach can cause significant congestion or even starvation of best-effort traffic. In Section 7.2.4 we will compare this approach with the integrated approach we are proposing.

A second approach to enforce a static link sharing policy as has been studied in [29] for class based queueing. With this approach, the capacity of a link is statically divided between guaranteed sessions and best-effort sessions. This approach also raises a number of problems. First, it is difficult to determine how much of the link capacity should be used for each traffic class without sacrificing resource utilization efficiency, because the ratio of guaranteed sessions over best-effort sessions changes over time. Allocating too much bandwidth for best effort sessions can introduce high blocking rate for guaranteed sessions, while allocating too little bandwidth for best effort sessions can cause serious congestion. Second, the traffic loads of the different traffic classes are not always evenly distributed over the network. For example, guaranteed sessions can be concentrated in one part of the network, while best-effort sessions can be concentrated in another part of the network. Static sharing policies cannot adapt to such an imbalance, so they can result in both poor resource utilization and congestion.

A semi-dynamic link sharing policy may be employed to overcome the problems of static link sharing. Under this sharing policy, a histogram of the measured link utilization of different traffic classes is used to determine what fraction of the link capacity should be assigned to the different traffic classes. However, this semi-dynamic strategy may not work well if there are sudden changes in the utilization. Often these changes are caused by external phenomena which are difficult or impossible to predict and whose effects are acute. Because the semi-dynamic algorithms are reactive rather than pro-active, latencies associated with their adaptive operation may make their reaction times too long.

Another approach is to use the actual measured link utilization, including that are used by guaranteed traffic as well as best-effort traffic, as the link state for the both traffic classes. This will route guaranteed sessions along the links with low utilization. However, link utilization is not always a good indicator of how much bandwidth available to reserve. For example, very bursty best-effort sessions can consume all link bandwidth, which leads to high untilization of the link, eventhough the link in question may appear to have available reservable bandwidth. The routing algorithm for guaranteed traffic may reject a request eventhough bandwidth are available to reserve.

Another extream is to simply ignore the load of guaranteed traffic and use the load of best-effort traffic alone to select paths for guaranteed sessions. This can lead to a high blocking rate in some scenarios, since it does not take the load of guaranteed traffic into consideration.

In our multi-class routing algorithm, link resources are dynamically partitioned between guaranteed traffic and best-effort traffic. By adjusting link costs, we discourage the routing algorithm for guaranteed traffic from selecting links congested with best effort traffic.

7.2 An Inter-class Sharing Architecture

We develop a multi-class routing algorithm that supports dynamic sharing of the link capacity among traffic classes. Since best-effort traffic will continue to be the dominate traffic class in integrated services networks, we focus on improving the resource utilization efficiency when the guaranteed sessions do not represent the dominating traffic class. Our design has two goals. First, it seeks to admit the same number of guaranteed sessions as would be admitted by a dedicated network. Second, it seeks to improve the performance of the best-effort traffic by optimizing the placement of the guaranteed sessions.

7.2.1 Relative Link Congestion

To achieve the goal of efficient inter-class resource sharing, the first step is to identify link congestion. Our algorithm uses a measure of max-min fair share rate as a barometer of network congestion. In general, it observes a threshold to determine whether a given link is congested. This threshold is dynamically calculated using network state information and permits a finer measure of link congestion than would be possible with a static threshold. With link-state routing, the max-min fair share rate information for all links of the network are known. From this rate information, it is easy to calculate the minimum and the mean max-min rates, which are represented by min and mean. The threshold that is used to determine whether a link is congested is a function of min and mean as described below.

Knowing the minimum and mean max-min fair share rates, we need to determine in what rate interval a link is congested and in what interval it is not. Contrary to our intuition, the mean max-min fair rate mean does not serve as a threshold to distinguish if a link is congested. Congested session should have traffic loads well above average, so we want to select a threshold that is lower than, but tracks, the mean. We thus divide link rates into the following three intervals: $[\mathsf{min}, \mathsf{low\_mark})$, $[\mathsf{low\_mark}, \mathsf{high\_mark}]$, and $(\mathsf{high\_mark}, \infty)$, where

$$\mathsf{low\_mark} = \mathsf{min} + \frac{\mathsf{mean} - \mathsf{min}}{3} \tag{7.1}$$

$$\mathsf{high\_mark} = \mathsf{mean} - \frac{\mathsf{mean} - \mathsf{min}}{3} \tag{7.2}$$

This divides the interval $[\mathsf{min}, \mathsf{mean}]$ into three equal subintervals (see Figure 7.3). When the max-min fair share rate of a given link is in the lower subinterval $[\mathsf{min}, \mathsf{low\_mark}]$, the link is considered congested and, therefore, its future use should be discouraged. If the max-min fair rate of a link is greater than $\mathsf{high\_mark}$, the link is not congested and its use should be encouraged.

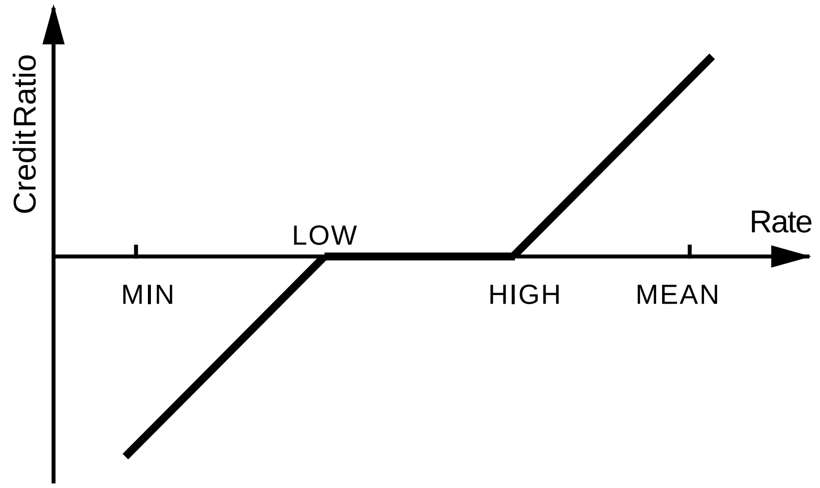


Figure 7.3: An inter-class load balance algorithm

7.2.2 Virtual Residual Bandwidth

In order to encourage or to discourage the use of a link, we assign a certain amount of "credits" to the link. We define the concept of "virtual residual bandwidth", which is the residual bandwidth adjusted according to the "credit bandwidth" defined below.

$$\text{virtual residual bandwidth} = \text{residual bandwidth} + \text{credit bandwidth} \tag{7.3}$$

The amount of credit to be assigned to a link depends on its link congestion condition. We use the following formula:

$$\text{credit ratio} = \begin{cases} \frac{r}{\mathsf{low\_mark}} - 1 & \text{if } r < \mathsf{low\_mark} \\ 0 & \text{if } \mathsf{low\_mark} \le r \le \mathsf{high\_mark} \\ \frac{r}{\mathsf{low\_mark}} - \frac{\mathsf{high\_mark}}{\mathsf{low\_mark}} & \text{if } r > \mathsf{high\_mark} \end{cases}$$

$$\text{credit bandwidth} = \text{credit ratio} * \mathsf{MAX\_CREDIT} * \text{residual bandwidth}$$

where MAX_CREDIT is the maximal ratio of adjustment to be made when the max-min fair share rate of the link is approaching 0. From the definition, the credit ratio is positive when the max-min fair rate r of the link is in the interval (high_mark, ∞) and is negative when r is in the interval [min, low_mark).

Other definitions for the credit ratio are of course possible. Our experiments suggest the credit ratio has to meet the following guidelines to be effective:

- For a link heavily loaded with best-effort traffic, the difference between real residual bandwidth and virtual residual bandwidth should be large enough so that this congested link can be avoided.

- The change of the value of virtual link residual bandwidth should be smooth. For this reason, we create an interval of max-min fair share value that yields credit 0.

7.2.3 Dynamic Link Sharing

Our inter-class load sharing policy works as follows. A negative credit is added to the residual bandwidth of a link if the link is heavily loaded with best-effort traffic. Similarly, a positive credit is added to lightly loaded links. In a multi-class routing environment the use of "virtual" residual bandwidth instead of "real" residual bandwidth provides a more realistic measure of the congestion on a given link. Moreover, this measure of congestion tracks changes in the network load and is therefore no need to statically split link capacity between guaranteed sessions and best-effort sessions. Guaranteed sessions avoid using links that are heavily loaded with best-effort traffic.

While dynamic link sharing works well for most scenarios, as we show below, it can potentially create a number of problems. First, even though we focus on the case that best effort sessions dominate the traffic load, it is still important to consider what happens when the guaranteed traffic load is both high and uneven. Under these conditions, it is possible that the guaranteed sessions reserve all the capacity of certain links, which may result in starvation of the best effort traffic. To avoid this problem, we assume that a certain fraction of the capacity of each link (e.g., 5% or 10%) is reserved for best-effort traffic, and we evaluate the impact of this restriction later in the chapter.

Another potential problem is that the dynamic link sharing algorithm may force guaranteed sessions to use longer physical paths, which increases resource consumption and can potentially result in higher blocking rates. However, as was shown in Chapter 5, for light loads of guaranteed traffic, the blocking rates for different routing algorithms are all zero or close to zero, so this should not be an issue for light loads. Moreover, our simulation results in Section 7.2.4 show little increase in the blocking rate, even when the guaranteed traffic load is quiet heavy.

7.2.4 A Multi-Class Routing Algorithm

When routing traffic with bandwidth guarantees, it is shown in Chapter 5 that the shortest-distance path based on the link residual bandwidth performs consistently well for both evenly unevenly distributed loads. The distance of a given path P is determined by the equation

$$\text{dist}(P) = \sum_{i \in P} \frac{1}{R_i} \tag{7.4}$$

where R_i is the residual bandwidth of link i. For best effort traffic, it is shown in [70] that a polynomial based distance function

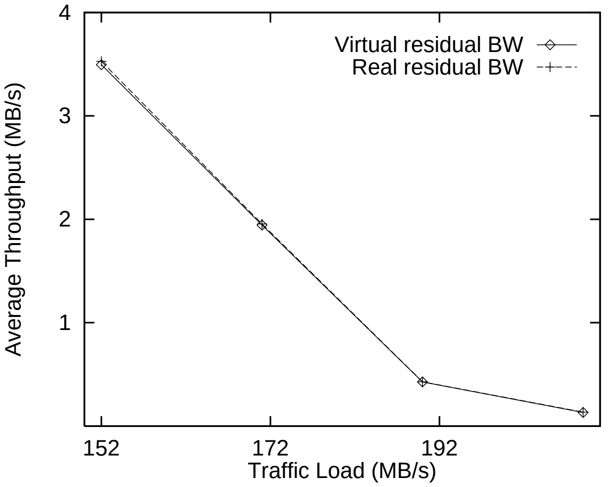
$$\text{Shortest--dist}(P, n) = \sum_{i \in P} \frac{1}{r_i^n} \tag{7.5}$$

where r_i is the max-min fair share rate for a new connection on link i, performs consistently better than either the minimum-hop path or the widest path.

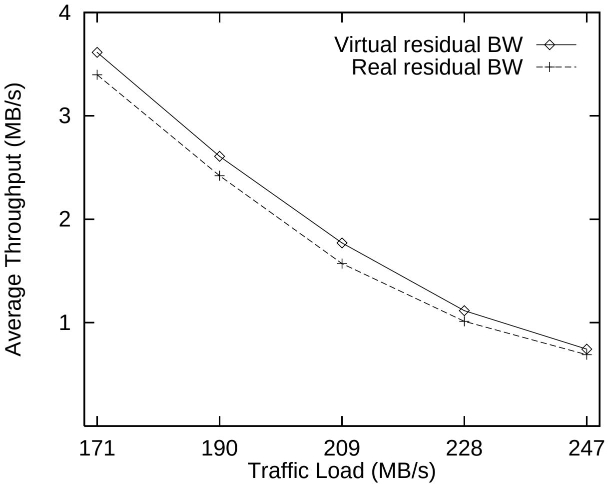
Given the two algorithms for the individual service classes, our multi-class routing algorithm is simple. For a best-effort session, it uses the shortest-path routing algorithm with the distance function of Equation (7.5) as the cost. For a guaranteed session, it uses the that shortest-path algorithm with the distance function of Equation (7.4) as the cost, but using the virtual residual bandwidth R_i as defined in Equation (7.3) as the rate:

```
Multi_Class_Routing(alg_best_effort, alg_bw_guarantee, F, G)
void *(alg_best_effort)(), *(alg_bw_guarantee)();
flow F;
topology G;
{
    topology G_virtual;

    if (F is a best-effort traffic flow)
        alg_best_effort(F, G);
    else (F is a flow of guaranteed traffic) {
        G_virtual = credit_adjustment(G);
        alg_bw_guarantee(F, G_virtual);
    }
    return;
}
```

Hence, the proposed multi-class requires no changes to and is independent of, routing algorithms employed for individual service classes. The only change needed is the use of the virtual residual bandwidth instead of the real residual bandwidth when routing guaranteed sessions. Note that the impact on throughput of the multi-class routing algorithm can of course be different when different algorithms are used for individual service classes.



(a) Evenly distributed load



(b) Unevenly distributed load

Figure 7.4: Average throughput as a function of network load: MCI topology and 50% high bandwidth traffic, 90% of maximal reservation ratio. The Shortest-dist(P,1) path is used for best-effort traffic.

7.3 Performance Evaluation

Our evaluation of the proposed multi-class routing algorithm characterizes the effects of topology, traffic load, degree of unevenly distributed load, maximum reservation ratio over a link, routing algorithms used for best-effort sessions, and the ratio of guaranteed to best-effort traffic. We compare several algorithms. Most of the evaluation focuses on comparing the performance of the dynamic inter-class resource sharing algorithm algorithm described in Section 7.2.4 with an approach in which the traffic of each class is routed independently. We will sometimes refer to the latter approach as isolated routing. We also compare our multi-class routing algorithm with an algorithm that uses static link partitioning.

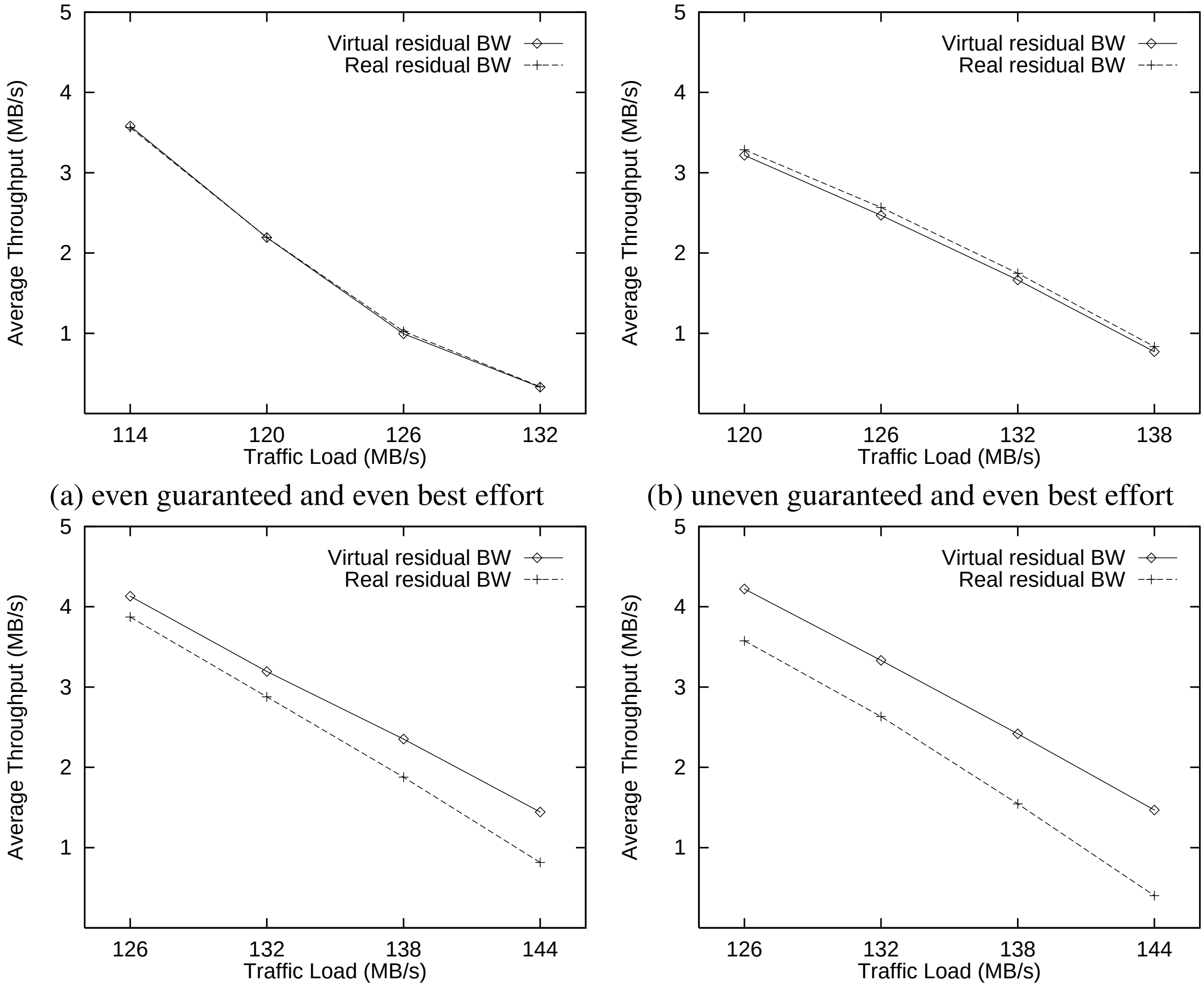
Most of our simulation results are for the small cluster topology, since it is easier to understand inter-class sharing behavior with a small topology than with a large one. It also has more manageable simulation times. As we described above, we expect that large topologies, such as the MCI topology, will have regions with uneven traffic loads; the cluster topology should be viewed as an example of such a region. Obviously, in a large topology, the average performance results will mix the performance of regions with even and uneven load, so the performance effects of inter-class routing may be less pronounced, depending on the mix of traffic. Given the many parameters, it is impractical to try to get a single number characterizing performance. Instead, the goal is to show when and under what circumstances dynamic inter-class resource sharing is desirable and to evaluate its ability to achieve its goal without reducing performance in network regions that have an even load.

To simplify our evaluation, we assume that all guaranteed sessions require bandwidth guarantees, and the routing algorithm for guaranteed bandwidth sessions is the shortest distance path algorithm that uses the link cost of Equation (7.4). For best-effort traffic, we consider both the Shortest-dist$(P, 1)$ path and the shortest-dist$(P, 0.5)$ path, as studied in Chapter 4, to characterize the performance improvement of the proposed routing framework. Our evaluation uses the MCI topology and the G3 topology (see Chapter 3) and a default routing information update interval of 30 seconds. We note that the discussion on the performance of inter-class sharing can also be applied to the case when the guaranteed sessions require delay guarantees.

7.3.1 Impact of Traffic Load

Figure 7.4 shows the average throughput (MB/s) as a function of the network load for the MCI topology. The traffic is split evenly between traffic requiring bandwidth guarantees and best effort traffic, with each accounting for 50% of the data transferred. The routing algorithm used for high-bandwidth best-effort traffic is the Shortest-dist$(P, 1)$ path. For all traffic loads considered, no guaranteed sessions are blocked.

When the traffic load is evenly distributed, we see virtually identical performance for the best-effort sessions, regardless of whether the actual residual bandwidth or virtual residual bandwidth is used. This is not unexpected. The function of the integrated routing framework is to balance the load of the guaranteed traffic so as to avoid using links that are congested with best-effort traffic. For an even load, the ratio of the traffic from different classes on each link should be approximately evenly distributed. In that case, the dynamic inter-class sharing



(a) even guaranteed and even best effort

(b) uneven guaranteed and even best effort

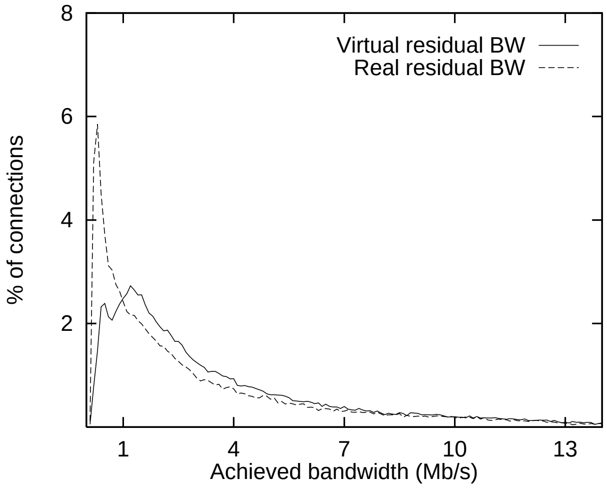
(c) even guaranteed and uneven best effort

(d) uneven guaranteed and uneven best effort

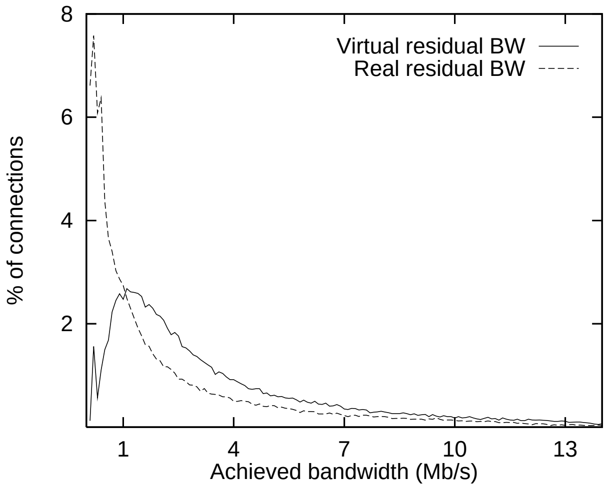
Figure 7.5: Average throughput as a function of network load: cluster topology, 50% guaranteed traffic and 50% best-effort traffic, and 90% of maximal reservation ratio. The Shortest-dist$(P, 1)$ path is used for best-effort traffic.

should, by design, have little or no impact on the overall performance, which is what we observe.

For an unevenly distributed load, where 80% of the best-effort traffic is distributed on the upper half of the network and 70% of the guaranteed traffic concentrated on the lower half of the network, we see an increase of up to 13% in the average throughput for best-effort sessions. Without dynamic link sharing, guaranteed sessions can be directed to links that are already



(a) even guaranteed and uneven best effort



(b) uneven guaranteed and uneven best effort

Figure 7.6: Variation of achieved throughput: cluster topology, 50% guaranteed traffic and 50% best-effort traffic (144 MB/s), and 90% of maximal reservation ratio. The Shortest-dist$(P, 1)$ path is used for best-effort traffic.

congested with best-effort traffic. With dynamic link sharing, links that are congested with best-effort sessions are avoided by the routing algorithm for guaranteed sessions. For an uneven load, this improvement is significant since the performance index is average throughput. We will discuss the distribution of the per-flow bandwidth later.

Figure 7.5 shows the average throughput for the six-node cluster topology, where 50%

of traffic is best-effort traffic and 50% is guaranteed traffic. The routing algorithm used for best-effort traffic is the Shortest-dist$(P, 1)$ path. For the case of unevenly distributed best-effort traffic, 72% of best effort sessions are concentrated on the links between node pairs (1, 4) and (3, 5). For the case of unevenly distributed guaranteed traffic, 72% of traffic are distributed between the node pairs other than (1, 4) and (3, 4), (4, 5), and (3, 5).

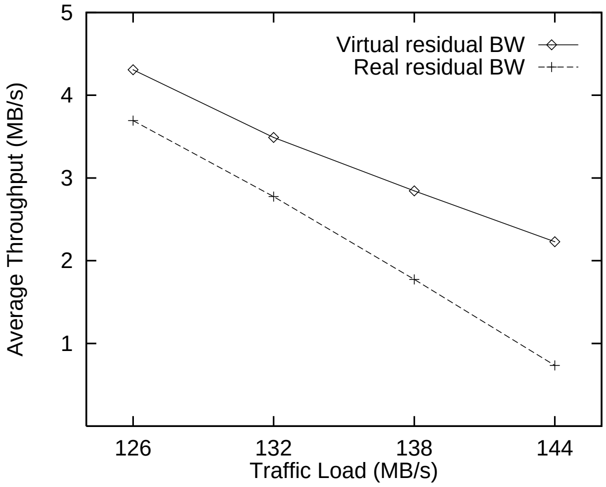
The four graphs in Figure 7.5 correspond to different combinations of evenly and unevenly distributed loads. In all four cases, no flows are blocked. For evenly distributed load (Figure 7.5 (a)), we see similar performance regardless of whether real or virtual residual bandwidth is used. Thus, our inter-class sharing mechanism does not hurt the performance of either traffic class when the load is evenly distributed, which was one of our design goals. When the load for guaranteed traffic is unevenly distributed, we see a very small performance degradation for best-effort traffic. The reason for this degradation is that, as a result of the inter-class sharing mechanism, guaranteed sessions sometimes use a slightly longer and more resource-intensive path, which takes resource away from best effort sessions.

When the best-effort traffic is unevely distributed (Figure 7.5 (c) and (d)), we observe performance gains for best effort traffic. In some scenarios, *e.g.*, when the load is as heavy as 144 MB/s, the performance improvement for best effort traffic is as high as a factor of 3, because guaranteed sessions are routed around links (1, 4), (3, 4), and (4, 5) that are heavily loaded with best-effort traffic. Note that we are not suggesting that this is a typical performance improvement. Rather, our results indicate that the dynamic inter-class resource sharing algorithm produces performance improvements in the most congested regions of the network, when the traffic load is unevenly distributed.

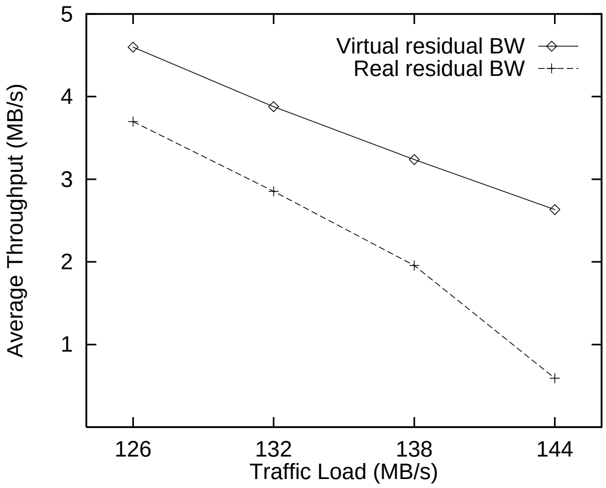
Figure 7.6 shows the throughput distribution of best-effort sessions for the load of 144 MB/s in Figure 7.5(c) and (d), i.e. for uneven best effort traffic. We see that the sessions that benefit the most from inter-class routing are the sessions with the worst performance, so inter-class routing results in a more even load distribution. Dynamic inter-class routing shifts many connections that achieve an average throughput between 0 and 1 Mb/s to a higher bandwidth region. This is done by directing guaranteed sessions to less congested links.

7.3.2 Impact of Best-effort Traffic Volume

In the previous experiments, the traffic is equally distributed between best-effort and guaranteed sessions. For the results in Figure 7.7, 75% of the traffic is best-effort traffic. With the increased volume of best-effort traffic, links where best-effort traffic is concentrated become even more congested. As a result, it becomes even more important to route guaranteed sessions to other links whenever possible. Thus, inter-class dynamic resource sharing results in a higher performance improvement when compared with the case of a 50/50 workload.



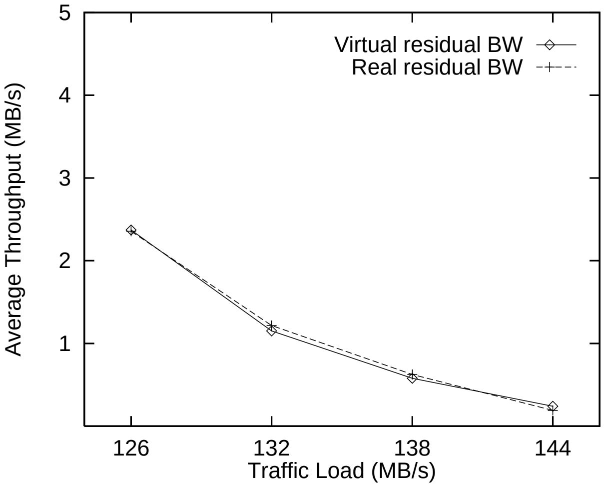
(a) even guaranteed and uneven best effort



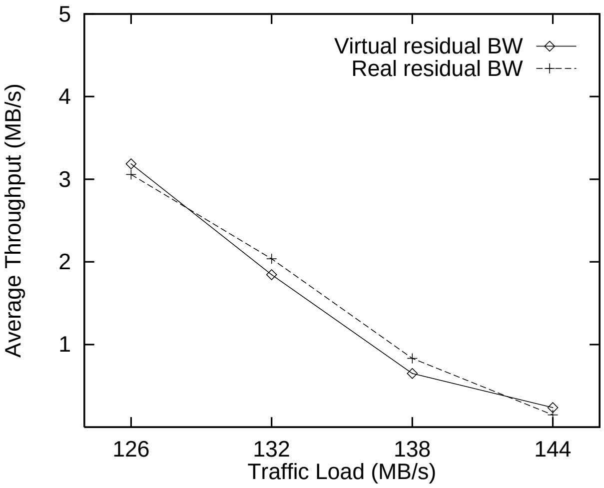
(b) uneven guaranteed and uneven best effort

Figure 7.7: Average throughput as a function of network load: G topology, 25% guaranteed sessions and 75% high-bandwidth sessions, and 90% of maximal reservation ratio. The Shortest-dist$(P, 1)$ path is used for best-effort traffic.

We emphasize that the proposed scheme is intended for use in an environment where the guaranteed traffic load is low compared to the overall network capacity. In such case, the bandwidth blocking rate is not a concern. When the volume of guaranteed traffic is higher, using virtual residual bandwidth can potentially increase the bandwidth blocking rate. Figure 7.8 shows the performance when 75% of the traffic is guaranteed traffic. We see that the



(a) even guaranteed and uneven best effort



(b) uneven guaranteed and uneven best effort

Figure 7.8: Average throughput as a function of network load: G topology, 75% guaranteed sessions and 35% high-bandwidth sessions, and 90% maximal reservation ratio. The Shortest-dist$(P, 1)$ path is used for best-effort traffic.

performance experienced by best effort traffic using inter-class routing and isolated routing is very similar. Depending on the traffic load, either algorithm may have a small performance edge over the other, but on average the performance for best effort traffic is about even. Table 7.1 lists the bandwidth blocking rates for guaranteed sessions. We observe a slightly higher blocking rate for guaranteed sessions when the dynamic inter-class resource sharing algorithm is used.

Traffic load (MB/s)	126	132	138	144
Virtual residual bandwidth	0.408%	0.789%	1.306%	2.196%
Real residual bandwidth	0.381%	0.725%	1.160%	1.85%

(a) even guaranteed and uneven best effort

Traffic load (MB/s)	126	132	138	144
Virtual residual bandwidth	0.031%	0.179%	1.151%	2.174%
Real residual bandwidth	0.008%	0.071%	0.432%	1.088%

(b) uneven guaranteed and uneven best effort

Table 7.1: Bandwidth blocking rate: cluster topology, 75% guaranteed sessions, 25% best-effort sessions, and 90% maximal reservation rate. The Shortest-dist$(P, 1)$ path is used for best-effort traffic.

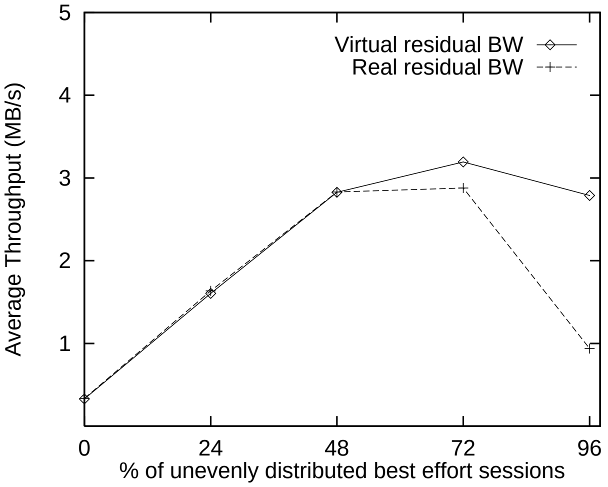
The reason is that, when the network load is heavy, the inter-class resource sharing algorithm can cause flows to use longer, resource-intensive paths, which can increase the blocking rate for guaranteed sessions.

7.3.3 Impact of the degree of Imbalance

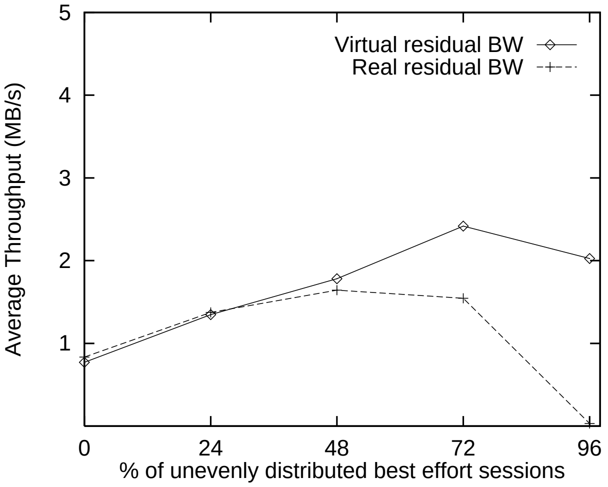
In Figure 7.9, we show the average bandwidth of best-effort traffic as a function of the fraction of best effort sessions that are concentrated between nodes 1-4 and 3-5 of the cluster topology, i.e. higher values mean that the traffic is more uneven. The results are for a distribution of 50% traffic in the guaranteed class and 50% in the best effort class. The total traffic load in both figures is fixed: 132 MB/s for the top figure and 138 MB/s for the bottom figure. We see that the performance gain from using dynamic inter-class resource sharing increases with the volume of unevenly distributed traffic.

7.3.4 Impact of the Maximum Reservation Ratio

As explained in Section 7.2.3, to avoid starvation of best effort sessions, guaranteed sessions can only reserve a certain fraction of the capacity of each link, which we define as the *maximum reservation ratio*. In the previous experiments, we used a maximum reservation ratio of 90%. In Figure 7.10, we show the average throughput as a function of the maximum reservation ratio for a given traffic load. For different maximum reservation ratios, the average bandwidth stays relatively constant when isolated routing is used for each traffic class. When using the multi-class routing algorithm, we see an increase in performance of as much of 50% to 60% as the



(a) even guaranteed and uneven best effort, 132 MB/s



(b) uneven guaranteed and uneven best effort, 138 MB/s

Figure 7.9: Average throughput: cluster topology, 50% guaranteed sessions and 50% high-bandwidth sessions, and 90% maximal reservation rate. The Shortest-dist$(P, 1)$ is used for best-effort traffic.

maximum reservable ratio is increased from 60% to 100%. Increasing the maximum reservable bandwidth ratio basically gives the routing algorithms more freedom. The multi-class routing algorithm can use this additional flexibility to move guaranteed traffic session away from links that have high loads of best effort traffic. In fact, the use a multi-class routing algorithm seems to eliminate the need for a static maximum reservable ratio. The isolated routing algorithm cannot take advantage of this additional freedom.

Table 7.2 shows the corresponding bandwidth blocking rate for guaranteed traffic. High blocking rates can be observed when only a small fraction of the link capacity is reservable by guaranteed traffic. This is not surprise: low maximum reservable ratios limit the freedom of routing algorithms and result in poor performance for both traffic classes. The results in Table 7.2 clearly show that higher reservation ratios translate into higher network throughput. Since these results are for a 50/50 distribution of the traffic between the guaranteed and best effort classes, the low maximum reservable ratios can be viewed as a simple static partioning of network resources between the two traffic classes. In such a case, we observe the poor performance for both best effort and guaranteed sessions. Figure 7.10 combined with Table 7.2 show that dynamic inter-class resource sharing combined with a high maximum reservable ratio results in the best performance.

Maximal reservation ratio (%)	60	70	80	90	100
Virtual residual bandwidth	3.402%	0.400%	0.000%	0.000%	0.000%
Real residual bandwidth	2.459%	0.278%	0.000%	0.000%	0.000%

(a) even guaranteed and uneven best effort

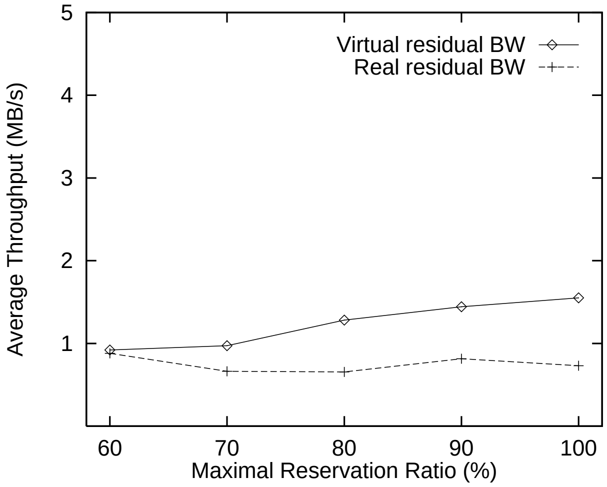
Maximal reservation ratio (%)	60	70	80	90	100
Virtual residual bandwidth	5.055%	0.424%	0.000%	0.000%	0.000%
Real residual bandwidth	3.379%	0.309%	0.000%	0.000%	0.000%

(b) uneven guaranteed and uneven best effort

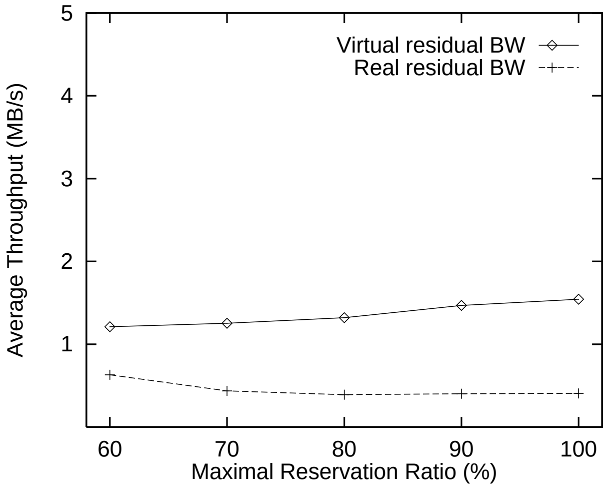
Table 7.2: Bandwidth blocking rate: cluster topology, 144 MB/s, 50% guaranteed sessions and 50% high-bandwidth sessions.

7.3.5 Impact of Routing Algorithms for Best Effort Traffic

In all the previous evaluations, we used the Shortest-dist$(P, 1)$ path routing algorithm for best effort traffic. Figure 7.11 shows the performance for the same scenarios as Figure 7.5, but the routing algorithm used for best-effort traffic is the Shortest-dist$(P, 0.5)$ path algorithm. Note that the Shortest-dist$(P, 0.5)$ path algorithm is less sensitive to bandwidth availability, and more sensitive to hop count than the Shortest-dist$(P, 1)$ path algorithm. A first observation is that the Shortest-dist$(P, 0.5)$ path results in lower performance than the Shortest-dist$(P, 1)$ path for link costs based on both virtual and real residual bandwidth. The reason is that the Shortest-dist$(P, 0.5)$ path algorithm is more likely to use the minimum-hop path than the



(a) even guaranteed and uneven best effort

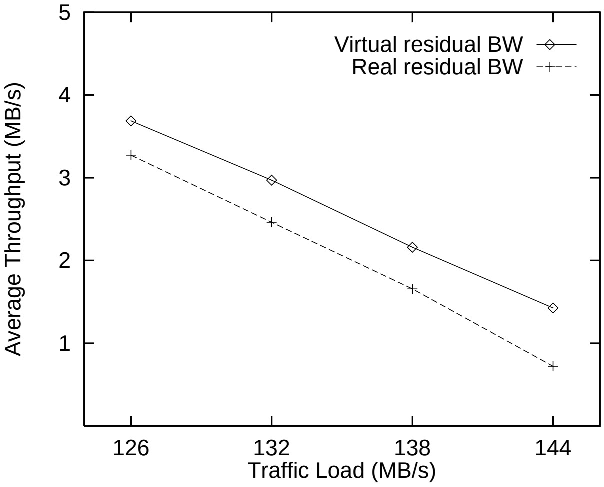


(b) uneven guaranteed and uneven best effort

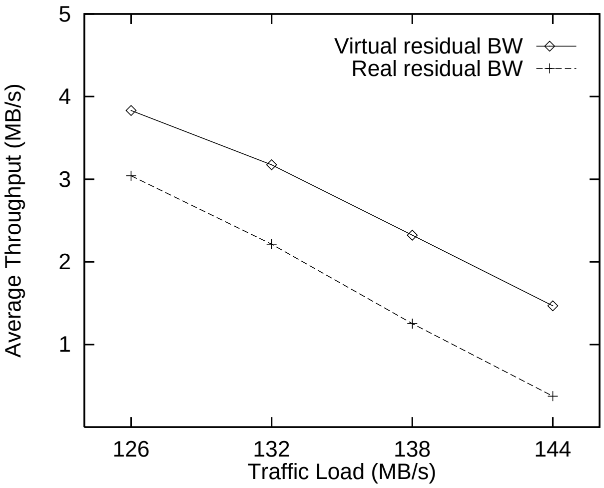
Figure 7.10: Average throughput as a function of the maximal reservation rate: cluster topology, 144 MB/s, 50% guaranteed sessions and 50% high-bandwidth sessions. The Shortest-dist$(P, 1)$ is used for best-effort traffic.

Shortest-dist$(P, 1)$ path algorithm and is therefore less "agile", i.e. it cannot avoid links with low residual bandwidth.

Furthermore, when comparing Figures 7.5 and 7.11, we see that the use of multi-class routing results in a higher average throughput improvement for the Shortest-dist$(P, 0.5)$ path algorithm than for the Shortest-dist$(P, 1)$ path algorithm. The reason is that inter-class routing is less



(a) even guaranteed and uneven best effort



(b) uneven guaranteed and uneven best effort

Figure 7.11: Average throughput as a function of network load: cluster topology, 50% guaranteed sessions and 50% high-bandwidth sessions, 90% of maximal reservation ratio. The Shortest-dist$(P, 0.5)$ is used for best-effort traffic

likely to create links that are bottlenecks for best effort traffic, i.e. with high best effort traffic loads relative to the residual bandwidth, which is the scenario that the Shortest-dist$(P, 0.5)$ path algorithm does not handle well.

7.4 Discussion

In the previous sections, we focused on the case where the high priority traffic class requires bandwidth guarantees. However, we believe that the algorithm and insight should be useful for a broader set of service classes. The reason is simple: when multiple service classes coexist in the network, the notion of differentiating between these service classes by giving one class the first chance to claim bandwidth seems both typical and intuitive. For example, when allocating connections with delay guarantees according to the IETF guaranteed service class, delay is controlled through bandwidth. In [68], an iterative Bellman-Ford (IBF) routing algorithm for traffic with delay guarantees is described, and the link residual bandwidth used there could be replaced by virtual residual bandwidth.

Recently, it has been suggested [16, 76] that per-flow state information can be kept in the edge nodes of the network while the nodes inside the network maintain only the aggregated state information of these flows. A flow of differentiated service has higher priority than a best effort flow. To admit a differentiated flow, the network must know the resource availability inside the network. Hence, source routing based on link residual bandwidth is likely to be adopted in the network. Using the inter-class sharing model, we can route the flows of differentiated service around links that are congested with best effort sessions.

We have focused on multi-class routing for a network with two classes of services. For a network that supports more than two classes of services, if all but the best effort service require resource reservation, our approach should be directly applicable. If the network supports more than two services with strict priority, our approach does not directly apply. However, we can extend the credit calculation scheme of Equation (7.4) in Section 7.2.2 to consider the traffic load of service classes with lower priority. For example, we can assign credit for each of these service classes, but use different weights (i.e. the slopes in Figure 7.3) for different service classes. More research is needed to define and evaluate such extensions.

7.5 Related Work

While several studies have addressed routing support for multiple classes of service, inter-class resource sharing for data networks has not been addressed.

Matt and Shankar have studied delay and throughput based type-of-service routing in [72]. IBM's System Network Architecture (SNA) introduces the concept of Class of Service (COS)-based routing [1, 40] where several classes of service: interactive, batch, and network control, were used. In addition, users can define additional classes. When starting a data session, an application or device would request a COS. Routing would then map the COS into a statically configured route which marks a path across the physical network. These studies did not address routing support for inter-class resource sharing.

More recently, a routing scheme called Real-Time Network Routing (RTNR) [3] is proposed for a fully-connected circuit-switched network. RTNR handles multiple classes of service, including voice and data at fixed rates. It utilizes a sophisticated per-class trunk reservation mechanism with dynamic bandwidth sharing between classes. When alternate routing is required, RTNR utilizes the loading on all trunks in the network to select a path.

In contrast to these studies, we address routing support for resource sharing among service classes with different priorities. Our inter-class sharing scheme is simple and general. It does not require data traffic to be sent at a fixed rate, nor does our scheme require a trunk reservation mechanism with dynamic bandwidth sharing between classes.

7.6 Summary

In this chapter, we showed that multi-class routing is more than simply putting together optimal routing algorithms of individual traffic classes. An inter-class level of resource balancing is essential to achieving high network throughput when the network load is unevenly distributed. Instead of statically partitioning link resources across various service classes, we used a dynamic link resource sharing scheme and let the routing algorithm determine how link resources should be split among flows of different service classes.

At the core of our multi-class routing algorithm is a concept of "virtual residual bandwidth", which is the real residual bandwidth adjusted to account for the link congestion conditions for the best effort traffic. The key idea is to direct guaranteed sessions to the paths that are less congested with best-effort traffic. The algorithm does not require any change in the routing algorithms used for individual traffic classes and it is applicable to routing algorithms that use residual bandwidth as link state.

The simulation results show the effectiveness of the proposed dynamic inter-class resources sharing models. Compared with the multi-class routing algorithm that does not address inter-class resource sharing, we observe significant performance improvement for best-effort traffic when the load is heavy and unevenly distributed. This improvement is more significant if the volume of best-effort traffic is larger, that is, there are more best-effort sessions that are unevenly distributed, a higher portion of link capacity is reservable, and the routing algorithm for best-effort traffic is more likely to select the minimum-hop paths. For many scenarios, the local performance improvement in regions with uneven load distribution is significant. We did not observe any performance penalty in regions of the network where the load is evenly distributed.

Chapter 8

Conclusions

Future integrated services networks will support multiple classes of service to meet the diverse QoS requirements of applications. Some of these QoS requirements impose strict resource constraints on the paths being used to ensure end-to-end performance guarantees, while other applications desire best-effort high throughput. The goal of QoS routing is to select paths that satisfy these constraints while achieving high resource efficiency.

QoS routing is challenging because of the complexity of selecting a path with multiple QoS constraints as is needed for traffic with delay guarantees and because of the diversity of QoS requirements for different traffic classes. The different intra-class sharing behavior in different service classes and the dynamics of inter-class sharing of network resources among the traffic classes make it difficult to achieve efficient resource utilization under a wide range of traffic conditions. Routing algorithms are at the core of the QoS routing protocols that address these issues.

This thesis presented a study of QoS routing algorithms for a network that provides multiple classes of service with delay guarantees, bandwidth guarantees, and best effort high throughput requirements. This chapter summarizes our results and presents some direction for future research.

8.1 Contributions

In this dissertation, we demonstrated that QoS routing is both desirable and feasible. Toward this goal, we developed a two-level integrated QoS routing framework. At the intra-class level, routing algorithms for individual service classes are developed. These algorithms achieve high intra-class resource sharing efficiency and have computational costs within a constant factor of traditional shortest path algorithms. At the inter-class level, a dynamic resource sharing algorithm is proposed. This algorithm provides a simple way to balance the competing demands of different traffic classes to achieve high inter-class sharing efficiency.

The approach taken is pragmatic. We considered practical service models, realistic network topologies, and diverse traffic loads, and focused on practical routing algorithms. The proposed routing framework is evaluated through an extensive set of simulations. This dissertation makes the following original contributions.

For best-effort traffic, we focused on traffic that requires high throughput (e.g., bulk data transfer). We developed polynomial distance based routing algorithms that make use of congestion state information—the max-min fair share rate—as link state. Compared with minimal-hop routing and shortest-widest path routing, the Shortest-dist$(P, 1)$ path algorithm and Shortest-dist$(P, 0.5)$ path algorithm perform consistently well for a variety of traffic load and network topologies. To achieve even higher bandwidth without reducing the fair share of single-path sessions, we proposed a novel prioritized multi-path routing algorithm, in which lower priority paths share the bandwidth left unused by higher priority paths. This leads to a new multi-level prioritized max-min fairness model. Simulation results show an increase of up to 35% in throughput by using a second path, while single-path sessions also enjoy an increase of up to 5%.

For traffic requiring bandwidth guarantees, we evaluated several promising routing algorithms. The evaluation considers not only the call blocking rate but also the fairness for requests with different bandwidths, robustness to inaccurate routing information, and sensitivity to the change of the routing information update interval. We showed that routing algorithms that favor fewer hops and at the same time also balance load, e.g., the shortest-distance path algorithm, perform well. The bandwidth efficiency of using pre-computed paths for bandwidth intervals is comparable to that of computing paths on demand, which implies the feasibility of class-based routing. As long as the routing information update interval is significantly smaller than the mean session holding time, the blocking rate stays stable although an increased number of sessions are misrouted.

For traffic requiring delay guarantees, the general QoS routing problem of finding a path that meets delay, delay jitter, and buffer space constraints is NP-complete. However, we showed that QoS routing is polynomial if the network service discipline is rate-proportional (e.g., weighted fair queueing) and the relationship between QoS constraints is exploited in the algorithm design. The resulting algorithms simply iterate the Bellman-Ford algorithm over all residual bandwidth values. To achieve high network throughput, we identified four candidate optimality criteria: bandwidth to reserve, hop count, path load, and delay. The simulation results show that an efficient path must consider both load balancing and resource consumption in terms of hop count and the amount of bandwidth to reserve in order to perform consistently well. While the widest-shortest path algorithm outperforms other algorithms, the shortest-delay path algorithm has an edge when the load is light and unevenly distributed. Furthermore, we proposed an approximation algorithm for the iterative Bellman-Ford algorithm that only requires a constant number of iterations of the Bellman-Ford algorithm. These approximation algorithms run almost eight times faster than the original algorithms for our MCI topology. Notably, the approximation algorithm for the shortest-delay path performs consistently well for different topologies and load distributions.

For an integrated service network, an architecture that simply combines per-class routing algorithms may cause serious congestion for low priority traffic, because the algorithms for guaranteed traffic can take bandwidth away from best-effort traffic to optimize performance for guaranteed sessions only. On the other hand, a fixed partition of the link capacity to different traffic classes can be inefficient. We introduced a simple dynamic inter-class resource sharing architecture that takes the load of lower priority traffic into consideration while routing a higher priority session. The resulting multi-class routing algorithm performs consistently well as long as the guaranteed sessions are not the dominant traffic classes when the network load is heavy. In cases when the network load is unevenly distributed, it can improve the performance for low priority traffic significantly without sacrificing any performance for high priority traffic.

In summary, QoS routing is desirable because using carefully selected routes can significantly reduce the blocking rate for guaranteed traffic and improve the throughput for best-effort traffic. QoS routing is also feasible because routes that meet QoS constraints can be selected by either directly using shortest path algorithms, e.g. when routing best-effort traffic and routing traffic with bandwidth guarantees, or by iterating the shortest path algorithm a constant number of times, e.g. when routing traffic with delay guarantees. Even in the region of light guaranteed traffic load where the call blocking rate is not an issue, QoS routing for guaranteed traffic can be an effective mechanism to direct traffic load, and therefore, to avoid congestion or starvation of best-effort traffic. Our study suggests that there is no need for substantial changes to existing routing protocols as long as the link state information requested by these routing algorithms is advertised in the network.

8.2 Future Work

This dissertation has mainly focused on developing practical QoS routing algorithms that can make efficient use of network resources and are computationally inexpensive. Although it is a very important first step towards making QoS routing a reality, some issues remain to be addressed while others need to be explored further.

One important issue to address is scalability. Hierarchical aggregation is a common technique for scaling. For example, the Internet relies on separate intra-domain (e.g., OSPF) and inter-domain (e.g., BGP) routing algorithms, while ATM uses the PNNI routing protocol [27], which supports a multi-level hierarchy. The QoS routing algorithms we proposed can essentially be applied to routing at different hierarchical levels (e.g., both the intra-domain and inter-domain level). The challenge is how to aggregate the network state information without significant loss of accuracy. In [62], the complexity of routing algorithms that consider the inaccuracy of the routing information is studied, but the authors assume that the accuracy of the link state is known. In practice, the degree of inaccuracy depends on how network state information is aggregated, the frequency of the routing information updates, and network load, and it is therefore unknown. More research is needed to be better understand these issues and to develop new techniques for aggregating link state with multiple attributes.

Another scalability concern is the increase in routing table size, since flows with different QoS requirements may require the selection of different paths. The problem of routing table size can be addressed through employing source routing or class-based routing as discussed in Chapter 5. However, it is not clear how some of the techniques extend to other traffic classes (e.g. traffic with delay guarantees), and the general scalability issue for per-flow reservations, as supported through RSVP, still remains.

QoS-based multicast routing is another important issue that needs investigation. It is challenging because different receivers may require different QoS and the receiver set can change over time. Other factors that make QoS-based multicast routing challenging include scalability issues when the groups can be very large and shared reservations are supported. Even if multicast sessions are established using the simple shortest path tree routing algorithm, it is not clear which of the unicast routing algorithms discussed in Chapters 4, 6, and 5 will achieve better resource efficiency, when they are used to construct the shortest-path tree.

Other areas where more research is needed include expanding the set of service models used in our study (e.g., routing for the recently proposed differentiated services [16, 76]), investigating the impact of call holding time, and comparing the performance of different routing information distribution mechanisms (e.g., updates triggered by load change versus by fixed time interval).

Bibliography

[1] V. Ahuja. Routing and Flow Control in SNA. *IBM Systems Journal*, 18(2):298–314, 1979.

[2] J.M. Akinpelu. The Overload Performance of Engineered Networks with Nonhierarchical and Hierachical Routing. *Bell System Technical Journal*, pages 1261–1281, September 1984.

[3] G.R. Ash, J.S. Chen, A.E. Frey, and B. D. Huang. Realtime network routing in a dynamic class-of-service network. In *Proceedings of the 13h ITC*. ITC, 1992.

[4] ATM Forum Traffic Management Specification Version 4.0, October 1995. ATM Forum/95-0013R8.

[5] S. Bahk and M. Elzarki. Dynamic Multipath Routing and How it Compares with Other Dynamic Routing Algorithms for High Speed Wide Area Networks. *ACM SIGCOMM 92*, September 1992.

[6] J.C.R. Bennett and H. Zhang. WF^2Q: Worst-case Fair Weighted Fair Queueing. In *Proceedings of IEEE INFOCOM'96*, May 1996.

[7] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1987.

[8] D.P. Bertsekas. Dynamic Behavior of Shortest Path Routing Algorithms for Communication Networks. *IEEE Transactions on Auto. Control*, AC-27, 1982.

[9] V.A. Bolotin. Modeling Call Holding Time Distributions for CCS Network Design and Performance Analysis. *IEEE JSAC*, 12(3):433–438, April 1994.

[10] F. Bonomi and K. Fendick. The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service. *IEEE Network*, 9(2):25–39, March/April 1995.

[11] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. *IETF RFC 1633*, June 1994.

[12] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. *IETF RFC 2205*, September 1997.

[13] L. Breslau, D. Estrin, and L. Zhang. A Simulation Study of Adaptive Source Routing in Integrated Service Networks. *USC CSD Technical Report*, Sep., 1993.

[14] A. Charny, D.D. Clark, and R. Jain. Congestion Control With Explicit Rate Indication. In *Proc. ICC'95*, June 1995.

[15] A. Charny, K.K. Ramakrishnan, and A. Lauck. Scalability Issues for Distributed Explicit Rate Allocation in ATM. In *Proc. IEEE INFOCOM'96*, 1996.

[16] D. Clark and J. Wroclawski. An Approach to Service Allocation in the Internet. *IETF Internet Draft <draft-clark-diff-svc-alloc-00.txt>*, July 1997.

[17] David Clark, Scott Shenker, and Lixia Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. *ACM SIGCOMM 92*, August 1992.

[18] David D. Clark. Adding Service Discrimination to the Internet. Preprint, 1995.

[19] David D. Clark, Van Jacobson, John Romkey, and Howard Salwen. An analysis of tcp processing overhead. *IEEE Communications Magazine*, pages 23–29, June 1989.

[20] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorihtms*. The MIT Press, Cambridge, MA, 1990.

[21] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet. *IETF Internet Draft <draft-ietf-qosr-framework-01.txt>*, July 1997.

[22] R.L. Cruz. Quality of Service Guarantees in Virtual Circuit Switched Networks. *IEEE JSAC*, pages 1048–1056, August 1996.

[23] Peter B. Danzig, Sugih Jamin, Ramon Caceres, and Danny Mitzel. Characteristics of wide-area TCP/IP conversations. *ACM SIGCOMM 91 Conference*, September, 1991.

[24] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. *IETF Internet Draft <draft-ietf-ipngwg-ipv6-spec-v2-01.txt>*, November 21 1997.

[25] Juan Miguel del Rosario and Alok Choudhary. High performance I/O for parallel computers: Problems and prospects. *IEEE Computer*, 27(3):59–68, March 1994.

[26] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *ACM SIGCOMM 89*, 19(4):2–12, August 19-22, 1989.

[27] PNNI SWG (Doug Dykeman ed.). PNNI Draft Specification. ATM Forum 94-0471R10, October 1995.

[28] E.W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, pages 1:269–271, 1959.

[29] S. Floyd and V. Jacobson. Link-sharing and Resource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, August 1995.

[30] S. Floyd and V. Jacobson. Random Early Detection for Congestion Avoidence. *IEEE/ACM Transactions on Networking*, 4(1):397–413, August, 1993.

[31] A. Fraser. Towards a Universal Data Transport System. *IEEE JSAC*, pages 803–816, Nov 1983.

[32] J.J. Garcia-Luna-Aceves. Routing Management in Very Large-Scale Networks. *Future Generation Computer Systems*, 4, 1988.

[33] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979. (page 214).

[34] Rainer Gawlick, Charles Kalmanek, and K.G. Ramakrishnan. On-line Routing for Permanent Virtual Circuits. In *Proc. IEEE INFOCOM'95*, 1995.

[35] R.J. Gibbens, F.P. Kelley, and P.B. Key. Dynamic Alternative Routing — Modelling and Behaviour. In *Proceedings of the 12th International Teletraffic Congress*, June 1988.

[36] G.A. Gibson, D.F. Nagle, K. Amiri, F.W. Chang, E.M. Feinberg, H. Gobioff, C. Lee, B. Ozceri, E. Riedel*, D. Rochberg, and J. Zelenka. File Server Scaling with Network-Attached Secure Disks. In *ACM SIGMETRICS '97*, pages 272–284, Seattle, Washington, June 15-18 1997. ACM.

[37] G.A. Gibson, D. Stodolsky, F.W. Chang, W.V. Courtright II, C.G. Demetriou, E. Ginting, M. Holland, L. Neal Q. Ma, R.H. Patterson, R. Youssef J. Su, and Jim Zelenka. The Scotch Parallel Storage Systems. In *Proceedings of the IEEE CompCon Conference*. San Francisco, CA, March 5-8 1995.

[38] S. Golestani. A Self-Clocked Fair Queueing Scheme for Broadband Applications. In *Proceedings of IEEE INFOCOM'94*, pages 636–646, Toronto, Canada, June 1994.

[39] P. Goyal and H.M. Vin. Generalized Guaranteed Rate Scheduling Algorithms: A Framework. Technical Report Technical Report TR95-30, Department of Computer Science, UT Austin, Texas, 1995.

[40] J.P. Gray and T.B. McNeil. SNA Multi-System Networking. *IBM Systems Journal*, 18(2):263–297, 1979.

[41] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic. *IEEE/ACM Transactions on Networking*, To appear, December, 1998.

[42] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks. *IEEE JSAC*, 9(7):968–981, September 1991.

[43] R. Guerin and A. Orda. QoS-based Routing in Networks with Inaccurate Information: Theory and Algorithms. In *Proc. IEEE INFOCOM'97*, 1997.

[44] R. Guerin, A. Orda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions. *IETF Internet Draft <draft-guerin-qos-routing-ospf-00.txt>*, November 1996.

[45] S. Gupta, K.W. Ross, and M. El Zarki. On Routing in ATM Networks. In M. Steenstrup, editor, *Routing in Communications Networks*, pages 49–74. Prentice Hall, 1995.

[46] E. Hahne. Round-Robin Scheduling for MaxMin Fairness in Data Networks. *IEEE Journal on Selected Areas in Communications*, 9(7), September 1991.

[47] C Hedrick. Routing Inofrmation Protocol. RFC 1058, SRI Network Information Center, June 1988.

[48] Van Jacobson. Congestion Avoidance and Control. *ACM SIGCOMM 88*, pages 273–288, 1988.

[49] J. Jaffe. Algorithms for Finding Paths with Multiple Constraints. *Networks*, 14(1):95–116, Spring 1984.

[50] J.M. Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, COM-29(7):954–962, July 1981. Correspondence.

[51] R. Jain. *The Art of Computer Performance Analysis*. Wiley, 1991.

[52] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and R. Vishwanathan. ERICA Switch Algorithm: A Complete Description. *AF-TM 96-11721, ATM Forum Traffic Management Working Group*, August 1996.

[53] S. Jamin, P. Danzig, S. Shenker, and L. Zhang. A Measurement-based Admission Control Algorithm for Integrated Services Packet Networks. In *Proceedings of SIGCOMM'95*, pages 2–13, Boston, MA, September 1995.

[54] S. Jamin, S. Shenker, and P. Danzig. Comparison of Measurement-based Admission Control Algorithms for Controlled-Load Service. In *Proceedings of IEEE INFOCOM'97*, Kobe, Japan, April 1997.

[55] L. Kalampoukas, A. Varma, and K.K. Ramakrishnan. Rate Allocation Algorithm for ATM Networks Providing Max-Min Fairness. In *Proceedings of 6th IFIP International Conf. on High Performance Networking, HPN '95*, Palma De Mallorca, Balearic Islands, Spain, September 11-15 1995.

[56] L. Kleinrock and F. Kamoun. Hierarchical Routing for Large Networks. *Computer Networks*, 1, 1977.

[57] C. Kosak, D. Eckhardt, T. Mummert, P. Steenkiste, and A. Fisher. Buffer Management and Flow Control in the Credit Net ATM Host Interface. In *Proceedings of the 20th Conference on Local Computer Networks*, pages 370–378, Minneapolis, October 1995. IEEE.

[58] H.T. Kung, T. Blackwell, and A. Chapman. Credit Update Protocol for Flow-Controlled ATM Networks: Statistical Mulitplexing and Adaptive Credit Allocation. In *Proceedings of the SIGCOMM '94 Symposium on Communications Architectures and Protocols*, pages 101–114. ACM, August 1994.

[59] H.T. Kung and K. Chang. Receiver-Oriented Adaptive Buffer Allocation in Credit-Based Flow Control for ATM Networks. In *IEEE INFOCOM '95*, volume 1, pages 239–252, Boston, Massachusetts, April 1995. IEEE.

[60] H.T. Kung and Robert Morris. Credit-Based Flow Control for ATM Networks. *IEEE Network Magazine*, 9(2):40–48, March/April 1995.

[61] K. Lang and S. Rao. Finding Near-Optimal Cuts: An Empirical Evaluation. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 212–221, 1993, Austin, Texas.

[62] W.C. Lee. Topology Aggregation for Hierarchical Routing in ATM Networks. *ACM Computer Communication Review*, 25(2):82–92, April 1995.

[63] W.C. Lee, M.G. Hluchyj, and P.A. Humblet. Routing Subject to Quality of Service Constraints in Integrated Communication Networks. *IEEE Network*, 9(4):14–16, July/August, 1995.

[64] J. Liebeherr, I.F. Akyildiz, and A. Tai. A Multi-level Explicit Rate Control Scheme for ABR Traffic with Heterogeneous Service Requirements. *Submitted for Publication*, July 1995.

[65] D. Lin and R. Morris. Dynamics of Random Early Detection. In *ACM SIGCOMM '97*, pages 115–126, Cannes, France, Sepetember 1997.

[66] Q. Ma and K.K. Ramakrishnan. Queue Management for Explicit Rate Based Congestion Control. In *ACM SIGMETRICS '97*, pages 39–51, Seattle, Washington, June 15-18 1997. ACM.

[67] Q. Ma and P. Steenkiste. On Path Selection for Traffic with Bandwidth Guarantees. In *IEEE International Conference on Network Protocols*, Atlanta, Georgia, October 1997.

[68] Q. Ma and P. Steenkiste. Quality-of-Service Routing for Traffic with Performance Guarantees. In *IFIP Fifth International Workshop on Quality of Service*, pages 115–126, NY, NY, May 1997.

[69] Q. Ma and P. Steenkiste. Routing Traffic with Quality-of-Service Guarantees in Integrated Services Networks. In *The 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 98)*, Cambridge, England, July 1998.

[70] Q. Ma, P. Steenkiste, and H. Zhang. Routing High-Bandwidth Traffic in Max-Min Fair Share Networks. In *ACM SIGCOMM '96*, pages 206–217, Stanford, CA, August 1996.

[71] I. Matta and A. U. Shanka. Dynamic Routing of Real-Time Virtual Circuits. In *Proc. IEEE INCP'96*, 1996.

[72] I. Matta and A. U. Shankar. Type-of-Service Routing in Datagram Delivery Systems. *IEEE JSAC*, 13(8):1411–1425, October 1995.

[73] J.M. McQuillan, I. Richer, and E. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[74] J. Moy. OSPF Version 2. *IETF RFC 2178*, October 1997.

[75] D.J. Nelson, K. Sayood, and H. Chang. An Extended Least-hop Distributed Routing Algorithm. *IEEE Transactions on Communications*, COM-38(4):520–528, April 1990.

[76] K. Nichols, V. Jacobson, and L. Zhang. A Two-bit Differentiated Services Architecture for the Internet. *IETF Internet Draft <draft-nichols-diff-svc-arch-00.tex>*, Nov 1997.

[77] A. Parekh and R. Gallager. A Generalized Processor Sharing Approach to Flow Control-the Single Node Case. *IEEE/ACM Transactions on Networking*, 3(1):344–357, June, 1993.

[78] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[79] C. Pornavalai, N. Shiratori, and G. Chakraborty. QoS Based Routing Algorithm in Integrated Services Packet Networks. In *IEEE International Conference on Network Protocols*, Atlanta, Georgia, October 1997.

[80] J. B. Postel. Transmission Control Protocol. RFC 793, September 1981.

[81] A. B. Przygienda. *Link State Routing with QoS in ATM LANs*. PhD thesis, Dipl. Inf. Ing. ETH, 1995.

[82] K. Ramakrishnan, D. Chiu, and R. Jain. Congestion avoidance in computer networks with a connectionless network layer. In *ACM SIGCOMM'88*, pages 303–313, Stanford, CA, August 1988.

[83] K. K. Ramakrishnan and Peter Neuman. Integration of Rate and Credit Schemes for ATM Flow Control. *IEEE Network Magazine*, 9(2):49–56, March/April 1995.

[84] K.K. Ramakrishnan and S. Floyd. A Proposal to Add Explicit Congestion Notification (ECN) to IPv6 and to TCP. *IETF Internet Draft <draft-kksjf-ecn-00.txt>*, November 1997.

[85] K.K. Ramakrishnan and Raj Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Transactions on Computer Systems*, 2(8):158–181, May 1990.

[86] K.K. Ramakrishnan, Raj Jain, and Dah-Ming Chiu. Congestion Avoidance in Computer Networks With a Connectionless Network Layel. Part IV: A Selective Binary Feedback Scheme for General Topologies Methodology. Technical Report DEC-TR-510, Digital Equipment Corporation, 1987.

[87] S. Rampal. *Routing and End-toend Quality of Service in Multimedia Networks*. PhD thesis, North Carolina State University, August, 1995.

[88] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). *IETF RFC 1771*, March 1995.

[89] J.B. Rosen, S.Z. Sun, and G.L. Xue. Algorithms for the Quickest Path Problem. *Computers and Operations Research*, 18(6):579–584, 1991.

[90] H.F. Salama, D.S. Reeves, and Y. Viniotis. A Distributed Algorithm for Delay-Constrained Routing. In *Proceedings of IEEE INFOCOM'97*, Kobe, Japan, April 1997.

[91] M. Schwartz and T.E. Stern. Routing Techniques Used in Computer Communication Networks. *IEEE Transactions on Communications*, COM-28(4), April 1980.

[92] S. Shenker, D.D. Clark, and L. Zhang. A Scheduling Service Model and a Scheduling Architecture for an Integrated Service Packet Network. *preprint*, 1993.

[93] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. *IETF Internet Draft <draft-ietf-intserv-guaranteed-svc-07.txt>*, February 1997.

[94] J. Sole-Pareta, D. Sarkar, J. Liebeherr, and I.F. Akyildiz. Adaptive Multipath Routing of Connectionless Traffic in an ATM Network. In *Proc. IEEE ICC'95*, May 1995.

[95] M. Steenstrup. *Routing In Communications Networks*. Prentice Hall, Englewood Cliffs, NJ 07362, 1995.

[96] Martha Steenstrup. Inter-Domain Policy Routing Protocol Specification: Version 1. RFC 1479, July 1993.

[97] D. Stiliadis and A. Varma. Frame-based Fair Queueing: A New Traffic Scheduling Algorithm for Packet-Switched Networks. In *ACM SIGMETRICS 96*, Philadelphia, PA, May 1996.

[98] D. Stiliadis and A. Varma. A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms. In *Proceedings of IEEE INFOCOM'97*, Kobe, Japan, April 1997.

[99] H. Suzuki and F. A. Tobagi. Fast Bandwidth Reservation Scheme with Multi-link and Multi-path Routing in ATM Networks. In *Proc. IEEE INFOCOM'92*, 1992.

[100] C. Topolcic. Experimental Internet Stream Protocol, version 2 (ST-II). *IETF RFC 1190*, October 1990.

[101] Z. Wang and J. Crowcroft. Shortest Path First with Emergency Exits. *ACM SIGCOMM 90*, September 1991.

[102] Z. Wang and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE JSAC*, 14(7):1288–1234, September 1996.

[103] J. Wroclawski. Specification of the Controlled-Load Network Element Service. *IETF Internet Draft <draft-ietf-intserv-ctrl-load-svc-04.txt>*, November 1996.

[104] D. Zappala, D. Estrin, and S. Shenker. Alternate Path Routing and Pinning for Interdomain Multicast Routing. Technical Report 97-665, USC Computer Science Department, 1997.

[105] H. Zhang. Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, 83(10), October 1995.

[106] L. Zhang. Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks. *SIGCOMM 90*, pages 19–29, 1990.

[107] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Communications Magazine*, 31(9):8–18, September 1993.

[108] W. Zhao and S. Tripathi. Routing Guaranteed Quality of Service Connections in Integrated Services Packet Networks. In *IEEE International Conference on Network Protocols*, Atlanta, Georgia, October 1997.