

**A Study of Bandwidth-sharing Mechanisms in Connection-oriented
Networks**

A Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy

Computer Science

by

Xiangfei Zhu

May 2008

© Copyright by

Xiangfei Zhu

All rights reserved

May 2008

APPROVAL SHEET

This dissertation is submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
Computer Science

Xiangfei Zhu

This dissertation has been read and approved by the examining committee:

Prof. Malathi Veeraraghavan (Advisor)

Prof. Alfred C. Weaver (Chair)

Prof. Paul F. Reynolds, Jr.

Prof. Marty Humphrey

Prof. Maité Bradt-Pearce

Accepted for the School of Engineering and Applied Science:

Dean, School of Engineering and Applied Science

May 2008

Abstract

There is an increasing interest in optical connection-oriented networks to support the high-speed and predictable-service requirements of applications used in the scientific research community. Various experimental testbeds have been created. One area of networking research in these testbed projects is dynamic bandwidth sharing because the number of universities and national research laboratories involved in these scientific projects is large enough to make the use of dedicated circuits (leased lines) prohibitively expensive. One reference point for dynamic bandwidth-sharing mechanisms in connection-oriented networks is the telephone network. We refer to the bandwidth-sharing mechanism used in the telephone network as the immediate-request (IR) mode because there is no provision for making advance reservations for circuits. While this mechanism can be used in new high-speed optical connection-oriented networks, we show that some applications require high per-call bandwidth, and this bandwidth level is a large fraction of even the highest-rate links in use today, the combination of which makes the IR mode unsuitable. Therefore, we need alternative bandwidth-sharing modes for these networks and applications.

We propose two new bandwidth-sharing mechanisms, both of which are book-ahead (BA) reservation schemes for high-speed connection-oriented networks. First, we define two types of reservation requests: i) session-type requests, which specify the required bandwidth and duration (useful for applications such as remote visualization), and ii) data-type requests, which specify the amount of data to be transferred (useful for file transfers). For *session-type requests*, we propose two variants: a **BA- n scheme**, in which a user specifies n acceptable call-initiation time options, and a **BA-First scheme**, in which a user accepts any call-initiation time. We construct two novel analytical models for these two variants of BA schemes. Using the analytical and simulation models,

we show that BA schemes overcome the limitation of the immediate-request mechanism, making it feasible to support applications even when the per-call bandwidth is large relative to link capacity. An additional purpose of these models is to assist implementors of these schemes as well as administrators of networks using these implementations by allowing them to test out the effects of different design choices and parameter values.

Our second book-ahead mechanism called **Varying-Bandwidth Delayed Start (VBDS)** is for *data-type* (file transfer) requests. By having applications specify file sizes in their reservation requests, a VBDS scheduler is able to assign a varying-bandwidth allocation for different time ranges instead of a single fixed-rate allocation based on bandwidth availability. Through simulations, we show that this approach improves system performance when compared to fixed-bandwidth schemes significantly. Furthermore, when we compare it to packet switching, we found that, for moderate-to-large-sized files, connection-oriented networking with VBDS is a strong alternative to packet switching for file transfers. VBDS favors large files while packet switching favors moderate-sized files. For small files, we do not recommend using connection-oriented networks because the scheduling overhead will be larger than the file transfer time as these networks operate at high speeds.

As a complement to the analytical and simulation studies used in the design of our two proposed BA schemes, we undertake an implementation and experimental study of the IR bandwidth-sharing mode in high-speed connection-oriented networks. We designed and deployed a three-node *connection-oriented* network called CHEETAH (Circuit-switched High-speed End-to-End Transport Architecture). Switches are located at Research Triangle Park, North Carolina, Atlanta, Georgia, and Oak Ridge, Tennessee. We designed and implemented a CHEETAH software package to run on Linux end hosts. Among other functions, this software package includes a signaling module to enable end users and applications to dynamically initiate requests for dedicated end-to-end circuits for immediate use, and receive/respond to requests when initiated by remote end hosts. We run experiments of IR mode call setups and releases and use these experiments to obtain measurements for end-to-end circuit setup delays and per-switch signaling message processing delays. These measurements are useful for analytical and simulation studies of connection-oriented networks, and also in network design and configurations.

Acknowledgments

The work presented in this dissertation is impossible without the guidance and support from my advisor, Professor Malathi Veeraraghavan. I would like to express my sincerest gratitude and appreciation to her, not only for her extensive knowledge and enlightening direction, but also for her continuous encouragement.

I thank Professor Alfred C. Weaver, Professor Paul F. Reynolds, Jr., Professor Marty Humphrey, and Professor Maité Bradt-Pearce for serving on my defense committee and providing constructive comments. I thank Hua Li and Professor Edwin K. P. Chong from the Colorado State University for providing the simulation results of the packet switching scheme in Chapter 4.

I thank the postdoctoral fellows and graduate students in our group, Xuan Zheng, Tao Li, Xiuduan Fang, Mark E. McGinley, Zhanxiang Huang, Helali Bhuiyan, and Anant Padmanath Mudambi, with whom I enjoyed collaboration and discussions. I also thank my fellow graduate students and friends, Xiang Yin, Xinyu Liu, Lingling Cui, Lingjia Tang, Qin Chen, Tenghui Zhu, Hao Huang, Gang Zhou, Jun Feng, and Lin Gu. They have made my life in Charlottesville enjoyable and memorable.

In addition, I would like to thank the National Science Foundation for funding this research.

Finally, I want to thank my wife Yan Liu, my parents Huajun Zhu and Jiaolan Sun, and my sister Xiangle Zhu, for their unconditional love and support. I dedicate this dissertation to them.

Contents

1	Introduction	1
1.1	Hypothesis and Background	1
1.2	Motivation and Related Work	5
1.3	Outline	8
1.4	Key Contributions	10
2	Book-ahead Mechanisms for Session-type Requests	12
2.1	Related Work	13
2.2	Book-ahead Mechanism	15
2.3	Analytical Model	16
2.4	Numerical Results	25
2.5	Sensitivity Analyses	37
2.6	Multi-link Book-ahead Schemes	42
2.7	Conclusions	46
3	An Improved Model for the BA-First Mechanism	48
3.1	The BA-First Mechanism	48
3.2	Analytical Model	49
3.3	Numerical Results	57
3.4	Conclusions	61
4	Book-ahead Mechanisms for Data-type Requests	62

<i>Contents</i>	viii
4.1 Motivation and Related Work	63
4.2 Problem Definition	65
4.3 VBDS Mechanism	66
4.4 Analysis and Simulation Results	71
4.5 Conclusions	80
5 Immediate-Request Bandwidth-Sharing Mode	81
5.1 Background and Related Work	83
5.2 CHEETAH Wide-area Network	86
5.3 End-Host CHEETAH Software	96
5.4 Call Setup Delay Measurements	101
5.5 Conclusions	104
6 Conclusions and Future Work	105
6.1 An Integrated View	105
6.2 Future Work	109
Bibliography	111

List of Figures

1.1	Example of a distributed queueing system for a multi-link network.	4
1.2	Internet2 Dynamic Circuit Network Topology (reprinted from [1]).	7
2.1	Book-ahead mechanism in the single-link scenario.	15
2.2	DTMC for an example system: $m = 1, K = 3, l = 2, h_1 = 1, h_2 = 2,$ and $n = 1$. . .	22
2.3	Performance of different bandwidth sharing mechanisms. $m = 1, l = 2, h_1 = 7,$ $h_2 = 9, r_1 = 0.1, r_2 = 0.9,$ and $K = 11$	29
2.4	Performance of different bandwidth sharing mechanisms. $m = 10, l = 1, H = 300,$ and $K = 2000$	31
2.5	Gaps in the advance-reservation horizon when call-initiation time is unrestricted. .	32
2.6	Performance of different bandwidth sharing mechanisms with two classes of calls. $m = 10, l = 2, h_1 = 100, h_2 = 300, r_1 = 0.3, r_2 = 0.7,$ and $K = 2000$	33
2.7	Performance of BA-3 and BA-First under different values of the advance-reservation horizon, K , and number of channels, m . $l = 1, h_1 = 200,$ and load= 100%.	35
2.8	Minimum K/H values: $m=2, H=300,$ offered load=100%, BA-First.	36
2.9	Bell-shaped call-initiation time distribution: $m = 10, K = 2000,$ offered load=96%. .	39
2.10	Multiclass calls under same offered load: $m = 10, K = 2000, H = 300,$ offered load= 100%	40
2.11	Multiclass calls under same call arrival rate, $m = 10, K = 6000, c_1 = 1, c_2 = 10$. . .	42
2.12	A linear network.	42
2.13	Performance of different bandwidth-sharing mechanisms in a 3-hop network. $L = 3,$ $m = 10, H = 1,$ and $K = 10$	43

2.14	Classified call blocking probabilities. $L = 3$, $m = 10$, $H = 1$, and $K = 10$	43
2.15	Utilization comparison of the regular schemes and the schemes with trunk reservations.	44
2.16	Dependence on the network scale.	45
3.1	An illustration of a BA-First scheduling system	49
3.2	Example transition diagram for $m = 2$ and $K = 4$	51
3.3	Restricted IR vs. BA-First under different values of m	58
3.4	The effect of reservation window size on the system performance.	58
3.5	Offered system load vs. reservation window size.	59
3.6	The comparison of BA-First mode and $M/D/m/p$ queueing model.	60
4.1	Example of the VBDS mechanism.	70
4.2	File latency comparison between analytical and simulation results.	73
4.3	A comparison of VBDS file latency and mean file transfer delay for files requesting same R_{max} ; link capacity is 100 channels.	73
4.4	Throughput comparison for different values of T_d (0.05, 0.5, 1, and 2 sec).	75
4.5	Utilization comparison for different values of T_d (0.05, 0.5, 1, and 2 sec).	75
4.6	Number of change points in $\gamma(t)$ for different values of T_d	76
4.7	A comparison of FBDS, PS, and VBDS throughputs.	78
4.8	A comparison of FBDS, PS, and VBDS normalized delays.	79
4.9	Comparison of the improvements to files of different size ($\rho = 0.94$, $R_{max} = 4$). . .	79
5.1	CHEETAH concept.	84
5.2	Wide-area CHEETAH testbed topology (data-plane).	87
5.3	IP addressing in the CHEETAH network.	89
5.4	Creating IPsec tunnels on the CHEETAH network control plane.	95
5.5	End-host CHEETAH software architecture.	97
5.6	RSVP-TE module architecture.	99

6.1 Different bandwidth-sharing mechanisms. 107

List of Tables

2.1	Transition matrix for the example Markov chain shown in Fig. 2.2.	24
2.2	Dependence of K/H on m : offered load=100%, BA-First.	37
4.1	Notation.	66
5.1	TE-link configuration at the MCNC Sycamore SN16000 switch.	90
5.2	Signaling delays incurred in setting up a circuit between zelda1 and wukong across the CHEETAH network	102

Chapter 1

Introduction

1.1 Hypothesis and Background

The hypothesis of this research is that well-designed algorithms employing immediate-request and book-ahead bandwidth-sharing mechanisms will lead to efficient utilization of modern high-speed connection-oriented networks. We test this hypothesis by designing, evaluating, and comparing various bandwidth-sharing mechanisms using analytical, simulation, and experimental methods.

1.1.1 Background on connection-oriented networks

Connection-oriented networks are networks in which an end-to-end circuit/virtual circuit(VC) is established prior to data transfer. There are two types of connection-oriented networks, packet switched and circuit switched. Packet-switched connection-oriented networks are also referred to as virtual circuit networks. Examples of packet-switched connection-oriented networks include X.25 and Asynchronous Transfer Mode (ATM) networks. Examples of circuit-switched (connection-oriented) networks include Synchronous Optical NETWORKing (SONET), Wavelength-Division Multiplexing (WDM), and the telephone network.

The bandwidth-sharing mode used in the telephone network is an immediate-request call-blocking mode. As call requests arrive, channels are allocated on a first-come first-service basis until the link capacity runs out at which point calls are blocked (rejected). As calls complete, channels are freed for use by newly arriving calls. The bandwidth of high-speed circuit-switched

networks, such as SONET and WDM networks, is currently not shared in this mode. Providers only offer a leased-line service on these high-speed connection-oriented networks. To subscribe to a leased line service, a customer typically signs a contract with a service provider to hold the circuit for a long duration (on the order of months to years). Leased lines are still quite expensive at high speeds, due to a lack of sharing.

To enable dynamic call-by-call bandwidth-sharing in high-speed connection-oriented networks, Generalized Multi-Protocol Switched (GMPLS) control-plane protocols [2] were invented. These control-plane protocols consist of signaling protocols and routing protocols. Signaling protocols, such as Resource Reservation Protocol - Traffic Engineering (RSVP-TE), are used to set up and release circuits on a call-by-call basis [3]. Routing protocols, such as Open Shortest Path First - Traffic Engineering (OSPF-TE), are used to disseminate topology, reachability and loading conditions [4]. Such information is necessary to support the call setup procedure.

The protocols defined as part of the GMPLS suite support only an immediate-request mode of bandwidth sharing. However, prior research [5] shows that if the per-channel bandwidth requirement of applications is high relative to link capacity, it is difficult to combine high utilization with a low call blocking probability using the immediate-request mode of bandwidth sharing. Many of the applications used in large eScience projects, such as the Large Hadron Collision (LHC) project [6], require high per-channel bandwidth relative to link capacities. Therefore, to support such applications, we need alternative bandwidth-sharing modes in high-speed connection-oriented networks.

1.1.2 Background on general resource-sharing mechanisms

There are two mechanisms to share resources, queueing and advance reservation (book-ahead). The fundamental difference between these two mechanisms is that a reservation system has a reservation phase, while a queueing system does not. In a reservation system, a customer specifies service requirements in the reservation phase, and the system scheduler returns a service schedule based on resource availability and the customer's requirements. The request is rejected if the system cannot meet the customer's requirements. In contrast, a customer in a queueing system simply arrives for service when needed. If all servers are busy, the customer waits in a queue until a server becomes

available.

We consider two types of requests in reservation systems: session-type requests and data-type requests. A **session-type** request specifies a required number of servers and duration, while a **data-type** request specifies a required number of service units and an acceptable range of the number of servers that the customer can utilize simultaneously [7]. The scheduler returns an allocation with a certain number of servers for a certain duration to meet the service units requirement. Examples of session-type requests include the reservation of a conference room, and the reservation of a circuit for a remote-visualization session or a video conference. Examples of data-type requests include the reservation of bandwidth for file transfers. At the time that a reservation is made, a customer explicitly or implicitly specifies the desired service parameters. These parameters may be random variables following certain distributions rather than deterministic variables. Therefore, the schedules returned by the scheduler may also be non-deterministic. For example, in a doctor's office, it is difficult to predict the duration of an appointment; therefore, a patient may find that she needs to wait for a short period when she arrives at the office at her assigned appointment time.

1.1.3 Hypothesis formulation

We apply the general resource-sharing mechanisms described in Section 1.1.2 to our problem of bandwidth sharing in connection-oriented networks. For the reservation-based modes, we assume that the service parameters provided in the reservation requests are deterministic, and hence the schedules returned by the scheduler are also deterministic.

Queueing systems are often described using Kendall's notation $A/B/m/k$, where A describes the arrival process, B describes the service time distribution, m indicates the number of servers, and k describes the capacity of the system, i.e., the maximum number of customers in the system including those in service. A queueing system can be a **bufferless queueing** system if the system does not provide buffer space, i.e., $k = m$, or a **buffered queueing** system if it provides a non-zero buffer, i.e., $k > m$.

In the following paragraph, using an example, we illustrate why it is feasible to only use bufferless queueing for calls in a connection-oriented network. Fig. 1.1 shows the topology of a network

with three switches and eight end hosts. X_1 , X_2 , and X_3 are circuit switches or connection-oriented packet switches. H_1, \dots, H_8 are end hosts connected to these switches. Call C_1 requests a circuit from host H_1 to H_2 , and the call is routed through switches X_1 , X_2 and X_3 . Call C_2 arrives after Call C_1 and requests a circuit from host H_3 to H_7 . We consider two approaches for the call setup process, a distributed approach and a centralized approach. In the distributed approach, each switch has a switch controller, which accepts a call-setup request and checks for bandwidth availability. If the requested bandwidth is available, the switch controller reserves the bandwidth for the call, and sends the request to the next hop. If the requested bandwidth is unavailable, the switch controller's behavior depends on whether it uses buffered queueing or bufferless queueing for these call-setup requests. The call is rejected immediately if bufferless queueing is used. On the other hand, if the controller has a buffer to hold call-setup requests, the request waits in the queue. Consider call C_1 in Fig. 1.1 as an example. If buffered queueing is used, and the bandwidth requested by C_1 is available on links H_1X_1 and X_1X_2 , but unavailable on link X_2X_3 , C_1 is placed in the queue of switch X_2 by its switch controller. During its wait period in this queue, the bandwidth reserved in links H_1X_1 and X_1X_2 for call C_1 remains idle. Such situation will cause a poor network utilization. Another problem with buffered queueing is the potential for deadlocks caused by simultaneous calls from opposite directions.

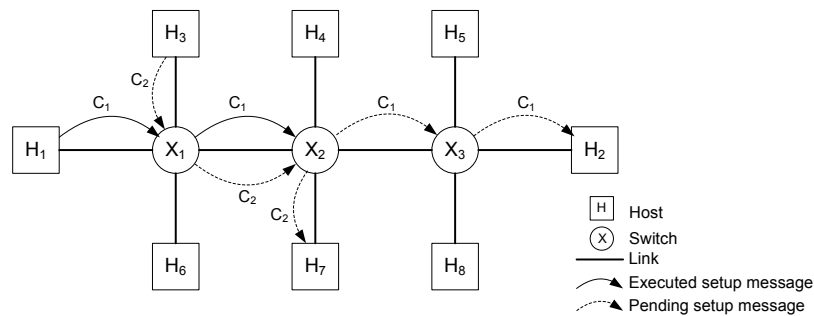


Figure 1.1: Example of a distributed queueing system for a multi-link network.

In the centralized approach, a centralized switch controller receives call-setup requests for the whole network. System utilization can be improved if the controller schedules calls out-of-sequence. If bandwidth is not available on all links of the route for a call that arrived first, a second

call waiting in the queue could be scheduled if bandwidth is available on all the links of its route. For example, while call C_1 cannot be scheduled because the bandwidth is insufficient on link X_2X_3 , call C_2 , which arrives after call C_1 and requests a circuit from host H_3 to H_7 , can be scheduled first. However, this causes a fairness problem because short-path calls stand a better chance of being scheduled than long-path calls.

From the above example, we see that buffered queueing can lead to utilization or fairness problems, especially under high loads. However, bufferless queueing mechanisms can be used in network bandwidth sharing because calls are either accepted immediately or rejected, and hence do not suffer from the utilization and fairness problems caused by queueing. In fact, the immediate-request mode of bandwidth sharing, which is used in the telephone network, and supported by the GMPLS architecture for high-speed connection-oriented networks, is bufferless queueing. Therefore, our hypothesis formulates a study of the immediate-request (queueing) mechanism in conjunction with the book-ahead (reservation) mechanisms for an efficient sharing of bandwidth in modern high-speed connection-oriented networks.

1.2 Motivation and Related Work

Why is it necessary to revisit the question of bandwidth sharing in connection-oriented networks? The most widely deployed connection-oriented network is the global telephone network. It has been in existence for more than a hundred years. It is constructed with DS0-based ($64kb/s$) circuit switches. Bandwidth is clearly shared on this connection-oriented network. Therefore, why should we revisit the question of bandwidth sharing in connection-oriented networks?

Circuit switching dominated the field of networking until the mid-1970s. Experiments with packet switching began in the mid-sixties [8], primarily for computer-to-computer data transfers. A communication network was required to connect large mainframe computers to dumb terminals as well as to interconnect mainframe computers. Packet switching was demonstrated to be superior to circuit switching for this form of communication due to the bursty nature of computer-generated data. It allowed for a more efficient sharing of bandwidth resources. In the 30-year period following

these initial experiments, packet switching became the dominant technology of the Internet. More specifically, the type of packet switching used in the Internet is connectionless packet-switching, whereby there is no bandwidth reservation phase prior to data transfer for a given flow.

However, there has been a renewed interest in connection-oriented networking, with a particular resurgence of interest in circuit-switched technologies. This is because of recent advances in optical communications and optical switching technologies. Due to technological difficulties with optical computing and optical memory, optical packet switching is still in its infancy (computing is needed to process packet headers and memory is needed for the buffers, an essential part of a packet switch). On the other hand, optical circuit switches can be readily built using different technologies [9] [10], and are available from a number of vendors.

In conjunction with the above-described bottom-up technological drive for higher-speed optical circuit switches, a top-down application-driven push for high-speed circuit-switched networks has been created by large-team scientific projects, such as the Terascale Supernova Initiative project [11], and the Large Hadron Collider (LHC) [6] project. These eScience applications require predictable high-speed network services, which are difficult, if not impossible, to provide across the connectionless Internet. These applications include large (terabyte-to-petabyte) file transfers, remote visualization, remote computational steering, and remote instrument control.

As a response to the simultaneous top-down and bottom-up drivers, NSF created the Experimental Infrastructure Network (EIN) program in 2003, and invested \$30 million to create experimental network testbeds. As part of this program, our group was selected to implement our proposed CHEETAH (Circuit-Switched End-to-End Transport Architecture) testbed, which is a high-speed circuit-switched network. More recently, Internet2 has taken the bold step of adding a Dynamic Circuit (DC) network to their backbone network [12] as a complementary service to their connectionless IP backbone network (which is called Abilene). GMPLS-enabled connection-oriented switches, which support both SONET and Ethernet, are deployed in the backbone nodes of Internet2 as shown by the “stars” in Fig. 1.2. This enables Internet2 users to create point-to-point circuits across the Internet2 infrastructure. Other examples of connection-oriented testbeds are Energy Sciences Network 4 (ESnet4) in the United States [13], CANARIE’s CA*net 4 in Canada [14],

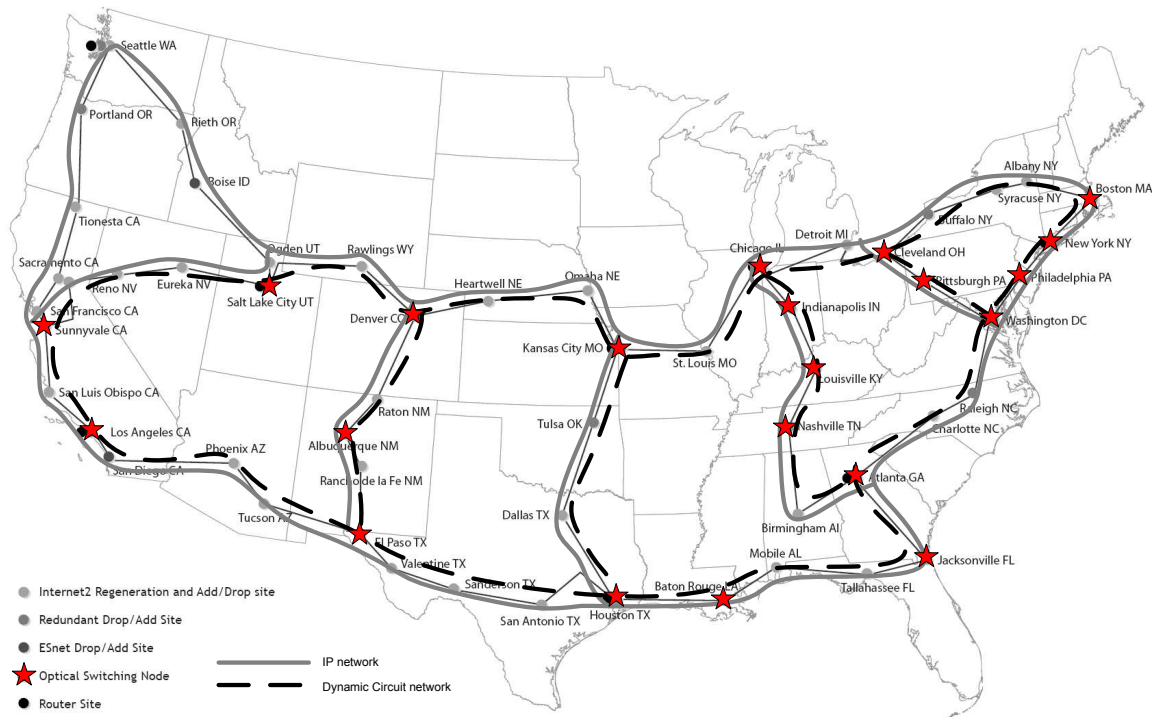


Figure 1.2: Internet2 Dynamic Circuit Network Topology (reprinted from [1]).

UKLight in the United Kingdom [15], SURFnet in Netherlands [16], and JGN2 in Japan [17].

The EIN testbeds, the Internet2 DC network, and the other connection-oriented testbeds, differ from the global telephone network in significant ways. The difference lies not only in the significant difference in data rates, but also in the applications run on the network and channel densities. For example, candidate applications for these new connection-oriented networks require large amounts of per-circuit bandwidth relative to link capacity. In other words, link bandwidth is partitioned into a “small” number of channels. In contrast, in telephone networks, since each voice call requires only 1 DS0 (i.e., 64kbps), and link capacities are on the order of multiple Mbps (e.g., a T3 link has a capacity of 45Mbps), the number of channels into which the link bandwidth is partitioned is well over 100 (e.g., 672 in a T3 link). Another difference is that call-duration distributions for the new applications will most likely differ from those of telephone calls. For example, a scientist, wanting to reserve bandwidth for a remote-visualization session, is likely to ask for a fixed call duration on the order of a few hours. This makes the call duration deterministic in contrast with the random call holding time in the telephone network. These differences (i.e., in channel densities and

requirements) make it necessary to revisit previously defined algorithms for bandwidth-sharing in connection-oriented networks.

Project teams deploying the various experimental connection-oriented testbeds as part of the EIN program, the new Internet2, and other related programs worldwide, are focusing on the implementation aspect of bandwidth-sharing systems. For example, the DOE funded a project called OSCARS [18] to develop a scheduler for the ESnet to accept book-ahead reservations for bandwidth from scientists. While this scheduler is being implemented, there are no analytical or simulation studies to understand how to select parameters, such as the advance-reservation time horizon, per-call circuit rate, etc., for the scheduler aspects that will become increasingly important as usage builds on these networks. Also, there are ongoing debates on whether to support immediate-request or book-ahead bandwidth-sharing modes in the Internet2 DC network. To address these gaps, we focus on the problem of designing and evaluating bandwidth-sharing mechanisms for connection-oriented networks in this context of new optical circuit-switched technologies and new application requirements.

1.3 Outline

This dissertation is organized into 6 chapters. This chapter provides background information, describes the motivation for our work, and summarizes our contributions.

In Chapter 2, we consider two variants of the book-ahead (**BA**) mechanism for *session-type* requests: a BA- n scheme, in which a user specifies n call-initiation time options in the reservation request, and a BA-First scheme, in which a user accepts a reservation with any call-initiation time. BA- n schemes are suitable for applications that require human interactions, as they allow a user to specify a set of acceptable call initiation-time options based on his/her schedule. Other applications of BA- n schemes occur in the real-time systems. The BA-First scheme is suitable for applications that do not require human interactions. For example, if a file-transfer application issues a reservation request for a large file transfer, it could accept any assigned call-initiation time. We present a novel analytical model (discrete-time Markov chain model) for these two types of BA schemes. Using this

analytical model along with a simulation model, we compare BA-n, BA-First, and the immediate-request (IR) schemes. Numerical results show that, if the number of shared channels is small, e.g., 10, book-ahead mechanisms can achieve 95% utilization with a call-blocking probability of only 1%, while with the immediate-request scheme, call blocking probability is 23% even when utilization is only 80%. With our simulation model, we relax certain assumptions about call holding times, per-call bandwidth, and the path-length of calls to study the performance of these bandwidth-sharing mechanisms under more realistic conditions.

We present an improved analytical model for the BA-First scheme for *session-type* requests in Chapter 3. This new model is more scalable but does not apply to the BA-n scheme. We demonstrate the use of the new model as a design tool for systems using the BA-First bandwidth-sharing mechanism. We also show with comparative simulations that our analytical model can be used as a solution for $M/D/m/p$ queueing systems at moderate-to-high loads.

In Chapter 4, we propose a book-ahead bandwidth-sharing mechanism called Varying-Bandwidth Delayed Start (VBDS) for *data-type* requests. Unlike in a session-type reservation request, where holding time and requested bandwidth are specified, a data-type request specifies a file size and maximum acceptable bandwidth. The latter is necessary because a file server could have limitations dictated by its disk access rate (e.g., an IDE disk has a typical write-access rate of 400Mb/s [19]) or bus rate. The network scheduler determines the call holding time based on the file size and assigned bandwidth. A simple fixed-bandwidth scheduler, which we call Fixed-Bandwidth Delayed Start (FBDS), returns a fixed bandwidth allocation with rate equal to the maximum acceptable bandwidth specified in the request for the entire period of the file transfer, while a VBDS scheduler has the flexibility to assign different bandwidth levels for time ranges within the file transfer period based on bandwidth availability. The use of the VBDS mechanism allows connection-oriented networks to overcome a disadvantage relative to packet switched networks. In the latter, ongoing file transfer can take advantage of bandwidth freed up by the completion of other transfers, while this is not feasible in a connection-oriented network operated with a fixed-bandwidth allocation mode. Using simulations, we demonstrate that, with the VBDS scheme, a circuit-switched network can achieve the same level of throughput as a packet-switched network for moderate-to-

large-sized files. For small files, we do not recommend VBDS because the scheduling overhead will be larger than the file transfer time in high-speed networks. Further, we see that VBDS favors large files to moderate-sized files when compared to packet switching.

In Chapter 5, we describe our implementation and experimentations with the immediate-request bandwidth-sharing mode in our deployment of a wide-area connection-oriented network. The experimental network is called CHEETAH. It consists of off-the-shelf GMPLS-capable SONET switches (with Ethernet interfaces) deployed at three locations, Research Triangle Park, North Carolina, Atlanta, Georgia, and Oak Ridge, Tennessee. These switches have built-in GMPLS control-plane software to support the immediate-request mode. We designed and implemented a CHEETAH software package to run on Linux end hosts connected to the CHEETAH network. Among other functions, this software package includes a signaling module to enable end users and applications to dynamically initiate requests for dedicated end-to-end circuits and receive/respond to requests for circuits. We obtained measurements for typical end-to-end circuit setup delays and per-switch signaling processing delays across this network. We expect these measurements to be useful to other researcher in their analytical and simulation studies of connection-oriented networks. Chapter 6 summarizes our contributions, discusses future work, and concludes the dissertation.

1.4 Key Contributions

The key contributions of this work include:

1. We designed two book-ahead mechanisms for session-type requests, and analyzed these mechanisms using discrete-time Markov chain models and simulation models. The models can be used to determine appropriate values for configuration parameters such as the advance-reservation time horizon and the number of call-initiation time options. The more scalable one of our two models provides a tractable solution for the $M/D/m/p$ queueing systems at moderate-to-high loads. This work is published in an *IEEE Transactions on Communications* paper [20], and in a peer-reviewed paper in the *Proceedings of IEEE Global Telecommunications Conference (Globecom)* [21].

2. We designed a book-ahead mechanism for data-type requests that allows circuit-switched networks to overcome a disadvantage relative to packet-switched networks when used for file transfers. This mechanism is called Varying-Bandwidth Delayed Start (VBDS). Using simulations, we compared its performance with packet switching and showed comparable performance for moderate-sized files and better performance for large files.

3. We designed and deployed a wide-area circuit-switched optical network testbed (with three Ethernet-SONET switches and ten end hosts) supporting the immediate-request mode of bandwidth sharing, which includes an implementation of an end-host software package through which applications can request and release high-speed optical circuits dynamically. This work demonstrates the readiness of off-the-shelf switches for actual service offerings, as is now available on Internet2.

4. We obtained measurements of actual end-to-end call setup delays and per-switch processing delays across the deployed wide-area circuit-switched network. The per-switch processing delays can be used to estimate end-to-end call setup delays in large-scale networks, and can be used in analytical and simulation studies of connection-oriented networks. This work is reported in an *IEEE Journal on Selected Areas in Communication* paper [22], and in a peer-reviewed *Proceedings of IEEE International Conference on Communications (ICC)* publication [23].

Chapter 2

Book-ahead Mechanisms for Session-type Requests

We study book-ahead bandwidth-sharing mechanisms for session-type requests in this chapter. With session-type requests, users request a certain number of channels for a fixed amount of time. We present a **discrete-time Markov chain model (DTMC)** for a set of **book-ahead (BA)** bandwidth-sharing mechanisms. We use this analytical model in combination with a simulation model to compare BA mechanisms with the **immediate-request (IR)** mechanism, and to study the performance of different types of BA schemes. Our results show that, with a BA mechanism, lower call blocking probabilities can be achieved than with the IR mode. For example, when $m = 10$, at a system utilization of 80%, the call blocking probability with a basic BA mechanism is 2%, while with the IR mechanism it is 23.6%.

We compare two variants of BA mechanisms, a BA- n scheme, in which a user specifies n acceptable call-initiation time options, and a BA-First scheme, in which a user accepts a reservation with any call-initiation time. We allow for multiple call classes that differ in call durations. While the BA-First scheme achieves the high-utilization, low-call-blocking-probability operating point under an unrestricted call-initiation time policy, it could be limiting to requests that require human interactions. With a restricted call-initiation time policy, in which users can only request starting times that are separated by the minimum call holding time, the BA- n schemes perform almost as well as the BA-First scheme. We show that the advance-reservation horizon, which is the maximum time up to which the scheduler accepts advance reservations, can be fairly small. It needs to be only 4

times the call holding time (in a single-class case) for a 2% call blocking probability when $m = 10$. This factor increases to 14 when m is only 2 channels. We also show that call blocking probabilities of BA schemes increase if the call-initiation time distribution is bell-shaped, which is more realistic than the uniform distribution assumed in the analytical model for tractability. The call blocking probability increases as the deviation of the bell-shaped distribution decreases. In a system that provides users the flexibility to request different levels of bandwidth, we show that there are interesting interactions between the different call classes. Our model provides system designers insights on how many options a BA-n system should support in order to achieve a certain call blocking probability for all classes of calls. We also develop models to study the call blocking probability and fairness ratio of various bandwidth-sharing mechanisms in multi-link scenarios.

The remainder of this chapter is structured as follows: Section 2.1 surveys related work on book-ahead mechanisms. Section 2.2 describes our book-ahead mechanisms. In Section 2.3, we describe our DTMC model for the book-ahead mechanisms. Section 2.4 presents numerical results comparing our analytical and simulation models, comparing different bandwidth-sharing mechanisms (BA-First, BA-n, IR), and determining appropriate values for design parameters, such as the advance-reservation horizon. Section 2.5 presents sensitivity analyses on the call-initiation time distribution and per-call bandwidth. Section 2.6 studies the performance of book-ahead schemes in multi-link scenarios. Section 2.7 concludes the chapter.

2.1 Related Work

There are several research papers on book-ahead bandwidth-sharing (also referred to as “advance reservation”) mechanisms [7, 24–29]. Wolf and Steinmetz [24] outlined a framework for a book-ahead mechanism and discussed design issues. Ferrari, Gupta, and Ventre [25] proposed a distributed book-ahead service. Naik, Siegel, and Chong [7] proposed a class and priority based mechanism to schedule calls in oversubscribed preemptive networks. Two types of calls were considered: i) calls requesting bandwidth for file transfers (data-type requests), and ii) calls requesting a fixed amount of bandwidth for a fixed duration (session-type requests). A heuristic based on

sorting, preemption, and repositioning was proposed. Yuan, Tham, and Ananda [26] investigated book-ahead sharing with flexible bandwidth allocation. To increase utilization and reduce blocking probabilities, they exploited the potential flexibility of calls using a probe-based adaptive reservation approach. Most of these papers use simulations to evaluate their book-ahead mechanisms.

Greenberg, Srikant, and Whitt [27] proposed a scheme for sharing resources among book-ahead calls and immediate-request calls to avoid utilization loss from strict bandwidth partitioning. They assume the arrival and service rates of immediate-request calls are much higher than those of book-ahead calls, and obtain approximate estimates of system performance by decomposing a two-dimensional Markov chain model. Schelén and Pink [30] presented an architecture to allow network resource being shared between immediate and book-ahead calls without being pre-partitioned by scheduling book-ahead calls in batches and allowing preemption of immediate reservations. Coffman, Jelenkovic, and Poonen [28] mathematically analyzed the fraction of time that a resource is booked in a book-ahead system with a Poisson call arrival process. Virtamo [29] derived a closed-form solution for a single-server book-ahead system with deterministic call holding times of one or two timeslots, and an asymptotic analytical solution for a general holding time distribution. Numerical results for multi-server systems were provided by simulations. None of the above mentioned work considers book-ahead calls with multiple acceptable call-initiation time options. Interestingly, our results show that a book-ahead mechanism that only specifies one call-initiation time may perform worse than an immediate-request mechanism when system load is high.

In other areas relating to optimization of advance reservation systems, Bouras and Stamos [31] offered an algorithm to improve network utilization by delaying the admission decision in order to gather a number of requests and compute a better resource allocation. Their batch processing approach allows for better optimization, but suffers from a cost of speed and complexity. Given the eScience community's interest in multi-domain circuit/VC scheduling, we design our system for simplicity, wherein arriving calls are either immediately assigned an interval or rejected. In [32], Guérin and Orda examined the impact of advance reservations on the computational complexity of path selection. They focused primarily on the issue of computational complexity, while we are more interested in the call blocking probability and network utilization. Burchard [33] introduced

the idea of malleable reservations (reservations whose start times and bandwidth assignments can be adjusted based on the bandwidth availability) to overcome the fragmentation problem of book-ahead schemes. This idea applies only to data-type requests, while we consider both data-type and session-type requests in our research.

2.2 Book-ahead Mechanism

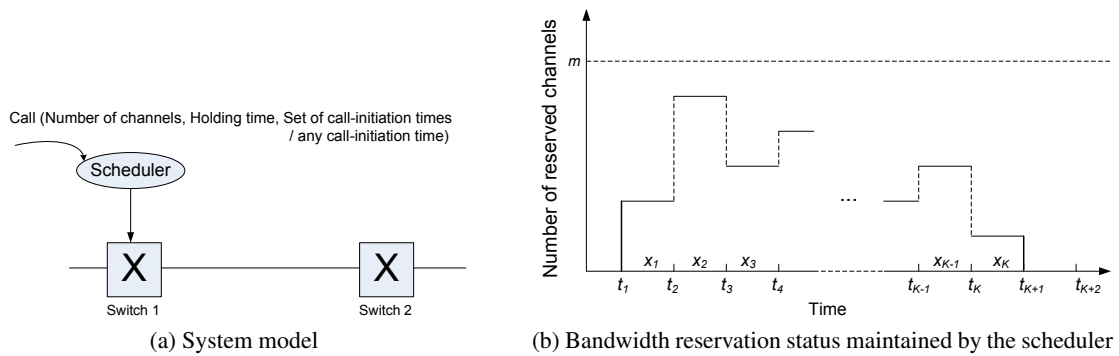


Figure 2.1: Book-ahead mechanism in the single-link scenario.

Our book-ahead mechanism works as follows in a single-link scenario. The link capacity C is discretized and expressed as m channels where the bandwidth of each channel is C/m . Time is also discretized for simpler management. Fig. 2.1a illustrates our system model with the scheduler managing the bandwidth of the link between switch 1 and switch 2. The scheduler maintains a database of reserved bandwidth as a function of time as illustrated in Fig. 2.1b. The maximum time horizon up to which the scheduler will accept reservation requests, as shown in Fig. 2.1b, is K . A call arriving at time t_1 can request one or more channels for any contiguous set of timeslots between t_2 and t_{K+2} .

As shown in Fig. 2.1a, a user requests bandwidth as a discrete number of channels and specifies call holding time as a discrete number of timeslots. A user also provides a preference-ordered acceptable set of call-initiation times (up to n options), or sets the indicator that any call-initiation time is acceptable. A BA-First call, which accepts any initiation time, is scheduled as early as possible, while for a BA- n call, which specifies n acceptable initiation times, the first admissible option

is scheduled from the user's preference-ordered list. A BA- n call arriving at time t_1 , requesting a channel for H timeslots, will be blocked if there is no channel available for H contiguous timeslots starting from any of the n call-initiation time options. A BA-First call will be blocked if a channel is not available for H contiguous timeslots in the entire K -timeslot horizon.

2.3 Analytical Model

We use a Discrete-time Markov Chain (DTMC) to model the book-ahead mechanism. We state our assumptions, derive the transition probability matrix, and provide an example.

2.3.1 Assumptions

We make the following assumptions:

1. The call-arrival process is Poisson with an aggregate rate λ . Measurements have shown that user-initiated TCP session arrivals on the Internet can be modeled with a Poisson process [34], and telephony traffic has long been modeled as a Poisson process [8]. Assuming that the new applications for which the book-ahead bandwidth sharing mechanism is being designed will follow similar trends, we make this assumption.
2. Further, we approximate the exponential distribution of call interarrival times with a geometric distribution. The smallest value of call interarrival time is thus one timeslot. The parameter of the geometric distribution, p , represents the probability that a call arrives within a timeslot. In other words, the only two options for the number of call arrivals in each timeslot is 0 or 1. Conditions under which this approximation is valid will be discussed in a later section that describes our selection of numerical values for the parameters of this model. We will see that, to obtain an accurate approximation, the duration of the timeslot needs to be very small.
3. There are l classes of calls. The arrival rate of class- i calls is λr_i . Each call specifies the required number of channels, the call holding time, and its preference regarding call-initiation

times. In our model, we assume that calls of all classes request only one channel but differ in their holding-time requests. The holding time of a class- i call is h_i timeslots.

4. In the BA- n scheme, the n call-initiation time options specified by the user in the call request (see Fig. 2.1a), denoted by s_1, s_2, \dots, s_n , are assumed to be uniformly distributed among the $(K - h_i + 1)$ eligible options for a class- i call. Each of the n options is assumed to be unique.
5. The time for reservation processing by the scheduler is considered negligible in the model, which allows us to have the system change states instantaneously on timeslot boundaries.

2.3.2 State transition probability matrix

We denote the state of the system by vector x , which consists of K components (x_1, x_2, \dots, x_K) , where x_i represents the number of channels that have been reserved for the i^{th} timeslot. For example, x_1 indicates the number of channels that have been reserved for the current timeslot (the first timeslot).

The state space S is therefore defined as

$$S \triangleq \{x = (x_1, x_2, \dots, x_K) : 0 \leq x_i \leq m \text{ for } i = 1, 2, \dots, K\}.$$

The size of the state space

$$N = (m + 1)^K. \quad (2.1)$$

Calls arrive on timeslot boundaries and since we neglect processing time, state transitions occur instantaneously at each timeslot boundary. If, at time t_{1+} (just after time instant t_1) the system state $S_{t_{1+}}$ is (x_1, x_2, \dots, x_K) as shown in Fig. 2.1b, the system will be in the same state at t_{2-} (just before time instant t_2). If no call arrives at time instant t_2 or the arriving call is blocked, the system will transition to state $(x_2, x_3, \dots, x_K, 0)$ at time t_2 . This is because x_2 , which is the number of channels reserved for the (t_2, t_3) timeslot when the system is in state S_{t_1} , becomes the first component of the state vector at time t_2 . Similarly, x_3 , the number of channels reserved for the (t_3, t_4) timeslot, becomes the second component of the system state S_{t_2} . In other words, all components of the state vector shift to the left by one at each timeslot boundary.

If a call arrives at time instant t_2 and successfully reserves some timeslots, the system state will change accordingly in addition to the above described left-shift of the state-vector components. For example, if the system is in state $(0, 1, 0)$ at time t_{1+} and a call arrives at t_2 and reserves a channel with a call-initiation time of t_3 and a holding time of two timeslots, then the system will transition to state $(1, 1, 1)$ at time t_2 . Based on these observations, we can see that the state transition probabilities from one state to another depend only on the current state, and are independent of the system history. Therefore this model is a discrete-time Markov chain. The model is also time-homogeneous because the state transition probabilities are independent of the time.

Before we derive the transition matrix of this Markov chain, for ease of description, we define a left shift operator “ \leftarrow ” on vector x as follows. If $x = (x_1, x_2, \dots, x_K)$, $\overleftarrow{x} = (x_2, x_3, \dots, x_K, 0)$. We also define a K -component vector $O_{i,j} = (O_1, O_2, \dots, O_K)$, where

$$O_x \triangleq \begin{cases} 1 & \text{if } i \leq x \leq j, \\ 0 & \text{otherwise.} \end{cases}$$

The transition probability from state x to state y , denoted by p_{xy} , is

$$p_{xy} = \begin{cases} 1 - p + pB_x & \text{if } y = \overleftarrow{x}, \\ pr_j q_{i,j} & \text{if } y = \overleftarrow{x} + O_{i-1, i+h_j-2}, \text{ and } y \in S, \\ 0 & \text{otherwise.} \end{cases} \quad (2.2)$$

where p denotes the probability that a call arrives in a timeslot, r_j denotes the probability that the arriving call belongs to class- j , $q_{i,j}$ denotes the probability that a class- j call is admitted with a call-initiation time of the i^{th} timeslot relative to its arrival instant, and B_x denotes the probability that an incoming call is blocked when the system is in state x .

The first row of (2.2) represents the case when no call arrives (probability of $(1 - p)$) and the case when a call arrives (probability of p) but is blocked (probability of B_x). Therefore the state vector only shifts to the left with no new reservations. The second row represents the case that the call is admitted with a call-initiation time of the i^{th} timeslot away from the call arrival instant. The last row shows that the transition probabilities to all other states are 0.

To calculate $q_{i,j}$, recalling that any of the n call-initiation time options, s_1, s_2, \dots, s_n , could be the one that requested the t^{th} timeslot, we first determine the probability that the t^{th} option was the one that requested the t^{th} timeslot, i.e., $s_t = i$. Before we can compute this probability, we define d_j to be the number of timeslots that are “ineligible,” i.e., that the starting instants of these timeslots, if specified as one of the call-initiation options, would lead to a blocked call because of a lack of resources in one or more of the following h_j timeslots. We can determine d_j for any given state x by examining the state vector and finding all those elements for which at least one or more of the following h_j timeslots are fully booked.

Successfully admitting the call with its channel reservation starting from the t^{th} timeslot implies that the t^{th} timeslot is not one of the ineligible d_j timeslots. Furthermore, if this timeslot was requested as the t^{th} call-initiation time option, i.e., $s_t = i$, it means that the previous $(t - 1)$ options were all denied. The implication of this statement is that the first $(t - 1)$ options all belong to the set of d_j ineligible timeslots. Finally since we require the n call-initiation time options to be distinct, while the first option could be any one of the d_j ineligible timeslots, the second option should necessarily be selected from the remaining $(d_j - 1)$ ineligible timeslots. We can apply reasoning similar to that needed in determining the hypergeometric probability mass function, $h(k; n, d, N)$, which is the probability of having k defective units in a random batch of n elements, drawn *without replacement* from a larger set of N elements, which is known to contain d defective elements [35]. This probability is:

$$h(k; n, d, N) = \frac{\binom{d}{k} \binom{N-d}{n-k}}{\binom{N}{n}}, \text{ for } k = 0, 1, \dots, \min\{d, n\}, n - k \leq N - d. \quad (2.3)$$

Our current problem is similar in the sense that once a timeslot is chosen for a particular call-initiation time option, it cannot be selected for a subsequent option. There are totally $(K + 1 - h_j)$ candidate timeslots (analogous to elements in the hypergeometric definition), which is known to contain d_j ineligible timeslots (defective elements). Using similar reasoning, we determine the probability $R_{t,i,j}$ that the first acceptable call-initiation time in a class- j call is its t^{th} option and that

$s_t = i$ as:

$$R_{t,i,j} = P(s_1, s_2, \dots, s_{t-1} \text{ are ineligible}) \cdot P(s_t \text{ is eligible} \mid s_1, s_2, \dots, s_{t-1} \text{ are ineligible}) \\ \cdot P(s_t = i \mid s_1, s_2, \dots, s_{t-1} \text{ are ineligible and } s_t \text{ is eligible}). \quad (2.4)$$

The first factor is analogous to the probability that all elements in a random batch of $(t - 1)$ elements are defective. Therefore, by applying (2.3) we have

$$P(s_1, s_2, \dots, s_{t-1} \text{ are ineligible}) = \begin{cases} h(t-1; t-1, d_j, K+1-h_j) = \frac{\binom{d_j}{t-1} \binom{K+1-h_j-d_j}{t-1-(t-1)}}{\binom{K+1-h_j}{t-1}} & t-1 \leq d_j, \\ 0 & \text{otherwise.} \end{cases}$$

Reducing the above,

$$P(s_1, s_2, \dots, s_{t-1} \text{ are ineligible}) = \begin{cases} \prod_{y=0}^{t-2} \frac{d_j-y}{K+1-h_j-y} & t-1 \leq d_j, \\ 0 & \text{otherwise.} \end{cases} \quad (2.5)$$

Consider the second factor of (2.4). For the t^{th} option, given the *without-replacement* selection procedure, only $(K + 1 - h_j - (t - 1))$ timeslots remain but the number of eligible timeslots are still $(K - h_j - d_j + 1)$ because all timeslots selected in the first $(t - 1)$ options are ineligible timeslots. Therefore:

$$P(s_t \text{ is eligible} \mid s_1, s_2, \dots, s_{t-1} \text{ are ineligible}) = \frac{\binom{K-h_j-d_j+1}{1}}{\binom{K+1-h_j-(t-1)}{1}} = \frac{K-h_j-d_j+1}{K-h_j-t+2}. \quad (2.6)$$

Similarly we can obtain the third factor of (2.4):

$$P(s_t = i \mid s_1, s_2, \dots, s_{t-1} \text{ are ineligible and } s_t \text{ is eligible}) = \frac{1}{K-h_j-d_j+1}. \quad (2.7)$$

Combining (2.5), (2.6) and (2.7) with (2.4), we get:

$$R_{t,i,j} = \begin{cases} \frac{1}{K-h_j-t+2} \prod_{y=0}^{t-2} \frac{d_j-y}{K+1-h_j-y} & t-1 \leq d_j, \\ 0 & \text{otherwise.} \end{cases} \quad (2.8)$$

Since any of the n call-initiation time options could be the selected t^{th} option and $q_{i,j}$ is the probability that a class- j call is admitted with a call-initiation time of the i^{th} timeslot relative to its arrival instant,

$$q_{i,j} = \begin{cases} \sum_{t=1}^n R_{t,i,j} & \text{if the } i^{\text{th}} \text{ timeslot is an eligible timeslot,} \\ 0 & \text{otherwise.} \end{cases} \quad (2.9)$$

Having computed $q_{i,j}$, we now have the state transition probabilities for the second row of (2.2). We next turn our attention to the blocking probability B_x in state x in order to compute the state transition probabilities corresponding to the first row of (2.2). Before we calculate B_x , we compute the blocking probability for a class- j call, denoted $B_{x,j}$. Recalling that a call is blocked if and only if resources are unavailable at all the n options specified by the user, $B_{x,j}$ is analogous to the probability that all elements in a random batch of n elements are defective in the hypergeometric example. Therefore $B_{x,j}$ can be calculated by applying (2.3) as follows:

$$B_{x,j} = \begin{cases} h(n;n,d_j,K+1-h_j) = \prod_{y=0}^{n-1} \frac{d_j-y}{K+1-h_j-y} & \text{if } n \leq d_j, \\ 0 & \text{otherwise.} \end{cases} \quad (2.10)$$

We find the aggregate call blocking probability B_x using a weighted average

$$B_x = \sum_{j=1}^l (r_j B_{x,j}), \quad (2.11)$$

where the weight r_j is the probability that the arriving call is a class- j call. Having computed $q_{i,j}$ and B_x , we now have the DTMC transition probability matrix specified in (2.2).

2.3.3 Example

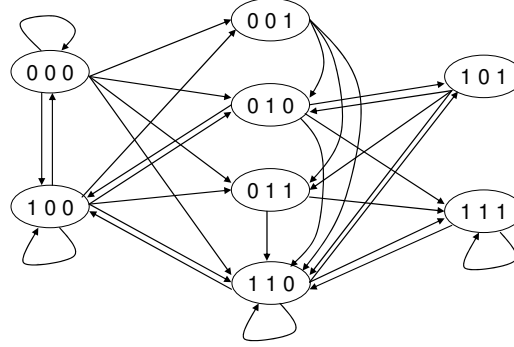


Figure 2.2: DTMC for an example system: $m = 1$, $K = 3$, $l = 2$, $h_1 = 1$, $h_2 = 2$, and $n = 1$.

We derive the transition probability matrix for the following example. Let m , the link capacity in channels, be 1, and K , the advance-reservation horizon, be 3 timeslots. We allow two call classes ($l = 2$). The holding time for class-1 calls (h_1) is 1 and for class-2 calls (h_2) is 2. The number of call-initiation time options allowed, n , is 1. Fig. 2.2 shows the transition diagram for this example. We explain in detail how we apply (2.2, 2.8, 2.9, 2.11) to determine the transition probabilities from state $(0, 0, 1)$.

Assume the system is in state $(0,0,1)$ at time t_1 . If no call arrives at time t_1 or a call arrives but gets blocked, the system will transition to state $(0,1,0)$. To calculate $p_{(0,0,1),(0,1,0)}$, the transition probability from state $(0,0,1)$ to state $(0,1,0)$, we first find the number of ineligible timeslots in state $(0,1,0)$ for the two call classes, $d_1 = 1$ and $d_2 = 2$. Using $K = 3$, $n = 1$, $l = 2$, $h_1 = 1$, $h_2 = 2$ in (2.10) we obtain

$$B_{(0,0,1),1} = \frac{d_1 - 0}{K + 1 - h_1 - 0} = \frac{1}{3}$$

and

$$B_{(0,0,1),2} = \frac{d_2 - 0}{K + 1 - h_2 - 0} = 1.$$

Applying (2.11) we get

$$B_{(0,0,1)} = r_1 B_{(0,0,1),1} + r_2 B_{(0,0,1),2} = \frac{1}{3} r_1 + r_2 = 1 - \frac{2}{3} r_1.$$

Now we compute the probability $p_{(0,0,1),(0,1,0)}$ by applying the first row of (2.2)

$$p_{(0,0,1),(0,1,0)} = 1 - p + pB_{(0,0,1)} = 1 - p + p\left(1 - \frac{2}{3}r_1\right) = 1 - \frac{2}{3}pr_1.$$

If a class-1 call arrives at time t_1 and requests a call-initiation time of t_2 , the system will transition to state $(1, 1, 0)$. To calculate the probability of this transition, we first calculate $q_{2,1}$ by applying (2.9)

$$q_{2,1} = R_{1,2,1} = \frac{1}{K - h_1 - t + 2} = \frac{1}{3 - 1 - 1 + 2} = \frac{1}{3}.$$

Then we can calculate $p_{(0,0,1),(1,1,0)}$ by applying the second row of (2.2)

$$p_{(0,0,1),(1,1,0)} = pr_1q_{2,1} = \frac{1}{3}pr_1.$$

If a class-1 call requests a call-initiation time of t_4 , it will transition to $(0,1,1)$ for which we can similarly compute the transition probability to be $\frac{1}{3}pr_1$. If the arriving call is a class-2 call which in this example means it requires two contiguous timeslots ($h_2 = 2$), all three timeslots are ineligible. This is because the channel is reserved for the (t_3, t_4) timeslot. If the call-initiation time requested is t_2 , then only one timeslot (t_2, t_3) is available. If the requested call-initiation time is t_3 , then again the call is blocked because timeslot (t_3, t_4) is reserved in state $(0,0,1)$. Finally a class-2 call cannot request t_4 since the advance-reservation horizon K is three, which does not permit the required two timeslots if t_4 is the call-initiation time.

The remaining entries in the state transition matrix for this example, shown in Table 2.1, can be derived in a similar manner. As can be observed from the table, the transition probabilities are independent of the reservation status in the first timeslot (the first component of vector x) because the information in this timeslot is irrelevant when the state shifts to the left by one at the end of the timeslot. Thus rows in the table that differ only in the first component, e.g., $(0,0,1)$ and $(1,0,1)$, have identical entries in all the columns.

2.3.4 Performance metrics

Session-type requests, especially those requests that provide n call-initiation time options, are typical for interactive applications. For such applications, delay is important only after data flow commences. An example is the telephony application. Users of the telephone network have a stringent delay requirement (a one-way delay of $150ms$ in a path with echo cancellers), which needs to be guaranteed only after a call is accepted and data flow starts. Such a delay requirement is easily met in a circuit-switched network because, once a call is admitted, it has guaranteed bandwidth dedicated for its exclusive use. Thus, instead of delay, we use call blocking probability, which is a more effective measure of book-ahead bandwidth-sharing schemes. Also, noting that call blocking probability can be traded off against network utilization, we use the following two metrics, P_B , the long-run blocking probability for calls, and U , the long-run utilization of the system, to characterize the performance of book-ahead sharing mechanisms.

To calculate these two metrics, we first calculate the steady-state probabilities, denoted by vector π , of this discrete-time Markov chain using the well-known formula [8]

$$\pi = e(I + E - P)^{-1} \quad (2.12)$$

where P denotes the transition matrix, I denotes the $(N \times N)$ matrix having ones along the main

Table 2.1: Transition matrix for the example Markov chain shown in Fig. 2.2.

	(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)
(0, 0, 0)	$1 - p$	$\frac{1}{3}pr_1$	$\frac{1}{3}pr_1$	$\frac{1}{2}pr_2$	$\frac{1}{3}pr_1$	0	$\frac{1}{2}pr_2$	0
(0, 0, 1)	0	0	$1 - \frac{2}{3}pr_1$	$\frac{1}{3}pr_1$	0	0	$\frac{1}{3}pr_1$	0
(0, 1, 0)	0	0	0	0	$1 - \frac{2}{3}pr_1 - \frac{1}{2}pr_2$	$\frac{1}{3}pr_1$	$\frac{1}{3}pr_1$	$\frac{1}{2}pr_2$
(0, 1, 1)	0	0	0	0	0	0	$1 - \frac{1}{3}pr_1$	$\frac{1}{3}pr_1$
(1, 0, 0)	$1 - p$	$\frac{1}{3}pr_1$	$\frac{1}{3}pr_1$	$\frac{1}{2}pr_2$	$\frac{1}{3}pr_1$	0	$\frac{1}{2}pr_2$	0
(1, 0, 1)	0	0	$1 - \frac{2}{3}pr_1$	$\frac{1}{3}pr_1$	0	0	$\frac{1}{3}pr_1$	0
(1, 1, 0)	0	0	0	0	$1 - \frac{2}{3}pr_1 - \frac{1}{2}pr_2$	$\frac{1}{3}pr_1$	$\frac{1}{3}pr_1$	$\frac{1}{2}pr_2$
(1, 1, 1)	0	0	0	0	0	0	$1 - \frac{1}{3}pr_1$	$\frac{1}{3}pr_1$

diagonal and zeros elsewhere, and E and e are, respectively, the $(N \times N)$ matrix and the $(1 \times N)$ row vector in which all elements equal one. After obtaining the steady-state probability vector π , we can calculate the call blocking probability and system utilization.

2.3.4.1 Call blocking probability

We denote the call blocking probability in each state of the model by vector B , where elements B_x are calculated using (2.11). The long-run average blocking probability P_B can then be calculated as

$$P_B = \pi B^T. \quad (2.13)$$

2.3.4.2 Utilization

Similarly, we denote the utilization of the system in state $x = (x_1, x_2, \dots, x_K)$ by u_x , which can be calculated as

$$u_x = \frac{\sum_{i=1}^K x_i}{K \cdot m}. \quad (2.14)$$

Then the average utilization U can be calculated by

$$U = \pi u^T, \quad (2.15)$$

where the components of the vector u are the utilization in each state u_x .

2.4 Numerical Results

We start with a discussion on how we select numerical values for significant input parameters. Among these are the call arrival probability (p), the link capacity in channels (m), average call holding time ($E[H]$), and the advance-reservation horizon (K). **Section 2.4.1** describes how we select numerical values for these parameters.

Due to the state-space explosion problem (see (2.1)), the DTMC can only be solved for small values of m . Hence we implement a simulation model, and compare numerical results obtained

from the two models for the $m = 1$ case. The simulation model is an event-driven system written in C++. Simulations for each variant of the book-ahead (BA) mechanism and the immediate-request (IR) mechanism are run for sufficiently long durations and repeated multiple times to obtain a large number of samples. **Section 2.4.2** describes how we verify and validate these two models, and shows that the results obtained from the two models for the $m = 1$ case match closely.

For larger values of m , we obtain numerical results using the simulation model. **Section 2.4.3** presents these results. We simulated the system for different traffic loads consisting of a single call class as well as multiple call classes. Our principal finding was that while the BA-First bandwidth-sharing mechanism lowers call blocking probability significantly relative to the IR scheme for small m , the performance gains obtained with the BA- n schemes decreases as load increases. The reason for this decrease is as follows. Since the n call-initiation time options specified in a BA- n scheme are unrestricted and assumed to be uniformly distributed across the whole advance-reservation horizon, the requested reservation periods for a call could overlap with the multi-timeslot ranges of already-accepted reservations, in which case, the call is blocked.

In **Section 2.4.4**, we restrict call-initiation time options to timeslot boundaries separated by the minimum call holding time (amongst the l call classes). Further, we assume that the call holding time of each class is restricted to be an integer multiple of the minimum call holding time. Under this restricted call-initiation times policy, we found that even BA- n schemes achieve the expected lowering of call blocking probability for small m .

2.4.1 Selection of numerical values for the model parameters

In Section 2.3 we stated our assumption that call interarrival times are exponentially distributed. Further we noted that we use a discrete random variable with the geometric distribution to approximate call interarrival times. As noted in [36], if a random variable X is exponentially distributed with parameter λ , then $Y = \lceil X \rceil$ is a geometric random variable with **parameter** $p = 1 - e^{-\lambda}$. Approximating X by $\lceil X \rceil$ is clearly inaccurate if the probability that $X < 1$ is very large. For example, if λ is 2 calls/second, then $P(X \leq 1) = 0.8647$. If $X \leq 1$, Y is set to 1, which makes the approximation inaccurate.

Consider the mean values of these two random variables, $1/\lambda$ and $1/p$; we can theoretically calculate the maximum value of λ for any given desired maximum deviation of the corresponding p from λ . If we want the deviation to be less than a given value σ , which means

$$\frac{|p - \lambda|}{\lambda} = \left| \frac{p}{\lambda} - 1 \right| = \left| \frac{1 - e^{-\lambda}}{\lambda} - 1 \right| < \sigma, \quad (2.16)$$

we can calculate the maximum value of λ that satisfies the above inequality for different values of σ . For example, if we want the deviation to be less than 5%, λ should be less than 0.104. If we want σ to be less than 2%, λ should be less than 0.041.

Any λ can be downscaled to a smaller value that meets the above requirement by changing the time unit, e.g., 100 calls/sec can be represented as 0.1 calls/ms. By downscaling we essentially divide time into small enough timeslots so that the geometric distribution is a good approximation of the exponential distribution.

The next parameter we consider is m , the **link capacity** in channels. The number of the states in the discrete-time Markov chain increases exponentially with m (see (2.1)). The size of the transition matrix quickly exceeds the maximum matrix size that our programming environment allows. Therefore, we limit m to 1 for the numerical results generated from our DTMC model. For simulations, we choose values of 2, 5, and 10 for m . As described in chapter 1, scientific applications require large per-circuit bandwidth, which results in this range of m .

When m increases (say from 1 to 10), the **holding time** H also needs to be increased to study the system under a high load because the system load $\rho/m = \frac{\lambda E[H]}{m} = \frac{pE[H]}{m}$, where $E[H]$ is the mean call holding time. For example, if $m = 2$, in order to create a high load ($\rho/m = 0.99$) when $p = 0.09$, $\max_{j=1,2,\dots,l} (h_j)$ must be at least 22. Therefore, for the analytical results (when $m = 1$) we chose holding times on the order of 10, but increase the values to be on the order of 100 when $m = 10$.

The final parameter, the **advance-reservation horizon** K , depends upon the largest call holding time and the number of call-initiation time options, n . In the above example, when $\max_{j=1,2,\dots,l} (h_j) = 22$, K must be at least 24 if $n = 3$ since call-initiation options should be distinct. The number of states in this example will be $(2 + 1)^{24}$ when $m = 2$ (see (2.1)), which makes the DTMC model difficult to

solve on a 32-bit computer. For the analytical results, we choose a maximum call holding time of 9 and n values of 1 and 3, which makes it sufficient to choose a K value of 11. For simulation results when holding time is on the order of 100 timeslots, we choose K to be on the order of 1000. This is because under the restricted call-initiation time policy, K needs to be $(n + 1)$ times $\max_{j=1,2,\dots,l} (h_j)$.

2.4.2 Model verification and validation

Model verification and validation are essential parts of the model development process. In [37], Raj Jain notes that

“Model validation consists of validating three key aspects of the model:

- 1. Assumptions*
- 2. Input parameter values and distributions*
- 3. Output values and conclusions”*

Jain further lists three potential validation techniques that can be used to validate each of the above aspects. These techniques are

- “1. Expert intuition*
- 2. Real system measurements*
- 3. Theoretical results” [37]*

Jain also points out that different validation methods are appropriate for models of a system in different stages of its life-cycle. Our goal for creating these models is to develop algorithms and associated parameters for an *initial* design and implementation of a BA system. As BA mechanisms have not yet been deployed, there are no “real system measurements” to validate the output values and conclusions. For such situations, Pace and Sheehan recommend, “Qualitative validation has to be used when adequate acceptable real world data do not exist to permit quantitative validation and is based mainly upon SME (Subject Matter Expert) and peer review.” [38] Therefore, we have relied on the intuition of experts and the peer review process to validate all three aspects of the models. This work is accepted for publication in *IEEE Transactions on Communications* [20].

Some applications being targeted by these BA sharing mechanisms, such as remote visualization, are brand new. They are just not supported today. Other applications, such as video conferencing, use leased lines. However, given the high cost of leased lines, we expect the pattern of call arrivals for video conferencing to be radically different than for leased lines when this application is supported with dynamic BA bandwidth-sharing mechanisms. Therefore, we have no real system measurements for predicting input parameter distributions, or validating assumptions. However, based on real system measurements of comparable applications, such as telephony [39], we assume that call arrival processes for these applications will also be Poisson.

A verification of the two models is performed by comparing the outputs of the two models for the same set of input values. As noted in [40], using alternative representations of the BA mechanism, i.e., our DTMC analytical model and a simulation model, we were able to uncover modeling errors and incorrect implementations of assumptions. Having obtained a close match of results from both models for the $m = 1$ case as shown in Fig. 2.3, we now have confidence in the models.

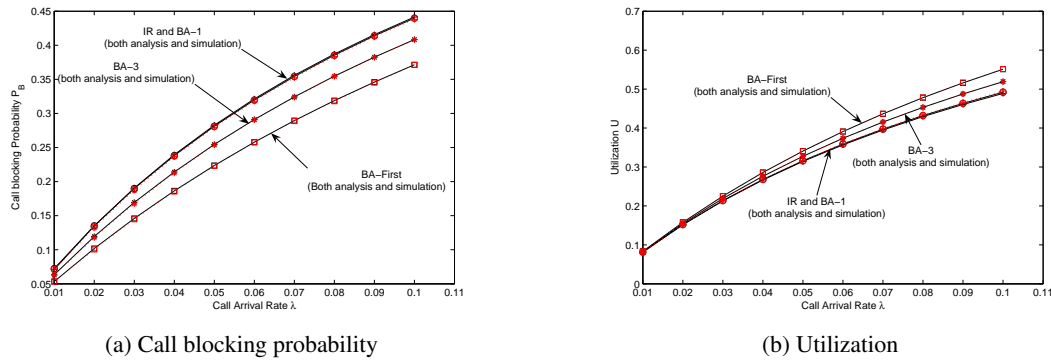


Figure 2.3: Performance of different bandwidth sharing mechanisms. $m = 1$, $l = 2$, $h_1 = 7$, $h_2 = 9$, $r_1 = 0.1$, $r_2 = 0.9$, and $K = 11$.

We choose two classes of calls for the $m = 1$ case shown in Fig. 2.3. The holding time of class-1 calls, h_1 , is 7 timeslots and the arrival rate factor $r_1 = 0.1$, while the holding time of class-2 calls, h_2 , is 9 timeslots and the arrival rate factor $r_2 = 0.9$. The advance-reservation horizon K is assumed to be 11 timeslots. Fig. 2.3 plots the results from four bandwidth sharing mechanisms: immediate-request (IR), book-ahead with 1 option (BA-1), book-ahead with 3 options (BA-3), and

book-ahead that accepts any timeslot (BA-First). 95% confidence intervals were calculated for the simulation results but were not included in the plots because the intervals were too small relative to the mean values. The plots show an excellent match of results from the analytical and simulation models.

Also, Fig. 2.3 shows that BA-1 results in a call blocking probability that is similar to the IR scheme, while BA-3 and BA-First result in lower call blocking probabilities and higher utilizations. Utilization, as seen in the Erlang-B formula [41], will reach a maximum value of 0.495 since m is only 1. The BA schemes improve this value marginally.

The performance gains of the BA schemes relative to the IR schemes for this set of input parameters is not significant. This is because m is only 1 here and K is small relative to h_2 . The performance gains will be more clear in the next section when we increase m , and correspondingly increase K .

2.4.3 Simulation results for larger values of m

2.4.3.1 Single call class

In this subsection, we assume all calls belong to one class, which means $l = 1$. We fix the call holding time but vary call arrival rates to study the performance of different bandwidth-sharing mechanisms under increasing load. The link capacity, m , is set to 10 channels. First, we set call holding time H (we use H instead of h_1 to denote the call holding time of class-1 calls because there is only one call class) to 300 timeslots and change the call arrival probability p from 0.008 to 0.04. A call arrival probability of 0.008 corresponds to an offered system load of 24%, and a call arrival probability of 0.04 corresponds to an offered system load of 120%. The offered load can be higher than 100% because calls can be blocked. The advance-reservation horizon K is fixed at 2000 timeslots.

The call blocking probability and system utilization under different sharing mechanisms are plotted with dashed lines in Fig. 2.4. [Fig. 2.4 also shows results for the restricted variants; these plots will be discussed in a later section.] Since the call blocking probability values of different mechanisms are too close to distinguish when the offered system load is low, in addition to the

linear-scale plot in Fig. 2.4a, we plot the call blocking probability using a base 10 logarithmic scale in Fig. 2.4b. We see that the book-ahead mechanism in which calls accept any available timeslot (BA-First) performs much better than the immediate-request mechanism. A 95% utilization is achievable at a call blocking probability of 2% when the BA-First scheme is used, while that high level of utilization is not even achievable with the IR scheme. A 100% utilization with BA-First is achievable, at which point, the call blocking probability is 6%.

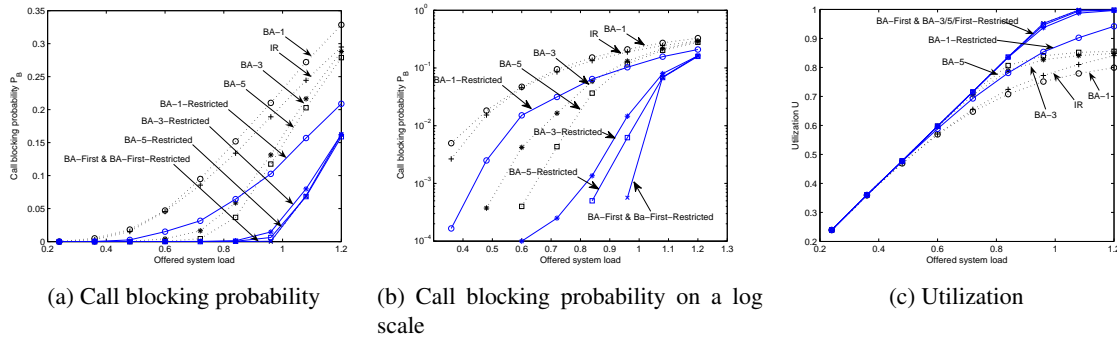


Figure 2.4: Performance of different bandwidth sharing mechanisms. $m = 10$, $l = 1$, $H = 300$, and $K = 2000$.

The BA- n mechanisms plotted with dashed lines in Fig. 2.4, BA-3 and BA-5, also clearly outperform IR when the system load is high but not overloaded (between 50% and 100%). In other words, we can achieve high utilization while keeping call blocking probability low even when applications require a high per-channel capacity relative to link capacity.

However, if only one call-initiation time option (BA-1) is provided, then the call blocking probability and utilization are worse than with the IR mechanism. This is caused by our assumption about call-initiation times. Since we assume that call-initiation times are uniformly selected from amongst $(K - h_j + 1)$ timeslots, there is a high probability that the reservation period requested, starting from the single call-initiation time option specified, overlaps with the holding time (which is multiple timeslots, specifically 300 in Fig. 2.4) of an already-admitted call. This probability increases with system load. This is also the reason why the performance gains of the BA- n schemes ($n > 1$) relative to the IR scheme decrease as offered load increases above 95%. While our assumption with regards to call-initiation time options was required for the DTMC model, these results

demonstrate the need to limit users to certain call-initiation time options based on call holding times. Without such constraints, “gaps” could form in the reservation timeline. A “gap” is a time period smaller than the call holding time in which the system is not fully utilized. Gaps are caused by book-ahead calls requesting arbitrary call-initiation times, independent of the call holding time. For example, Fig. 2.5 shows a gap in the reservation status of a 3-channel single-class system, in which the call holding time, H , is 4 timeslots. In this particular instance, the scheduler has accepted call-initiation time requests for two new calls at t_6 , while the last instant at which the system was fully reserved was the end of timeslot (t_2, t_3) . Thus, in the three-timeslot range, (t_3, t_4) , (t_4, t_5) , and (t_5, t_6) , the link occupancy is less than 3 channels. This becomes a “gap” since it is not long enough to admit a new 4-timeslot call. We consider a restricted call-initiation time policy, which would prevent such advance reservations, in the next section.

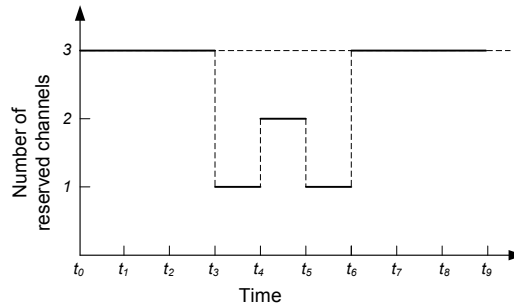


Figure 2.5: Gaps in the advance-reservation horizon when call-initiation time is unrestricted.

To study the impact of call holding time, we changed the call holding time H to 100 timeslots. We also changed the call arrival probability to range between 0.024 and 0.12 so that the same offered system load range is maintained (24% and 120%). We do not observe any significant differences relative to the plots shown in Fig. 2.4. The relative difference between the various BA schemes and the IR scheme stays the same.

2.4.3.2 Multiple call classes

Assuming two call classes, we set the call holding times, h_1 and h_2 , to 100 and 300 timeslots, respectively, and set the probabilities that an incoming call belongs to class-1 and class-2, i.e., r_1 and r_2 , to 0.3 and 0.7, respectively. Therefore the average call holding time is $h_1 r_1 + h_2 r_2 = 240$

timeslots. We change p from 0.01 to 0.05 to simulate the system under the same offered load range (24% to 120%) and plot the call blocking probability against offered system load in Fig. 2.6a.

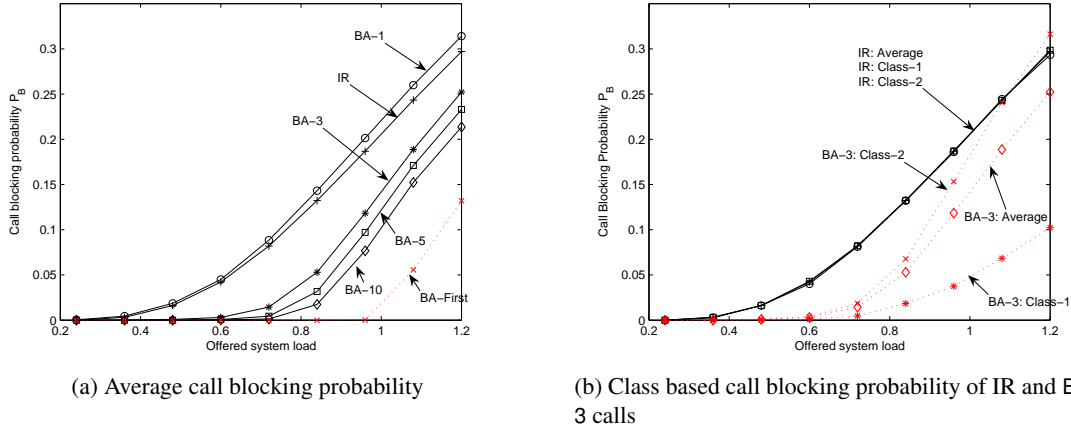


Figure 2.6: Performance of different bandwidth sharing mechanisms with two classes of calls. $m = 10$, $l = 2$, $h_1 = 100$, $h_2 = 300$, $r_1 = 0.3$, $r_2 = 0.7$, and $K = 2000$.

Unlike the results shown in Fig. 2.4 for the BA- n schemes (dashed lines), here the BA- n mechanisms outperform the immediate-request mechanism even when the system load is very high. Also, the performance gains of the book-ahead mechanisms are more significant than in the single class cases. The reason for this is an interesting play of effects between the two call classes (see Fig. 2.6b). Longer-duration calls will be blocked at a higher rate because of the reason provided in the previous subsection with regards to the assumed distribution for call-initiation times. As more longer-duration calls are blocked, shorter-duration calls enjoy a greater probability of fitting into gaps left in the reservation timeline. Thus, they enjoy a lower call blocking probability. The aggregate call blocking probability is much lower than with the IR scheme. The same phenomenon is observed even if we exchange the relative call-arrival rate factors to 70% for the 100-timeslot calls and 30% for the 300-timeslot calls.

2.4.4 Restricted call-initiation times

We present numerical results obtained from the simulation model when call-initiation time options in the BA- n schemes are restricted to fall on timeslot boundaries separated by the minimum call

holding time. In these simulations, we assume only one call class. The probability that a call chooses specific restricted timeslot boundaries is still assumed to be uniformly distributed.

2.4.4.1 Improved performance of BA-n schemes

The solid lines in Fig. 2.4 show the performance of different BA schemes under the restricted call-initiation time policy for a single class of calls with a holding time of 300 timeslots. As seen in Fig. 2.4, the performance of the restricted BA-n mechanisms is much better than the IR mechanism. Even the BA-1 scheme achieves a 28% reduction in call blocking probability relative to the IR scheme when the offered load is 120%. Using even just 3 options, the BA scheme performs almost as well as the BA-First scheme. As expected, there is no difference between BA-First with and without restrictions.

Comparing the solid and dotted utilization curves in Fig. 2.4, we see that while the BA-n schemes achieve a maximum utilization of 86% when call-initiation times are unrestricted, 100% utilization is achievable in the restricted case. For example, with the BA-3 scheme, a call blocking probability of 8% is achievable when operating the link at a 98.7% utilization with restricted call-initiation time options, while without restriction, the BA-3 scheme only achieved a maximum of 84.8% utilization with a 28.8% call blocking probability. It is important to achieve good results with the BA-n schemes, rather than with just BA-First, because we expect scientists to want the flexibility of selecting specific timeslots in which to reserve network bandwidth for their experiments, data transfers, or remote collaborations.

2.4.4.2 Impact of call holding time

Assuming a single call class, we obtain numerical results for three values of the call holding time, 100, 300, and 500 timeslots. We set the call arrival probability correspondingly so that the results can be compared for the same offered system load range. We repeat the simulations for three different values of m , 2, 5, and 10. K is fixed at a factor of 15 times the call holding time, i.e., $K = 15H$. The simulation results show that the call blocking probability is independent of the call

holding time for a fixed value of the offered load. However it does depend on m , whose effect we present in the next subsection.

2.4.5 Dependence of K on m and H

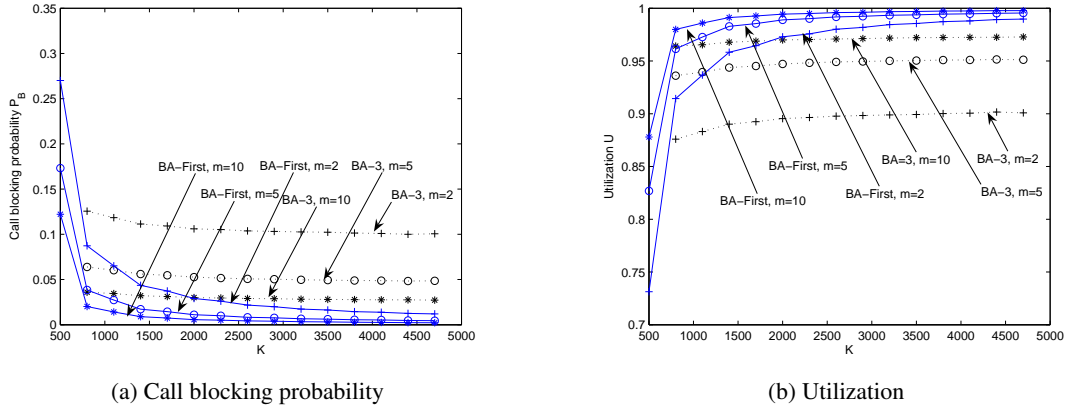


Figure 2.7: Performance of BA-3 and BA-First under different values of the advance-reservation horizon, K , and number of channels, m . $l = 1$, $h_1 = 200$, and load = 100%.

As stated in Section 2.4.1, the important input parameters are m , H and K . In this subsection, we present the impact of m and H on the selection of the advance-reservation horizon, K . The latter is a parameter that needs to be explicitly selected based on m , H , and the type of BA scheme used. It is obvious that the longer the advance-reservation horizon, the better the system performance. However, the longer the advance-reservation horizon, the greater the storage and computation needs of the scheduler. Given we do not have a closed form solution for the analytical model to derive an optimum value of K , we simulate the system to gain some understanding on acceptable values for K .

Fig. 2.7 shows simulation results for two book-ahead mechanisms, BA-3 and BA-First, for three values of m , 2, 5, and 10. Given the results of the previous subsection, which showed independence of system performance on call holding time, we simply fix the holding time to 200 timeslots. The call arrival probability p is chosen for different m so that the offered system load is fixed at 100% for all points shown in the plots. Given that the call-initiation time options are restricted, for the

BA-3 scheme, K should be at least 4 times the call holding time H to allow the user to specify three call-initiation time options. Therefore the plots for the BA-3 schemes start at $K = 800$.

Fig. 2.7 shows that the call blocking probability of the BA-First scheme approaches 0 as the advance-reservation horizon K increases. However, the performance of BA-3 is limited (the call blocking probability does not approach 0) even if K goes to infinity. The reason is that the limitation of 3 options overrides the performance gain caused by large K .

Fig. 2.7 also shows the performance improvement is small after K reaches a certain value. For example, when $m = 10$, the threshold is around 1500 for BA-3 and 2500 for BA-First, respectively. The threshold is smaller with BA-3 because, given the user is limited to specifying only 3 options, an increase in K does not yield further improvement. The implication of these results is that the reservation period does not need to be very long in order to reap the benefits of the book-ahead mechanism.

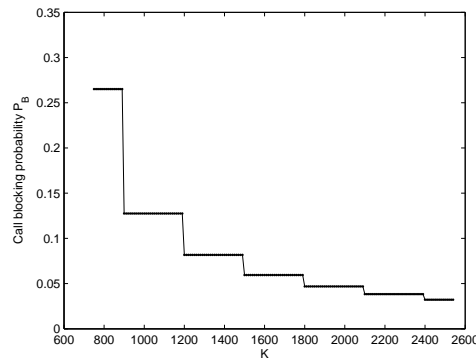


Figure 2.8: Minimum K/H values: $m=2$, $H=300$, offered load=100%, BA-First.

We observe from Fig. 2.8 that call blocking probability depends on the ratio K/H rather than on K itself. This result is explained by the fact that call-initiation time options are restricted to time boundaries separated by H timeslots. Fig. 2.7 shows that even for a small m ($m = 2$), a K/H factor of 6 is sufficient to operate the system at a call blocking probability of 5%.

From Section 2.4.4.2 we know that call blocking probability is dependent on m and offered load, but independent on H . Combining this observation with that noted in Fig. 2.8, which shows dependence on K/H rather than K , we list K/H values for different values of m corresponding to 3

Table 2.2: Dependence of K/H on m : offered load=100%, BA-First.

Call blocking probability	2%	5%	10%
$m = 2$	14	6	4
$m = 5$	5	4	3
$m = 10$	4	3	2

values of call blocking probability for the BA-First scheme in Table 2.2. It shows that to achieve a 2% call blocking probability, if $m = 2$, we need to select a much longer advance-reservation horizon K than if $m = 10$.

2.5 Sensitivity Analyses

In this section, we change some of the assumptions made in Section 2.3 to study the impact of these assumptions.

2.5.1 Nonuniform call-initiation time distribution

We change our assumption on the call-initiation time options from the uniform distribution (made for analytical tractability) to a more realistic bell-shaped distribution. For example, more airline customers reserve airline tickets one or two months in advance than six months in advance or just a few days in advance.

We assume that the first call-initiation time option specified by a user is generated using a “truncated” normal distribution. The Probability Density Function (PDF) $f_X(x)$ of this random variable X is given by

$$f_X(x) = \begin{cases} \frac{g(x)}{1 - P(X < -5 \text{ or } X > 5)} & \text{if } -5 \leq x \leq 5, \\ 0 & \text{otherwise,} \end{cases}$$

where $g(x)$ is the PDF of a normal random variable Z with mean $\mu = 0$ and standard deviation σ . We then map the advance-reservation window K of a BA- n scheme onto this $[-5, 5]$ interval and

obtain a new discrete random variable Y , whose Probability Mass Function (PMF) is given by

$$P(Y = i) = \begin{cases} \int_{-5 + \frac{10(i-1)}{K}}^{-5 + \frac{10i}{K}} f_X(x) dx & \text{if } 1 \leq i \leq K, \\ 0 & \text{otherwise.} \end{cases}$$

The probability that a user picks the i^{th} interval as the first choice can be computed as

$$P(s_1 = i) = P(Y = i).$$

Since the n options must be distinct from each other, the probability that a user picks the j^{th} interval ($1 \leq j \leq K$, $j \neq i$) as the second choice is expressed by

$$P(s_2 = j | s_1 = i) = \frac{P(Y = j)}{1 - P(Y = i)}.$$

Similarly the probability that a user picks the y_j^{th} interval as the j^{th} distinct choice is given

$$P(s_j = y_j | s_1 = y_1, s_2 = y_2, \dots, s_{j-1} = y_{j-1}) = \frac{P(Y = y_j)}{1 - \sum_{i=1}^{j-1} P(Y = y_i)}.$$

Fig. 2.9 plots call blocking probability as a function of the standard deviation σ of the base normal random variable Z for various BA schemes under the same offered load. The deviation of Z reflects the spread of the n options generated using the method described above. We vary the value of σ to study the impact of the deviation on the system performance. When σ is large, i.e., the n call-initiation time options are spread across the advance-reservation horizon and the “bell” is flat. As seen in Fig. 2.9, when σ is large, the call blocking probabilities are close to results obtained with uniformly distributed call-initiation time options. However, the call blocking probability increases as σ decreases, because the n options are likely to congregate in a small interval when σ is small. It has the same effect as if the size of the reservation horizon K is decreased. As more options are provided in the BA- n schemes, the value of σ at which the call blocking probabilities of the nonuniform schemes start to deviate from those of the uniform schemes increases.

It should be noted that we do not provide call blocking probabilities for the bell-shaped BA- n

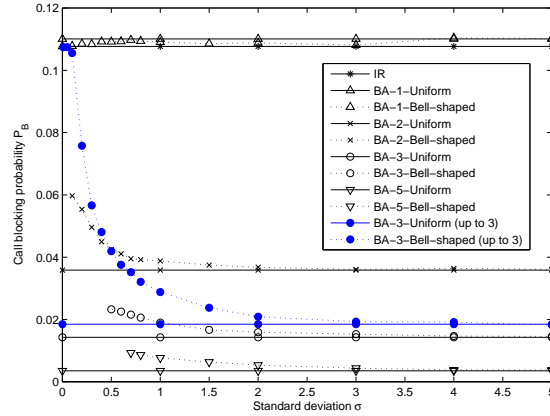


Figure 2.9: Bell-shaped call-initiation time distribution: $m = 10, K = 2000$, offered load=96%.

schemes in the area in which σ is small. When σ is small, a user is likely to pick the same option repeatedly. In other words, the enforcement of n distinct options makes the generated call-initiation time options no longer comply with the bell-shaped distribution, and the deviation of the base random variable Z no longer reflects the spread of the n options. Therefore, we stop decreasing σ when the probability that the n options are distinct is small. We argue that users will spread out the call-initiation time options when they are required to provide n distinct options. Therefore, σ will never be that small. Fig. 2.9 shows that the value of σ at which simulations is terminated increases as more options are provided in the BA- n schemes. This is because the standard deviation σ needs to be larger in order to provide more distinct options.

The above results are based on the assumption that users are forced to provide n distinct options in BA- n schemes. In reality, people may provide only one or two acceptable choices, and systems should allow users to provide *up to* n options instead of requiring exactly n options. For comparison, we plot the call blocking probability (with solid circles) for a BA-3 scheme in which the three options are not enforced to be distinct. Plots for both uniform and bell-shaped call-initiation time distributions are shown in Fig. 2.9. The call blocking probabilities of these two cases are generally higher than the cases in which the n options are enforced to be distinct. In particular, the call blocking probability for the scheme with bell-shaped call-initiation time distribution increases significantly as σ decreases to values smaller than 1.5. This is because the 3 call-initiation time

options are more likely to be the same when σ is small; therefore, the call blocking probability of the BA-3 scheme approaches the call blocking probability of the BA-1 scheme.

2.5.2 Relaxing the single-channel assumption

Another assumption we made in the analytical model and previous simulation studies is that all calls request only one channel. In this section, we study the impact of allowing calls to request an arbitrary number of channels.

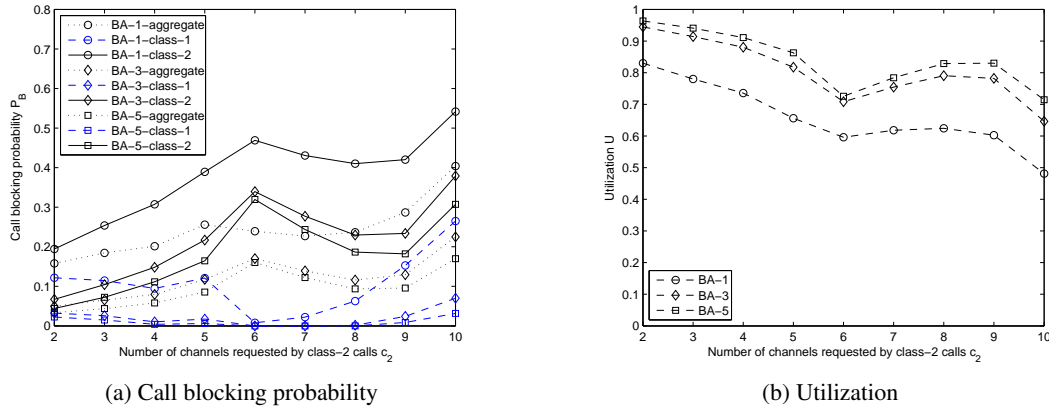


Figure 2.10: Muticlass calls under same offered load: $m = 10, K = 2000, H = 300$, offered load=100%

Fig. 2.10 shows the performance of a system that allows two classes of calls. The number of channels requested by class-1 calls, c_1 , is fixed to 1, while the number of channels requested by class-2 calls, c_2 , is varied between 2 and 10. An arriving call has an equal probability of belonging to class-1 or class-2. The offered system load $\rho = \frac{(c_1+c_2)\lambda H}{2m}$ is set to 100%. Fig. 2.10a shows that for all three BA schemes, i.e., BA-1, BA-3, and BA-5, the call blocking probability for class-2 calls increases when c_2 increases from 2 to 6. To explain this, first note that the link capacity in channels, m , is fixed to 10. As the number of channels requested by a call increases, the number of calls that the link can simultaneously accommodate decreases. In other words, a system with link capacity m and calls requesting c_2 channels is comparable to a system in which each call requests only one channel but the link capacity is only $\frac{m}{c_2}$. From [5] we know that the call blocking probability

increases as m decreases. This explains the increase in the call blocking probability of class-2 calls as c_2 increases up to 6.

The call blocking probability of class-2 calls starts to decrease when c_2 is larger than 6. This is because the arrival rate of class-2 calls decreases when c_2 increases, provided that the offered system load remains constant. Since each timeslot can accommodate at most one class-2 call when c_2 is larger than 6, a lower call arrival rate leads to a lower call blocking probability. Meanwhile, although the arrival rate of class-1 calls also decreases when c_2 increases, the call blocking probability keeps increasing in the 6 to 10 interval. This is because a class-2 call takes more channels in each timeslot when c_2 is larger, which results in less capacity left for class-1 calls in a time interval if a class-2 call is scheduled in the same interval first. The dotted lines in Fig. 2.10a shows that the aggregate call blocking probabilities increase with c_2 when $c_2 \leq 6$, but then start to decrease when $c_2 > 6$. This implies that a system designer should avoid allowing calls to request a significant fraction of the link capacity. The utilization curves shown in Fig. 2.10b confirm this observation.

The above study assumes that the offered system load remains the same as the number of channels requested by class-2 calls increases. This assumption holds for file transfers. Assigning more bandwidth to a call for a file transfer will result in a smaller holding time. Therefore, the offered load $\rho \propto (c_1 + c_2)\lambda H$ will stay the same. In contrast, for applications in which call holding times are independent of the assigned bandwidth, as c_2 increases, call blocking probability will increase assuming call arrival rates stay the same.

In a study of the number of call-initiation time options required to achieve a given call blocking probability, we fix c_2 at 10, and obtain the call blocking probability for class-2 calls and the aggregate call blocking probability. These are plotted against the number of call-initiation time options n in Fig. 2.11. The solid lines in Fig. 2.11 shows that the blocking probability of class-2 calls decreases as n increases, and this change is more evident when the offered system load is lower. Fig. 2.11 provides system designers insights on how many options the system should offer in order to achieve a certain call blocking probability for class-2 calls (solid lines) or a certain aggregate call blocking probability (dashed lines). For example, if a system is running under a 75% offered load, to ensure a call blocking probability for class-2 calls of no more than 10%, n should be at least 8.

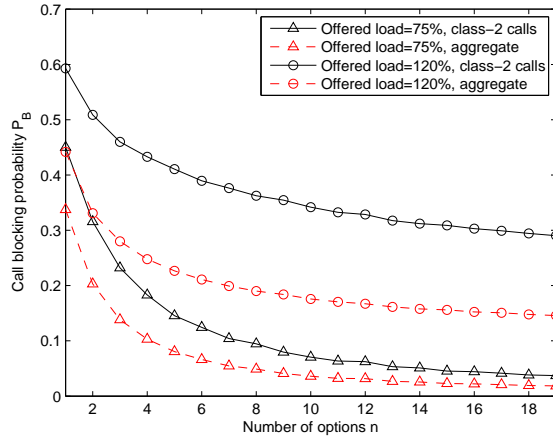


Figure 2.11: Multiclass calls under same call arrival rate, $m = 10, K = 6000, c_1 = 1, c_2 = 10$.

2.6 Multi-link Book-ahead Schemes

In this section, we study the performance of book-ahead mechanisms in multi-link scenarios. We extend our single-link simulation model to support multi-link networks.

2.6.1 Network model

We consider a linear network with L links, indexed by $1, \dots, L$, and $L + 1$ sources, indexed by $0, 1, \dots, L$, as shown in Fig. 2.12. All links have the same capacity of m channels. Source 0 generates **long-path calls** that traverse all the L links, and sources i ($1 \leq i \leq L$) generate **short-path calls** that traverse links i . We assume that the call arrival processes of all sources are identical independent Poisson processes with rate λ , i.e., $\lambda_i, i = 0, 1, \dots, L$, in Fig. 2.12 are all set to λ [42, 43].

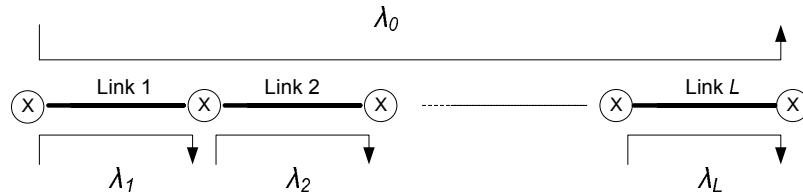


Figure 2.12: A linear network.

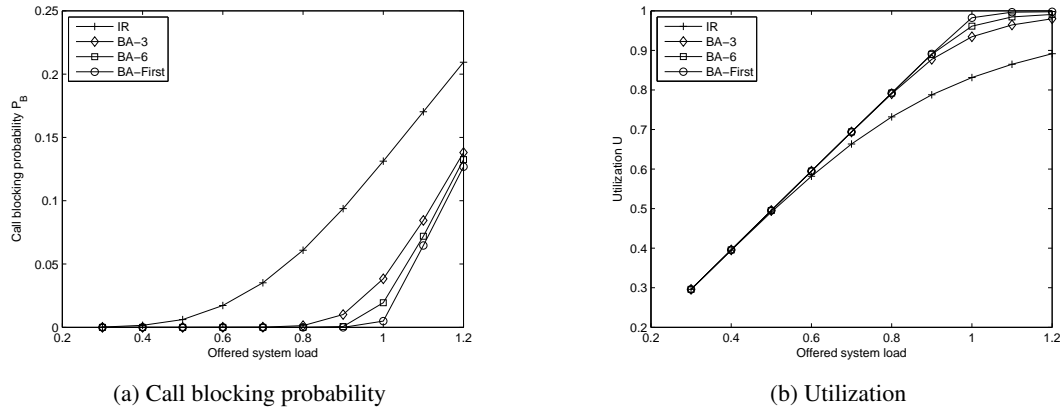


Figure 2.13: Performance of different bandwidth-sharing mechanisms in a 3-hop network. $L = 3$, $m = 10$, $H = 1$, and $K = 10$.

2.6.2 Numerical results

Fig. 2.13 plots call blocking probability and utilization in a 3-hop network. Fig. 2.13 shows that BA schemes outperform the IR scheme in the multi-link scenario as expected. Also, the number of advance-reservation options, n , does not need to be large in order to achieve a low call blocking probability and high utilization. For example, the BA-3 scheme achieves a similar value of call blocking probability as the BA-First scheme.

2.6.3 Fairness between long-path calls and short-path calls

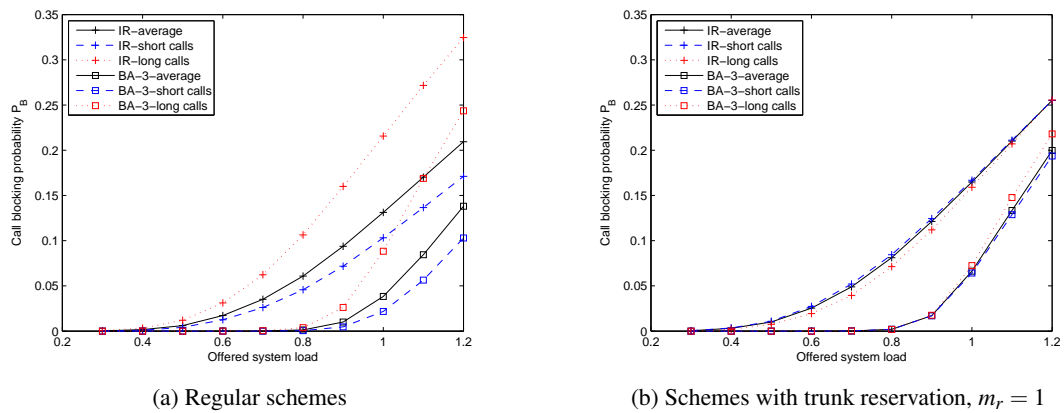


Figure 2.14: Classified call blocking probabilities. $L = 3$, $m = 10$, $H = 1$, and $K = 10$.

Besides efficiency, fairness is another essential consideration in bandwidth sharing. In this subsection we study the fairness ratio, which is defined as the ratio of the call blocking probability of long-path calls to that of short-path calls. Fig. 2.14a shows that the call blocking probabilities of long-path calls are higher than those of short-path calls in both the IR and the BA-3 schemes. For example, at 100% offered load, the call blocking probability for long-path calls in the IR scheme is 22%, while the call blocking probability for short-path calls is 11%, resulting in a fairness ratio of 0.5. In the BA-3 scheme, at the same load the ratio is 0.25.

We improve the fairness ratio using a well-known technique called “trunk reservation.” Trunk reservation is an admission control mechanism used in connection-oriented networks to prevent service degradation during overload conditions [8]. Under trunk reservation, in a 2-class system, when link utilization reaches a certain (high) level, calls of one of the classes are rejected to save resources for calls of the other class. While the specifics of how trunk reservation is used in the telephony network differ (given the goal to avoid service degradation), we apply the mechanism by using it to compensate for the fairness problem. When the link utilization is high (the number of free channels is not greater than m_r), the system will stop admitting short-path calls, and only admit long-path calls. Fig. 2.14b shows that the fairness ratio between long-path calls and short-path calls is improved significantly when trunk reservation is almost 1, i.e., the call blocking probability of long-path calls and short path calls are almost equal in both IR and BA-3 schemes. Fig. 2.15 shows that the cost of the trunk reservation mechanism is decreased utilization.

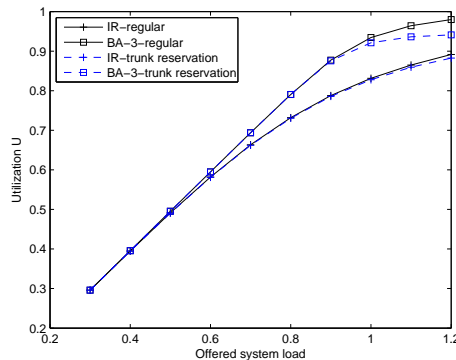


Figure 2.15: Utilization comparison of the regular schemes and the schemes with trunk reservations.

2.6.4 The impact of network scale

In this subsection, we study the impact of the network scale (in terms of the number of links L) on the design of book-ahead schemes. From Section 2.4, we know that call blocking probability depends on the link capacity m , the ratio of the advance-reservation window size to call holding times K/H , and the offered load. In multi-link scenarios, call blocking probability additionally depends on the number of links L .

We plot the call blocking probability for long-path calls as a function of K/H for BA-First systems with different values of L in Fig. 2.16. The offered load is fixed at 100%, and the link capacity m is set to 10. Fig. 2.16 provides insights on the selection of the advance-reservation window size in order to achieve a certain value for call blocking probability of long-path calls in networks of different scales. For example, to achieve a 5% call blocking probability, if $L = 3$, the advance-reservation window K should be 3 times of the holding time, while if $L = 15$, the factor should be increased to 6.

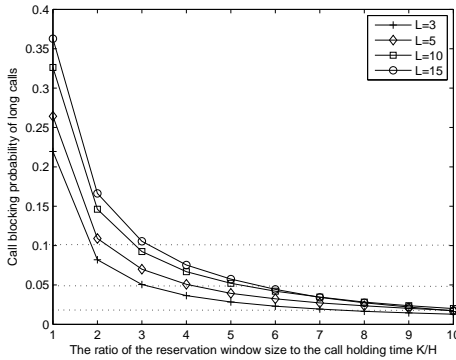


Figure 2.16: Dependence on the network scale.

The number of links in a network also has an impact on the number of trunks that need to be reserved for long-path calls in order to achieve fairness between long-path calls and short-path calls. For example, as shown in Section 2.6.3, one channel should be reserved for long-path calls in a 3-hop, 10-channel network in order to achieve a fairness ratio close to 1. However, in a 10-hop, 10-channel network, our simulation shows two channels should be reserved for long-path calls in order to achieve the same fairness ratio.

2.7 Conclusions

We presented a novel discrete-time Markov chain model of book-ahead bandwidth-sharing mechanisms for session-type requests in circuit-/VC-switched networks. We used this analytical model and a simulation model to understand the benefits of book-ahead (BA) bandwidth-sharing when compared to the immediate-request (IR) mode of bandwidth-sharing in circuit-switched networks. We studied two different BA schemes, BA-First and BA- n . In BA-First, the caller accepts any set of available timeslots, while in BA- n , the caller specifies n call-initiation time options and is admitted only if a channel is available starting from one of these n options for a contiguous set of timeslots equal in length to the requested holding time. When the number of shared channels m is small, e.g., $m = 10$, the BA-First scheme achieves 95% utilization with a call-blocking probability of only 1%, while in the IR scheme, call blocking probability is 23% even when utilization is only 80%. In the BA- n schemes, restricting the call-initiation time options to fall on timeslot boundaries separated by the minimum call holding time yields better results. Allowing users to request any call-initiation time, without regard to call holding times, results in increased blocking. This need for restrictions on call-initiation times is less important if multiple classes of calls with different call holding times are allowed; nevertheless call classes with longer holding times suffer higher call blocking probabilities than in the IR scheme if no such restrictions are imposed.

In a system with restricted call-initiation time options, we determined that the length of the advance-reservation horizon, K , required increases linearly with the holding time H . The ratio K/H is primarily dependent on m . For example, if $m = 2$, to achieve a 2% call blocking probability, the advance-reservation horizon needs to be a factor of 14 times the call holding time, while this factor drops to 4 when $m = 10$. Thus the extra data storage and processing required to accept and maintain advance reservations is not significant.

We showed that call blocking probabilities of BA schemes increase if the call-initiation time distribution is bell-shaped instead of uniform. The call blocking probability decreases with increasing deviation of the bell-shaped distribution. We also showed the interesting interactions between different classes of calls in systems that allows requests for arbitrary number of channels. Our

model provides system designers insights on how many call-initiation time options a BA-n system should support in order to achieve a certain call blocking probability for all classes of calls.

Our simulation results showed that BA schemes outperform the IR mechanism in multi-link scenarios too. We also showed that a fairness ratio close to 1 for long-path calls and short-path calls can be achieved using trunk reservation. Our multi-link simulation model also provides system designers insights on how to set the advance-reservation window size and the trunk-reservation parameters in order to achieve a certain call blocking probability and fairness ratio for networks of different scales.

Chapter 3

An Improved Model for the BA-First Mechanism

In this chapter, we present an improved analytical model for the BA-First bandwidth-sharing mechanism for *session-type* requests. This model is more scalable when compared to the model presented in Chapter 2, but it does not apply to the BA-n schemes.

We review the BA-First mechanism in Section 3.1. Section 3.2 describes the new model, and compares it with the general model presented in Chapter 2. Section 3.3 presents numerical results, showing the impact of various parameters. It also compares the results of our analytical model of the BA-First scheme with simulation results for an $M/D/m/p$ queueing system. Section 3.4 concludes this chapter.

3.1 The BA-First Mechanism

As described in the previous chapter, the BA-First mechanism for sharing the capacity of a single link works as follows. The link capacity C is discretized and expressed as m channels where the bandwidth of each channel is C/m . Time is discretized into equal-length intervals, of duration τ , as illustrated in Fig. 3.1. The reservation window consists of the set of future time intervals, for which the scheduler will accept reservation requests. The size of the reservation window is expressed as a number of time intervals, and denoted K . The scheduler maintains a record of the number of reserved channels for each time interval in the reservation window. At each time interval boundary, the reservation window slides to the right by one time interval.

Each call requests one channel for a duration of one time interval. Calls arriving in the current interval are assigned channels in the first-available reservation interval; in other words, even if the link is not fully occupied in the current interval, a newly arriving call will not be assigned a channel in that interval. Instead it will have to wait and be assigned to the earliest future interval (among the K intervals in the reservation window) in which a channel is available. A call will be blocked (rejected) if all channels have been reserved in every time interval of the entire K -interval reservation window.

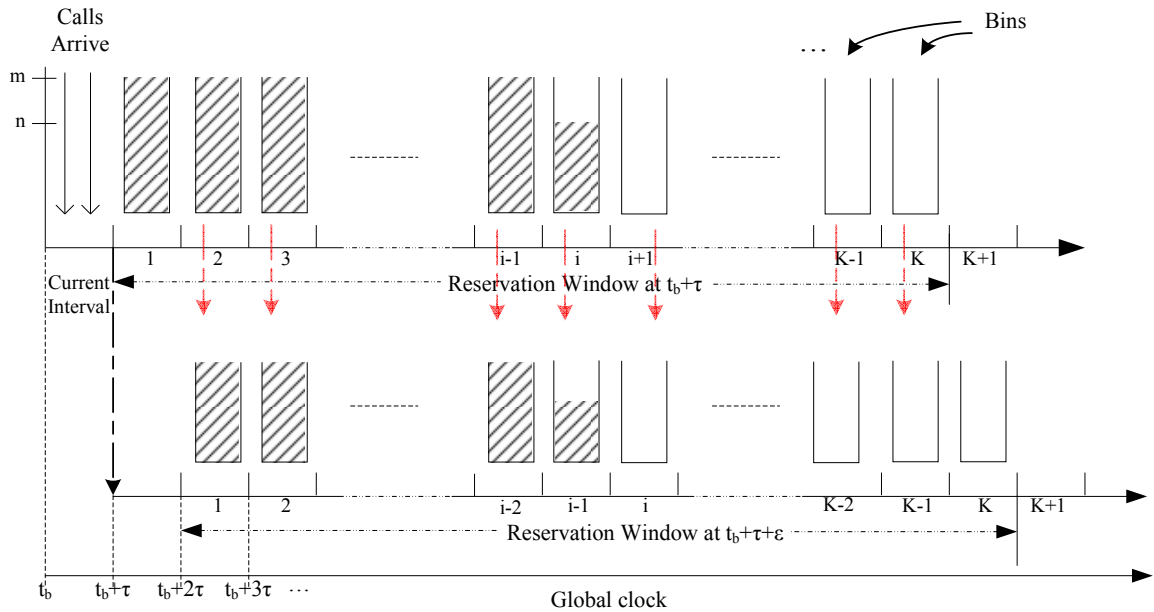


Figure 3.1: An illustration of a BA-First scheduling system

3.2 Analytical Model

We model the system with a non-homogeneous Continuous-Time Markov Chain (CTMC) in which we identify an embedded Discrete-Time Markov Chain (DTMC). We solve this model to obtain measures, such as call congestion, mean scheduling delay, and link utilization. We compare this model with the model presented in the previous chapter for general book-ahead schemes.

3.2.1 Non-homogeneous continuous-time Markov chain model

We assume calls arrive according to a Poisson process with rate λ . The time for reservation processing by the scheduler is considered negligible in the model, which allows us to have the system accept or reject calls even if they arrive just before the start of a time-interval boundary.

To help us conceptualize the system, we use the word “bin” to represent an entity that holds reservations for each of the K reservation intervals as shown in Fig. 3.1. Each bin holds reservations for its corresponding reservation interval. A bin is empty (i.e., the occupancy of a bin is 0) if no channel has as yet been reserved for its corresponding reservation interval. A bin is full (i.e., the bin occupancy is m) if all m channels are reserved for its corresponding reservation interval. Further, because a BA-First scheduler always tries to reserve the earliest available interval, if the i^{th} bin is not full, all bins after it must be empty. Therefore, the system state can be expressed as a 2-tuple (i, n) , where i is the index of the first bin that is not full, and n is the number of reserved channels in the i^{th} bin, as illustrated in Fig. 3.1. For all $1 \leq i < K$, the second element in the state vector, $n \leq (m - 1)$. For example, if $K = 4$ and $m = 2$, the first two bins are full, and the remaining two bins are empty, the state of the system is $(3, 0)$ because bin 3 is the first “non-full” bin and there are 0 entry in that bin. But when $i = K$, we need an additional state (K, m) to denote that all bins are full. The state space S is then defined as

$$S \triangleq \{s = (K, m) | s = (i, n) : 1 \leq i \leq K \ \& \ 0 \leq n \leq m - 1\}.$$

The size of the state space is $N = mK + 1$.

The system meets the Markov property requirement that the future state depends only on the current state, not on the past history. The state of the system changes when one of two events occurs: (1) a call arrives, or (2) a time-interval boundary is encountered. Since a call can arrive at any instant, the Markov chain is a CTMC. As calls arrive within the current time interval, the scheduler immediately either accepts the call by allocating it a channel in the earliest available reservation interval, or rejects the call. This causes the system state (i, n) to change upon each call arrival. Fig. 3.2A shows the state transitions for call arrivals in an example system in which $K = 4$

and $m = 2$.

Fig. 3.2B shows the state transitions at time-interval boundaries for our example system in which $K = 4$ and $m = 2$. The system state will necessarily change at these time-interval boundaries because the reservation window slides one time interval to the right causing all calls with reservations for channels in bin 1 to instantly turn “live,” reservations from bin 2 to be moved to bin 1, reservations from bin 3 to be moved to bin 2, and so forth, up to and including the last bin K , as illustrated in Fig. 3.1. This explains why in Fig. 3.2B if the state system is $(1, 1)$ just before a time-interval boundary is encountered, it changes to $(1, 0)$ at the time-interval boundary. The call with the single reservation in bin 1 goes live. Since bin 2 was necessarily empty (because bin 1 was not full, recall $m = 2$), in the new interval bin 1 remains the first non-full bin, and its occupancy is 0, i.e., the new state is $(1, 0)$. We arbitrarily pick one of two possible options to model when exactly the instantaneous change of state occurs at these time-interval boundaries. If the time-interval boundaries occur at $t_b, t_b + \tau, t_b + 2\tau, \dots$, as shown in Fig. 3.1, we select $(t_b + h\tau) + \epsilon$, where $h = 0, 1, 2, \dots$ and $\epsilon \rightarrow 0$, as the instant at which state changes occur. This convention statement is necessary for later formulations.

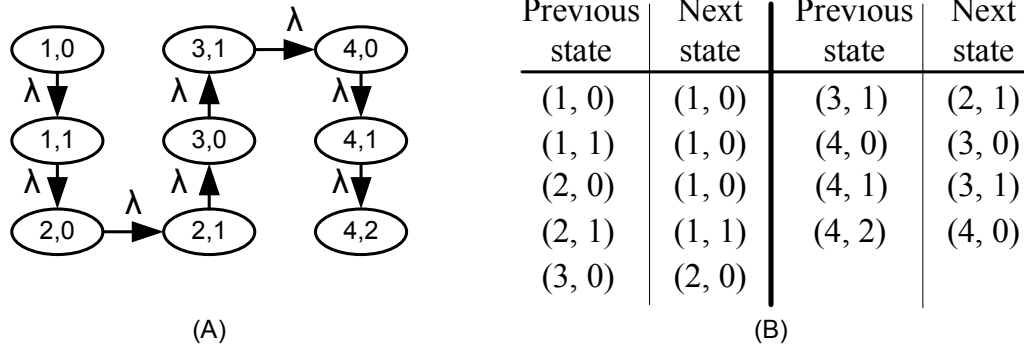


Figure 3.2: Example transition diagram for $m = 2$ and $K = 4$.

This CTMC is non-homogeneous because the system behavior at the time-interval boundaries is different from its behavior at other times. However, if we only look at the system at the time-interval boundaries (i.e., at time instants $t_b, t_b + \tau, t_b + 2\tau, \dots$), there is an embedded time-homogeneous DTMC.

3.2.2 Embedded discrete-time Markov chain model

We derive the transition probability matrix for the embedded DTMC in this subsection. The discrete time instants are the time-interval boundaries, $t_b, t_b + \tau, t_b + 2\tau, \dots$. As stated in the previous section, we adopted the convention that state transitions occur at some small time, ϵ , past each time-interval boundary. State transitions for call arrivals in the just-past time interval, e.g., $(t_b + (h-1)\tau, t_b + h\tau]$, where h is an integer, which would have occurred in the CTMC, are effectively aggregated into the one-step state transition in DTMC.

The transition probability in the DTMC from state (i, n) to state (j, q) , denoted by $P_{(i,n),(j,q)}$, is

$$P_{(i,n),(j,q)} = \begin{cases} 1 - F_A(m(K-1) + m - 1) & \text{if } i = 1 \text{ \& } (j, q) = (K, m), \\ G_A(m(j-i) + q) & \text{if } i = 1 \text{ \& } (j, q) \neq (K, m), \\ 1 - F_A(m(K-i+1) + m - n - 1) & \text{if } i \neq 1 \text{ \& } (j, q) = (K, m), \\ G_A(m(j-i+1) + q - n) & \text{if } i \neq 1 \text{ \& } (j, q) \neq (K, m), \end{cases} \quad (3.1)$$

where $F_A(a)$ is the Cumulative Distribution Function (CDF) of A , a Poisson random variable with parameter $\lambda\tau$, representing the number of call arrivals within a time interval, and $G_A(a)$ is defined as

$$G_A(a) = \begin{cases} P_A(a) & \text{if } a \geq 0, \\ 0 & \text{if } a < 0, \end{cases} \quad (3.2)$$

where $P_A(a)$ is the Probability Mass Function (PMF) of A .

Consider the first row of the Right-Hand Side (RHS) of (3.1). When $i = 1$, it means the first bin is not full. The number of reservations held in this bin, n , is irrelevant because when the next time instant in the DTMC (i.e., a time-interval boundary) arrives, these reservations turn into active calls. On the other hand, the number of calls that arrive in the just-past time interval is important. If this number is mK or higher, it means all the bins will become full and the system will transition to the (K, m) state. This is captured in the first row of (3.1), while the second row captures the case when the number of call arrivals is fewer than mK . As an example consider our $m = 2, K = 4$ system. If the source state is $(i = 1, n = 1)$ and the destination state of the transition is $(j = 4, q = 2)$, i.e.,

($j = K, q = m$), by applying the first row of the RHS of (3.1) we obtain the transition probability $p_{(1,1),(4,2)}$ as $(1 - F_A(7))$, which is the probability that more than 7 calls arrived during the current interval.

In the third and fourth rows of the RHS of (3.1), $i \neq 1$. Unlike when $i = 1$, n was irrelevant, here, when $i \neq 1$, n becomes important. For example, the first tuple in the new state j will equal $i - 1$, if the number of call arrivals in the just-past time interval is fewer than $m - n$. Again, based on whether the number of call arrivals is sufficient to completely reserve all the available channels across the entire reservation window or not, we get the third and fourth cases, respectively.

In the 2-channel, 4-timeslot example, if the original state is ($i = 2, n = 1$) and the destination state is ($j = 3, q = 0$), applying the last row of the RHS of (3.1), we obtain $G_A(3)$ as the transition probability $p_{(2,1),(3,0)}$, which is the probability that 3 calls arrived during the just-past time interval. In state ($i = 2, n = 1$), bin 1 was full, i.e., it held reservations for 2 channels, and bin 2 held a reservation for one channel. The reservations in bin 1 are irrelevant because the corresponding calls become active. When three calls arrive, the reservation that was in bin 2 in the state ($i = 2, n = 1$) goes into bin 1 when the time-interval boundary occurs. Simultaneously, the other two calls are placed in bin 2. Therefore bin 3 is the first non-full bin, making the new state (3,0).

Consider one more example to understand the reason for the second row in the definition of function $G_A(a)$ in (3.2). If the new state is (1,0) instead of (3,0), then we obtain the transition probability $p_{(2,1),(1,0)}$ as $G_A(-1)$, which is 0 according to (3.2). This is because if 0 calls arrived in the just-past interval, then the state transition will be from (2,1) to (1,1). If one or more calls arrived, then the transition will be to (1,2) or states with higher values of i . Since this transition (2,1) to (1,0) cannot occur, we obtain the 0 probability value for $p_{(2,1),(1,0)}$.

3.2.3 Performance metrics

To characterize the performance of the BA-First mechanism, in addition to the two metrics used in the previous chapter, i.e., the call congestion P_B , which is defined as the ratio of the number of blocked calls to the total number of call arrivals in a long observation interval [8], and the long-run utilization of the system U , we also examine the mean scheduling delay for admitted calls, denoted

by W . To calculate these three metrics, we first calculate the steady-state probabilities, denoted by vector π , of this discrete-time Markov chain using well-known techniques [8].

3.2.3.1 Call congestion

In any time interval, $(t_b + (h - 1)\tau, t_b + h\tau]$, where h is an integer, the average number of call arrivals is $\lambda\tau$. The number of calls blocked in this interval depends upon the state of the system at time $(t_b + (h - 1)\tau)$. We therefore use a vector $(b_{t_b} \triangleq b_{(t_b, i, n)}, (i, n) \in \mathcal{S})$, where each element $b_{(t_b, i, n)}$ denotes the blocked-call ratio in this interval if the system is in state (i, n) at the start of the interval. We define the accepted-call ratio as $Q_{(t_b, i, n)}$. Since the DTMC is time-homogeneous, for simplicity, we drop the t_b parameter, and compute $b_{(i, n)}$ as follows:

$$b_{(i, n)} = 1 - Q_{(i, n)} = 1 - \frac{\sum_{j=0}^{d_{(i, n)}} j P_A(j) + d_{(i, n)}(1 - F_A(d_{(i, n)}))}{\lambda\tau}, \quad (3.3)$$

where $P_A(i)$ and $F_A(i)$ are the PMF and CDF of the previously defined random variable A , respectively, and $d_{(i, n)}$ is the total number of channels that are available for reservation in that interval, which can be calculated as

$$d_{(i, n)} = \begin{cases} mK & \text{if } i = 1, \\ m(K - i + 1) + m - n & \text{if } i > 1. \end{cases} \quad (3.4)$$

The long-run call congestion, P_B , can then be computed as

$$P_B = \pi b^T, \quad (3.5)$$

where the components of the vector b are the blocked-call ratios computed in (3.3).

3.2.3.2 Utilization

Link utilization in a time interval $(t_b + (h - 1)\tau, t_b + h\tau]$, where h is an integer, depends upon the state of the system at time $t_b + (h - 1)\tau$, i.e., just before the state transition. It solely depends on

the occupancy of the first bin in state (i, n) because the occupancy of the first bin at time instant $t_b + (h - 1)\tau$ decides the link occupancy during the time interval $(t_b + (h - 1)\tau, t_b + h\tau]$. Therefore, if $i = 1$, which means the number of occupied channels in the first bin is less than m , the utilization of the system in this interval is $\frac{n}{m}$. Otherwise, the first bin is full, and the system utilization is 1. Again, we define the vector $(u \triangleq u_{(i,n)}, (i, n) \in S)$, where each element $u_{(i,n)}$ is the utilization if the system is in state (i, n) in any interval. Therefore $u_{(i,n)}$ can be calculated as

$$u_{(i,n)} = \begin{cases} \frac{n}{m} & \text{if } i = 1, \\ 1 & \text{if } i > 1. \end{cases} \quad (3.6)$$

The long-run link utilization U can then be calculated by

$$U = \pi u^T. \quad (3.7)$$

3.2.3.3 Scheduling delay

The scheduling delay of a call consists of a fractional part, which is the delay within the arrival interval, and an integral part, which is the number of intervals before the scheduled service interval in the reservation window. Consider an arbitrary interval $(t_b + (h - 1)\tau, t_b + h\tau]$, where h is an integer. Let $w_{(i,n)}$ be the average scheduling delay conditioned on the embedded DTMC being in state (i, n) at time $t_b + (h - 1)\tau$. The average is calculated over the calls admitted during the interval of interest. Denote $r_{(i,n)}$ and $z_{(i,n)}$ as averages of the fractional parts and integral parts of scheduling delays, respectively. We have $w_{(i,n)} = r_{(i,n)} + z_{(i,n)}$. Before computing $w_{(i,n)}$, we derive $a_{(i,n)}$, the conditional expected number of calls admitted in the interval, which can be computed as

$$a_{(i,n)} = \sum_{j=1}^{d_{(i,n)}-1} jP_A(j) + d_{(i,n)}(1 - F_A(d_{(i,n)} - 1)), \quad (3.8)$$

where $d_{(i,n)}$ is given in (3.4).

Let us focus on $r_{(i,n)}$ first. Recall that the arrival process is Poisson. Assume there are exactly j arrivals in the interval under consideration. It is a well-known result that, given that the number

of total arrivals is j in the interval, the expected time between the arrival instant of the l^{th} arrival and the end of the interval, i.e., $(t_b + h\tau)$, is $\tau(1 - \frac{l}{j+1})$ [44]. In our problem, the number of calls admitted within the interval is at most $d_{(i,n)}$. If $j \leq d_{(i,n)}$, the average fractional part of scheduling delays for j calls is $\tau(1 - (\sum_{i=1}^j \frac{i}{j+1})/j) = \frac{\tau}{2}$. If $j > d_{(i,n)}$, the average is computed over just the first $d_{(i,n)}$ calls, which results in $\tau(1 - \frac{d_{(i,n)}+1}{2(j+1)})$. Therefore, $r_{(i,n)}$ can be computed as

$$\begin{aligned} r_{(i,n)} &= \sum_{j=1}^{d_{(i,n)}} \frac{\tau}{2} P_A(j) + \sum_{j=d_{(i,n)}+1}^{\infty} P_A(j) \tau (1 - \frac{d_{(i,n)}+1}{2(j+1)}) \\ &= \tau - \frac{\tau}{2} F_A(d_{(i,n)}) - \frac{d_{(i,n)}+1}{2\lambda} (1 - F_A(d_{(i,n)} + 1)). \end{aligned} \quad (3.9)$$

Denote $v(i, n, j)$ as the sum of the integral parts of scheduling delays conditioned on the DTMC being in state (i, n) at time $t_b + (h-1)\tau$ for j calls admitted in the interval. Then, $z_{(i,n)}$ can be computed as

$$z_{(i,n)} = \frac{1}{a_{(i,n)}} \left(\sum_{j=1}^{d_{(i,n)}-1} P_A(j) v(i, n, j) + (1 - F_A(d_{(i,n)} - 1)) v(i, n, d_{(i,n)}) \right), \quad (3.10)$$

where $a_{(i,n)}$ is obtained in (3.8), and $v(i, n, j)$ can be computed by

$$v(i, n, j) = \begin{cases} \tau \sum_{l=1}^j \lfloor \frac{l-1}{m} \rfloor & \text{if } i = 1, \\ \tau \sum_{l=1}^j ((i-2) + \lfloor \frac{l+n-1}{m} \rfloor) & \text{if } i > 1. \end{cases} \quad (3.11)$$

The long-run mean scheduling delay W is

$$W = \frac{\sum_{(i,n) \in S} \pi_{(i,n)} a_{(i,n)} w_{(i,n)}}{\sum_{(i,n) \in S} \pi_{(i,n)} a_{(i,n)}}. \quad (3.12)$$

3.2.4 Model comparison

We compare the analytical model presented in this chapter for the BA-First mechanism with the model presented in the previous chapter. The latter is a model developed for the more general BA- n scheme; BA-First is a special case of BA- n in which $n = K$. By limiting this model to just BA-First schemes, we are able to improve the scalability of the model (recall that the BA- n model

could be solved on a general purpose computer for only the $m = 1$ case). There are two reasons for the improved scalability. First, since a BA-First scheduler always tries to reserve the earliest available interval, if the i^{th} interval is not full, all intervals after it must be empty. Therefore, the system state can be expressed by the index of the first interval that is not full and the number of reserved channels in that interval. The size of the state space is $(mK + 1)$. In contrast, a BA-n call can request any interval, therefore, the system state has to maintain information on the number of reserved channels for each interval within the reservation window, which makes the size of the state space much larger at $(m + 1)^K$. Second, to create a DTMC model of the BA-n system, we had to discretize time into very small intervals to allow for a geometric distribution approximation of the exponential distribution of call inter-arrival times. This has the effect of requiring the reservation window K to be on the order of thousands rather than hundreds. Recall that the state space size of this DTMC model is $(m + 1)^K$, which makes this larger K significant. These two reasons explain why the new model described in this chapter is more scalable than the general model presented in the previous chapter.

3.3 Numerical Results

We show numerical results obtained from the analytical model of the BA-First system, and show that this model can be used as a solution for an $M/D/M/p$ queueing system. As discussed in Section 2.4.2, we primarily relied on expert intuition and peer reviews for model validation, because this system is yet to be implemented and deployed. This work is published in a peer-reviewed paper in the *Proceedings of IEEE Global Telecommunications Conference (Globecom)* [21]. Model verification is achieved by a comparison of the results generated by the embedded DTMC model described in this chapter with the general DTMC model of Chapter 2.

3.3.1 Sensitivity analysis on the link capacity

Fig. 3.3 plots the call blocking probabilities of the BA-First and IR schemes under two different values of utilization, 80% and 90%, as a function of the link capacity m . The figure shows that

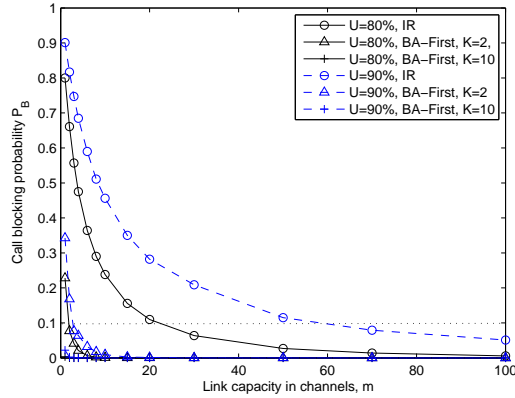


Figure 3.3: Restricted IR vs. BA-First under different values of m .

BA-First outperforms IR for all values of link capacity. However, when m is large, i.e., the per-call bandwidth is small relative to link capacity, the performance difference between BA-First and IR is small. The value of m at which BA-First should be employed increases as we increased our utilization goal. For example, to obtain a call blocking probability of 10%, if the desired utilization is 80%, BA-First should be employed instead of IR if $m \leq 22$. When the desired utilization is 90%, this threshold increases to $m = 60$.

3.3.2 The effect of K

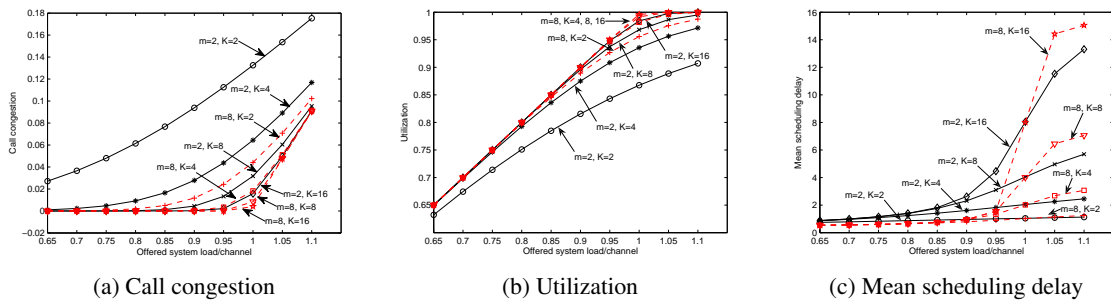


Figure 3.4: The effect of reservation window size on the system performance.

In this subsection we use the analytical model to study the effect of the reservation window size, K , on system performance for two values of m , 2 and 8. Fig. 3.4 plots three metrics, call congestion, link utilization, and mean scheduling delay. For a given value of m , which is determined by the

per-VC bandwidth granularity used for sharing the link, our model allows a designer to select an appropriate value for the reservation window size. For example, if we choose a system load/channel of 1 (since it is a blocking system, this load can be greater than 1), and want to run the system at a call congestion level of 4%, then if $m = 2$, we need a reservation window of 8 time intervals, but if $m = 8$, this number is only 4 time intervals. We also see that if $m = 8$, the call congestion and utilization plots for $K = 4$, $K = 8$, and $K = 16$ almost overlap, while the mean scheduling delay increases significantly as K increases. Thus $K = 4$ is the right choice when $m = 8$. In summary, these results show that increasing the reservation window size beyond a certain level can be detrimental to system performance. We should choose the smallest K value to achieve a desired call congestion level at the expected system load so that the mean scheduling delay remains low (especially at high loads).

3.3.3 System design

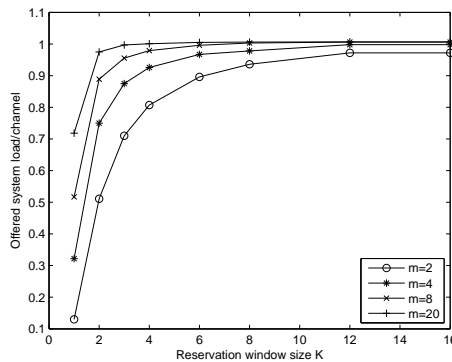


Figure 3.5: Offered system load vs. reservation window size.

The model can be used to provide guidelines for other design decisions too. Consider a system designer who wants to know the payoff of increasing the reservation window size, with respect to the multiplexing factor on a shared link. Our model can be used to help answer this question. Fig. 3.5 plots the offered system load/channel against K for a fixed call congestion, 1%, for different values of m . From this figure, we observe that for a shared link with $m = 4$, by increasing K from 2 to 4, the system load/channel can be increased from 75% to 93%. This is quite significant in that it

allows for a 23.5% increase in the number of endpoints multiplexed on to the link.

3.3.4 Use of our model as a solution for the $M/D/m/p$ queueing system

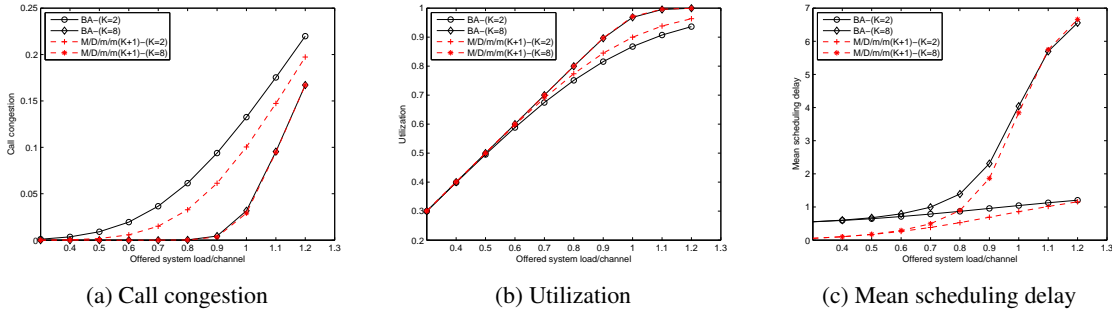


Figure 3.6: The comparison of BA-First mode and $M/D/m/p$ queueing model.

We recognized that our model for the BA-First scheme can be used as a solution for the $M/D/m/p$ queueing system. The reservation window K is comparable to the $p - m$ buffer in an $M/D/m/p$ queueing system. The call arrival process is exponential, and hence consistent with the ' M ' in the $M/D/m/p$ queueing system. Additionally, the fixed holding time in a BA-First model matches the ' D ' in the $M/D/m/p$ queueing system. Thus a BA-First system model with parameters m and K is comparable to a $M/D/m/m(K + 1)$ queueing system to have an equal number of buffer locations. As there is no existing analytical solution for the $M/D/m/p$ system (to our knowledge), we implemented a simulation model to verify our BA-First model.

Fig. 3.6 shows that the call congestion and link utilization metrics are almost the same for both systems. The mean scheduling delay of the BA-First mode is roughly half a time interval higher than that of the $M/D/m/m(K + 1)$ queueing system when the system load is low. This is due to the fractional part of the scheduling delay in the BA-First system, in which calls cannot be served immediately in the arrival intervals. At high loads, the mean scheduling delay is almost the same in the two systems. In summary, our model, designed for the BA-First scheme, can be used as an approximate solution for an $M/D/m/p$ queueing system at moderate-to-high loads.

3.4 Conclusions

While it is difficult to create a scalable analytical model suitable for the BA-n mechanism, we were able to model the BA-First system for session-type requests using a non-homogeneous CTMC. Observing the system at the discrete time-interval boundaries, we extracted an embedded DTMC, and solved it for the steady-state probabilities. Using these, we obtained solutions for significant metrics, such as call congestion, link utilization, and mean scheduling delay. We demonstrated the use of the model as a design tool with numerical results. For example, with the model, a designer can determine that a 23.5% increase in system load can be accommodated while maintaining a 1% call congestion rate when K , the size of the reservation window, is increased from 2 to 4 for a 2-channel link. Additionally, we showed that this model is a good approximation for an $M/D/m/p$ queueing system at moderate-to-high loads.

Chapter 4

Book-ahead Mechanisms for Data-type Requests

Geographically-distributed scientists often have to download large (e.g., terabyte-sized) files from remote sites [45]. In this chapter, we design a book-ahead mechanism for *data-type* requests, i.e., file transfers, in connection-oriented networks, and evaluate its performance. In this BA scheme, designed to support only file transfers (i.e., no session-type requests are permitted), reservation requests specify the size of the file to be transferred. A simple algorithm for the scheduler receiving these reservation requests is a greedy scheme in which it schedules each file transfer to start as soon as possible. We call this algorithm Fixed-Bandwidth Delayed Start (FBDS) because the bandwidth assigned to a request is fixed during the entire transfer time period, and the start of the transfer could be delayed if all bandwidth resources are already in use for other transfers. As the scheduler knows the transfer schedules of all ongoing file transfers, it can respond to the reservation request with a start time and corresponding bandwidth allocation.

There is a problem in this FBDS use of connection-oriented networks, in that a file transfer cannot take advantage of bandwidth that becomes available subsequent to its start. This is in contrast with connectionless packet-switched networks in which since each packet is transmitted at the full rate of a link; when one file transfer completes, other ongoing transfers can increase their sending rates.

To address this problem we propose an enhanced algorithm called Varying-Bandwidth Delayed Start (VBDS) for use in connection-oriented networks. With this mechanism, a file transfer can be

allocated varying levels of bandwidth for the duration of the transfer. Such an allocation is possible since all reservation requests specify file sizes; this allows the VBDS scheduler to keep a record of the completion times of all ongoing file transfers, thus allowing it to allocate a higher level of bandwidth at a later time of the file transfer. The cost of implementing VBDS is that circuit switches need to be reprogrammed multiple times within a transfer, while in the FBDS scheme, switches are only programmed at the start and the end of a call. Our VBDS algorithm is an improved version of VBLS scheme proposed in [46–48].

We demonstrate through simulations that the VBDS algorithm improves performance over the FBDS algorithm significantly, making it indistinguishable from packet switching. In practice, because packet switches have finite buffers, buffer-overflow losses can occur under high loads, requiring congestion control algorithms, such as TCP Slow Start and congestion avoidance. These algorithms degrade throughput, making our solution of using connection-oriented networks with VBDS a better alternative for file transfers. The overhead of scheduling makes VBDS-on-connection-oriented-networks solution unsuitable for files of small-to-moderate sizes. However, for large files, it definitely holds promise.

The rest of this chapter is organized as follows. Section 4.1 reviews related work on file transfer scheduling and discusses the motivation of this work. Section 4.2 defines the file transfer scheduling problem. Section 4.3 presents the VBDS algorithm. Section 4.4 analyzes the performance of VBDS, and compares it with FBDS and an idealized packet switching scheme. We conclude this chapter in Section 4.5.

4.1 Motivation and Related Work

We first review prior work on file transfer scheduling and bin packing algorithms. We then discuss problems with file transfers in connectionless IP networks, and the advantages of using connection-oriented networks for file transfers.

The seminal paper by Coffman et al. [49] describes file transfer scheduling schemes for a star network assuming all transfers are of unit bandwidth but arbitrary durations. The paper by Erlebach

and Jansen [50] extends this work to star and tree networks with arbitrary bandwidth and arbitrary durations. Both papers obtain competitive ratios for online greedy heuristics called list scheduling algorithms, characterizing their performance when compared to offline optimal solutions. The performance metric compared is makespan, the total time required to transfer a set of files. The basic **list scheduling (LS)** algorithm works as follows: if there is a call i such that its required bandwidth R^i is available on all links of the end-to-end path, LS schedules the first such call in the list of all calls. If not, it waits until one of the active transfers completes. This heuristic is extended for arbitrary bandwidth in a star network to Decreasing Bandwidth List Scheduling (DBLS), and for trees as Level-List Scheduling (LLS) and List Scheduling by Levels (LSL) schemes [50]. Both [49] and [50] require file transfer requests to specify a bandwidth requirement, and schedule the constant bandwidth requested for each transfer. As we noted in the beginning of this chapter, this poses a problem in that it could lead to poor utilization and larger delays. Our scheme aims at overcoming these drawbacks by having the scheduler make a varying bandwidth allocation.

In bin packing problems, the goal is to pack a given set of blocks into finite-sized bins using the smallest number of bins [51]. Our file transfer scheduler has to solve a similar problem, fitting a transfer into a schedule. In container loading problems, the goal is to fill a single bin of infinite height to a minimum possible height [52]. In the knapsack loading problem, each block has an associated profit and the problem is to select those blocks that will maximize profit when loaded into the bin [53]. This classification of packing problems is obtained from [54]. In none of these problems can the blocks be broken up into pieces to fit into bins, which is what happens when we schedule files with varying bandwidth allocation in different time ranges. Thus, the heuristics proposed for these problems do not apply. Solutions to the job shop scheduling problem [55] typically compute optimal schedules that can be used for the offline problem equivalent to our online problem. Our problem is online because resources have to be scheduled for a file transfer without information about file transfer requests that arrive subsequent to the request being scheduled.

It has been noted that transferring large datasets across high-bandwidth, long-delay links is inefficient in the current Internet [56–58]. The network service provided by the Internet is a connectionless best effort service. Transport protocols are required to provide congestion control, flow

control and error control. The most popular transport protocol, TCP, has been found inadequate on high-bandwidth, long-delay links [56–58]. The main problem is with its Additive Increase Multiplicative Decrease (AIMD) congestion control algorithm, which is considered too slow in adapting sending rates to available capacity and too drastic in cutting back when network congestion is detected. A number of modifications and enhancements to TCP have been proposed. These include BIC-TCP [59], HighSpeed TCP [60], Scalable TCP [61], and FAST TCP [62]. The main goal of these TCP variants is to determine the fair share of a flow faster and more accurately. In contrast, the share of a flow in a connection-oriented network is determined during circuit/virtual circuit(VC) establishment. Once a circuit/VC is set up, the fair share is known and fixed. Efficient transport protocols have been designed for connection-oriented networks to take advantage of this known rate. For example, Circuit-TCP (C-TCP) is a transport protocol designed for file transfers across dedicated circuits [63]. By disabling TCP’s Slow Start and AIMD algorithms, C-TCP outperforms both Reno-TCP and BIC-TCP, achieving high throughputs relative to link capacities, even in high-bandwidth, long-delay environments.

A Varying-Bandwidth Transport Protocol (VBTP) [64] was proposed to work in conjunction with VBLS scheduling in connection-oriented networks. The VBTP sender and receiver synchronously adapt their rates allocated at the time of reservation. Thus, if the capacity allocated to a transfer is 0 within some time ranges, the sending software stops sending data and refrains from reading data from the disk to avoid transmit-buffer overflows.

4.2 Problem Definition

We formulate a file transfer scheduling problem as follows: end host applications specify a triple for each file, (S, R_{max}, T_r) , where S is a file size, R_{max} is a maximum acceptable bandwidth, and T_r is a desired start time for the transfer. In theory, file transfers can be carried out at any data rate. However, in practice, various constraints of end hosts, such as disk speeds, bus rates, processing speeds, limit the maximum transfer rate. Hence, we require users to specify a maximum acceptable rate R_{max} , and then have the scheduler assign a bandwidth at some rate no higher than R_{max} .

Requiring applications to specify file sizes is atypical of current-day implementations such as FTP or GridFTP [65] [66]. However, if a transfer is requested by the server on which the file is stored, then file size information is readily available. File transfer applications, such as FTP, need to be upgraded anyway to work with transport protocols such as VBTP. A module that requests file servers to make scheduling requests on behalf of their clients can be added as part of these upgrades.

4.3 VBDS Mechanism

In this section, we describe our VBDS scheme for scheduling file-transfer requests on an m -channel single-link network. Table 4.1 lists our notation.

Table 4.1: Notation.

Symbol	Meaning
S^i	File transfer size requested by call i .
T_r^i	Start time requested by call i .
R_{max}^i	Maximum rate of call i , expressed as a number of channels.
$TRC^i = \{(B_k^i, E_k^i, \mathbf{C}_k^i), k = 1, \dots, \tau^i\}$	Time-Range-Channel allocation: channels in set \mathbf{C}_k^i are assigned to call i in time range k starting at B_k^i and ending at E_k^i .
$\gamma(t)$	Channel availability function: indicates the availability of each channel at time t . For more details, see Section 4.3.1.
m	Link capacity expressed as a number of channels.
χ	Per-channel bandwidth.
T_p	Time needed to program a crossconnection in a switch (switch programming time).
T_d	Unit of time discretization.
\mathbf{C}_a	The set of channels available in the current time range.
\mathbf{C}_p	The set of channels assigned to a call in the previous time range.
\mathbf{C}_o	The set of open channels, which are defined as the channels available in the current time range and were assigned to the call in the previous time range ($\mathbf{C}_a \cap \mathbf{C}_p$).
\mathbf{C}_n	The set of non-open channels, which are defined as the channels available in the current time range but were not assigned to the call in the previous time range ($\mathbf{C}_a \cap \bar{\mathbf{C}}_p$).

4.3.1 VBDS overview

Upon receiving a reservation request (S^i, R_{max}^i, T_r^i) , VBDS scheduler checks the existing bandwidth usage schedule, and returns a varying-bandwidth allocation. We call this varying-bandwidth allocation a Time-Range-Channel (*TRC*) vector. A *TRC* vector TRC^i allocated for the i^{th} transfer is characterized as $\{(B_k^i, E_k^i, \mathbf{C}_k^i), k = 1, 2, \dots, \tau^i\}$, where B_k^i is the start time of the k^{th} time range, E_k^i is the end time of the k^{th} time range, and \mathbf{C}_k^i is the set of channels allocated to the transfer in the k^{th} time range. The switch maintains an available channel function $\gamma(t)$. $\gamma(t)$ is expressed in the following form:
$$\left\{ \begin{array}{ll} b[m]_z & P_z \leq t < P_{z+1}, \\ \{11\dots 1\} & t \geq P_{z_{max}}, \end{array} \right. \quad \text{where } z = 1, 2, \dots, z_{max} - 1, \text{ and } b[m]_z \text{ is a bitmap with } m \text{ bits in which the } i^{th} \text{ bit indicates whether channel } i \text{ is available in the time range } [P_z, P_{z+1}) \text{ or not.}$$
 If $b[i] = 1$, then channel i is available. If $b[i] = 0$, then channel i is unavailable. The bitmap $\{11\dots 1\}$ indicates that all channels are available. z_{max} denotes the number of times that $\gamma(t)$ changes value before the time instant $t = P_{z_{max}}$ after which all m channels of the link remain available. Fig. 4.1a shows the link status of a 4-channel link. The shaded areas represent periods in which the channel has been reserved, while blank areas represent the periods in which the channel is available. In this example, $\gamma(t)$ changes values 6 times, i.e., $z_{max} = 6$. The six time instants at which the $\gamma(t)$ changes values are $P_1 = 0, P_2 = 10, P_3 = 30, P_4 = 60, P_5 = 70$, and $P_{z_{max}} = P_6 = 80$. The corresponding $\gamma(t)$ values at these instants are $\gamma(0) = \{0000\}, \gamma(10) = \{0011\}, \gamma(30) = \{0001\}, \gamma(60) = \{0111\}, \gamma(70) = \{0101\}$, and $\gamma(80) = \{1111\}$.

Upon receiving the three-tuple (S^i, R_{max}^i, T_r^i) in request i , the switch responds with TRC^i , which details the allocation of channels in different time ranges for request i . The allocation is made on a round-by-round basis, where a round consists of the procedures used to allocate channels for a time range that extends between two consecutive change points in $\gamma(t)$. The channels assigned to a call in a time range is recorded in set \mathbf{C}_p before proceeding to the next time range. In the next time range, the available-channel set \mathbf{C}_a is divided into two sets based on whether or not a channel was assigned to the call in the previous time range: open-channel set \mathbf{C}_o , which contains available channels that were assigned to the call in the previous time range, i.e., $\mathbf{C}_o = \mathbf{C}_a \cap \mathbf{C}_p$, and non-open-channel set \mathbf{C}_n , which contains available channels that were not assigned to the call in the previous time range,

i.e., $\mathbf{C}_n = \mathbf{C}_a \cap \overline{\mathbf{C}_p}$. The scheduler prefers to assign channels in \mathbf{C}_o to the call in the new time range because this reduces switch reprogramming. We define four cases based on the relation between the sizes of the sets \mathbf{C}_o and \mathbf{C}_n , and the value of R_{max}^i . At the end of each round, the scheduler computes the size of the residual data (i.e., the yet-to-be-transferred portion of the file), and starts the next round if this size is not zero.

4.3.2 Detailed description of VBDS

Start algorithm:

Set time $\nu \leftarrow T_r^i$, and the residual data size $\theta \leftarrow S^i$; $k \leftarrow 1$; $\mathbf{C}_o \leftarrow \phi$; $\mathbf{C}_n \leftarrow \phi$; $\mathbf{C}_p \leftarrow \phi$.

Repeat loop (start next round):

Find z such that $P_z \leq \nu < P_{z+1}$ in the channel availability function $\gamma(t)$.

If $\gamma(\nu) = \{00\dots0\}$ (No channel available in this time range)

Set $\nu \leftarrow P_{z+1}$. Continue (repeat loop).

else

Step 1: Record the start time of the current time range: $B_k^i \leftarrow \nu$. Divide the available-channel set \mathbf{C}_a into \mathbf{C}_o and \mathbf{C}_n : $\mathbf{C}_o \leftarrow \mathbf{C}_a \cap \mathbf{C}_p$, $\mathbf{C}_n \leftarrow \mathbf{C}_a \cap \overline{\mathbf{C}_p}$

Step 2: Determine which channel(s) to assign to the call in this time range.

Case 1: The number of open channels is the same as R_{max}^i , i.e., $|\mathbf{C}_o| = R_{max}^i$, then assign all open channels to the call: $\mathbf{C}_k^i \leftarrow \mathbf{C}_o$.

Case 2: The number of open channels is fewer than R_{max}^i , but the transfer will complete with just the open channels before the switch can even be programmed to crossconnect the non-open channels, i.e., $|\mathbf{C}_o| < R_{max}^i$ and $|\mathbf{C}_o| \chi T_p \geq \theta$, then assign all open channels to the call: $\mathbf{C}_k^i \leftarrow \mathbf{C}_o$. It is counter-productive to assign non-open channels to the call since the file transfer will complete before the switch can be programmed to crossconnect the non-open channels, an operation that is required prior to data transfer.

Case 3: The number of open channels is fewer than R_{max}^i , the number of all available channels is not greater than R_{max}^i , and the residual data size is large enough to warrant programming the switch to crossconnect the non-open channels, i.e., $|\mathbf{C}_o| < R_{max}^i$,

$|\mathbf{C}_a| \leq R_{max}^i$, and $|\mathbf{C}_o|\chi T_p < \theta$, then assign all available channels to the call: $\mathbf{C}_k^i \leftarrow \mathbf{C}_a$.

Case 4: The number of open channels is fewer than R_{max}^i , the number of all available channels is greater than R_{max}^i , and the residual data size is large enough to warrant programming the switch to crossconnect the non-open channels, i.e., $|\mathbf{C}_o| < R_{max}^i$, $|\mathbf{C}_a| > R_{max}^i$, and $|\mathbf{C}_o|\chi T_p < \theta$, then assign all open channels to the call, and assign $(R_{max}^i - |\mathbf{C}_o|)$ non-open channels with the longest available time ranges to the call. The reason for choosing the channels with the longest available time ranges is to reduce the number of switch reprogramming operations.

Step 3: Compute the residual data size and the end time of this time range. Open channels can continue data transfer right from the start of the time range, while non-open channels have to wait until the channels are crossconnected before data can be transferred on these channels. The total transfer capacity of the time range, C_t , can be computed as $C_t = (P_{z+1} - \nu)\chi|\mathbf{C}_o| + (P_{z+1} - \nu - T_p)\chi(|\mathbf{C}_k^i| - |\mathbf{C}_o|)$. The residual data size and the end time of this time range can now be computed as follows:

if $\theta < C_t$ (the residual data size is less than the transfer capacity of the current time range)

if $\theta < T_p\chi|\mathbf{C}_o|$ (the transfer does not require any additional available channels given the switch programming overhead)

$$E_k^i \leftarrow \nu + \frac{\theta}{\chi|\mathbf{C}_o|}, \theta \leftarrow 0$$

else

$$E_k^i \leftarrow \nu + T_p + \frac{\theta - T_p\chi|\mathbf{C}_o|}{\chi|\mathbf{C}_k^i|}, \theta \leftarrow 0$$

endif

Terminate repeat loop.

else

$$E_k^i \leftarrow P_{z+1}, \theta \leftarrow \theta - C_t$$

Continue repeat loop.

endif

Record the assigned channels in set \mathbf{C}_p : $\mathbf{C}_p \leftarrow \mathbf{C}_k^i$.

Increase the time range index: $k = k + 1$, and update $\gamma(t)$.

endif

End repeat loop.

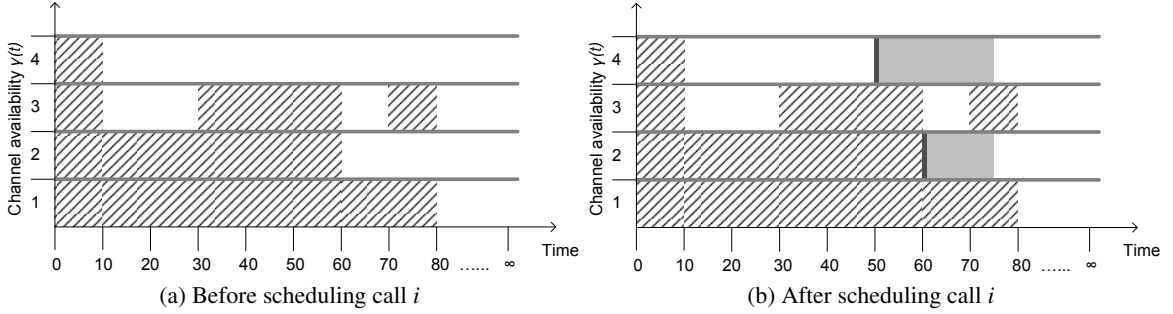


Figure 4.1: Example of the VBDS mechanism.

As an example of VBDS, consider scheduling a transfer of a $5GB$ file with a T_r of 50, and an R_{max} of 2 channels on a 4-channel link. Let the per-channel link bandwidth χ be $10Gbps$, and the unit of time discretization be $100ms$. Switch programming time is assumed to be 1 unit. Assume the link state is as shown in Fig. 4.1a. In the first time range $50 \leq t < 60$, $\mathbf{C}_p = \mathbf{C}_o = \emptyset$, and $\mathbf{C}_a = \mathbf{C}_n = \{4\}$. This is a Case-3 scenario, since $|\mathbf{C}_o| = 0 < R_{max}$, $|\mathbf{C}_a| = 1 < R_{max}$, and the residual data size is large enough to warrant programming the switch to crossconnect some non-open channels. Therefore, the scheduler assigns all the channels in the non-open-channel set \mathbf{C}_n , i.e., channel 4, to the call, and it determines TRC_1 to be $(50, 60, \{4\})$ at the end of the first round of the scheduling. Within this time range, after programming the switch in 1 time unit, $1.125GB$ data can be transferred in the remaining 9 time units ($10Gbps \times 9 \times 100ms$).

In the next range, which is the $60 \leq t < 70$ range, $\mathbf{C}_p = \mathbf{C}_o = \{4\}$, $\mathbf{C}_a = \{2, 3, 4\}$, and $\mathbf{C}_n = \{2, 3\}$. This is a Case-4 scenario where $|\mathbf{C}_o| = 1 < R_{max}$, $|\mathbf{C}_a| = 3 > R_{max}$, and it is worth programming the switch to crossconnect a non-open channel. Therefore, the scheduler allocates all channels in the open-channel set \mathbf{C}_o , which is channel 4, to the call, and a choice is made between the two non-open channels 2 and 3. The scheduler chooses channel 2 because it has a longer available range. At the end of the second round, the scheduler determines TRC_2 to be $(60, 70, \{2, 4\})$. Accounting for the time required to program the switch to crossconnect channel 2, $2.375GB$ can be transferred in the $60 \leq t < 70$ range. The size of the residual data is

$(5GB - 1.125GB - 2.375GB = 1.5GB)$.

In the third round, $\mathbf{C}_p = \mathbf{C}_o = \{2, 4\}$, $\mathbf{C}_a = \{2, 3, 4\}$, and $\mathbf{C}_n = \{3\}$. This is a case-1 scenario where $|\mathbf{C}_o| = R_{max}$. The scheduler assigns channels 2 and 4 to the call, and recognizes that the transfer of the residual 1.5GB data will be completed at $t = 75$. It determines TRC_3 to be $(70, 75, \{2, 4\})$. The final TRC for this request is: $\{(50, 60, \{4\}), (60, 70, \{2, 4\}), (70, 75, \{2, 4\})\}$, where each tuple is of the form (B_k, E_k, \mathbf{C}_k) , and τ , the number of ranges for this request, is 3. After scheduling this request, the channel availability function $\gamma(t)$ is updated as shown in Fig. 4.1b. The grey areas indicate the allocation for this request, and the dark grey areas represent the switch programming periods.

4.4 Analysis and Simulation Results

We analyze the performance of VBDS, and then compare it with FBDS and packet switching. The primary metric of interest is **throughput**, which is defined as the long-run average of the ratio of file size to file transfer delay. The file transfer delay of file i is defined as the time duration between the requested start time T_r^i and the instant when the transmission of the file is complete. We also obtain **file latency**, which we define to be the mean waiting time across all transferred files. Waiting time for a given file transfer i is defined as the time duration between T_r^i and the time instant at which the first bit of the file is transmitted. This is sometimes referred to as “response time” in other systems. When comparing VBDS with FBDS and packet switching, we also consider **normalized delay**, which is the file transfer delay divided by file size. From a file-transfer point of view, normalized delay is the “average bit delay.”

We describe our traffic model in Section 4.4.1. In Section 4.4.2, we describe how we verify our simulation model with theoretical results. In Section 4.4.3 we describe our sensitivity studies for the maximum rate requested by calls. We study the impact of a designer’s choice of the time-discretization unit time and the switch programming time on VBDS performance in Sections 4.4.4 and 4.4.5, respectively. In Section 4.4.6, we describe a simulation comparison of VBDS with FBDS and packet switching.

4.4.1 Traffic model

We select input parameters and distributions for our model based on real system measurements [34, 67]. We assume that file transfer requests arrive according to a Poisson process with rate λ based on the findings in [34, 67]. The requested start time T_r for each transfer is assumed to be the call arrival instant, i.e., all calls request to an immediate start. We assume that file sizes are distributed according to a bounded Pareto distribution [67]. Specifically, the probability density function of the file size is given by:

$$f_X(x) = \frac{\alpha k^\alpha x^{-\alpha-1}}{1 - (\frac{k}{p})^\alpha}, \quad k \leq x \leq p, \quad (4.1)$$

where α is the shape parameter, and k and p are the lower and upper bounds, respectively, of the allowed file-size range.

4.4.2 Verification of the simulation model against analytical results

To verify the implementation of our VBDS simulation model, we compare the results for a simple case in which all file requests set their R_{max} to match the link capacity, C , with a theoretical result available for this scenario, i.e., the analytical solution to $M/G/1$ queueing system. Let the call arrival rate and the service time be λ and X , respectively. The average waiting time $E(W)$ can be calculated as follows [41]:

$$E(W) = \frac{\lambda E[X^2]}{2(1-\rho)}, \quad (4.2)$$

where $E[X^2]$ is the second moment of X and $\rho \equiv \lambda E[X]$ is the system load. Using the file-size distribution specified in (4.1), and the link capacity C , we can obtain

$$E[X^2] = \frac{1}{C^2} \int_{-\infty}^{\infty} (x^2 f_X(x)) dx = \frac{1}{C^2} \int_k^p (x^2 f_X(x)) dx = \frac{1}{C^2} \left(\frac{k^\alpha}{1 - (\frac{k}{p})^\alpha} \right) \left(\frac{\alpha}{\alpha - 2} \right) \left(\frac{1}{k^{\alpha-2}} - \frac{1}{p^{\alpha-2}} \right). \quad (4.3)$$

Fig. 4.2 shows our numerical results comparing the mean waiting time calculated by the analytical model in (4.2) with our simulation results. Input parameter values are as follows: $k = 500MB$, $p = 100GB$, and $\alpha = 1.1$. We fix the mean file size and link capacity, and hence $E[X]$. This means

that to generate an increasing system load, we simply increase the call arrival rate λ . Note that for stability the system load must be below 1. Fig. 4.2 shows that the analysis and simulation results match closely.

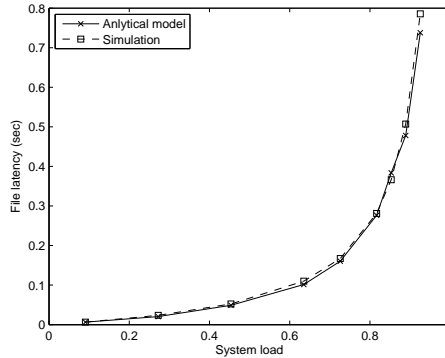


Figure 4.2: File latency comparison between analytical and simulation results.

4.4.3 Sensitivity of results to the per-call maximum-rate request

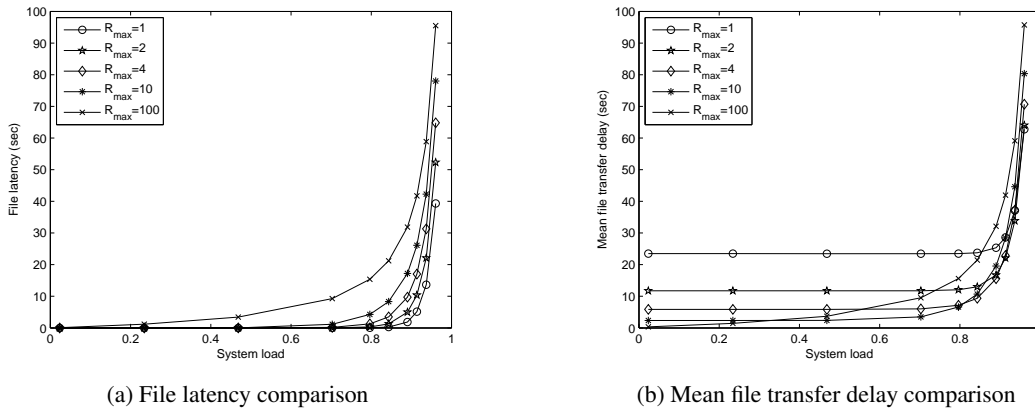


Figure 4.3: A comparison of VBDS file latency and mean file transfer delay for files requesting same R_{max} ; link capacity is 100 channels.

In this subsection, we carry out experiments to understand the impact of R_{max} . We assume that all calls request the same R_{max} , and study the system performance under different R_{max} values for 1, 2, 4, 10, and 100 channels, when the link capacity C is assumed to be 100 channels. The single channel rate χ is assumed to be $1Gbps$. Other parameter values are: $k = 500MB$, $p = 1TB$, and

$\alpha = 1.1$. The actual rate of a channel is not as significant as the ratio of C to R_{max} . As can be expected, if R_{max} for all calls is 1 channel, we obtain lower file latency (mean waiting time) than in the case where all files request an R_{max} equal to the link capacity (100 channels). However, if we consider the mean file transfer delay, which includes the file latency and the mean service time (transmission delay), we obtain different results, as shown in Fig. 4.3b. Under low loads, allocating only 1 channel per call results in the worst performance (highest file transfer delay), while under high loads, the maximum-rate allocation of 100 channels per call is the worst solution.

4.4.4 Sensitivity of results to the unit of time discretization

As shown in Section 4.3.1, a penalty with VBDS is that the network has to perform more bookkeeping and the tracking of file sizes in order to know when all ongoing transfers will complete, while in IP networks where routers do not maintain such state information. This information is kept in the form of $\gamma(t)$. In order to reduce the number of changes in $\gamma(t)$ so that we can reduce the complexity of the algorithm, we discretize time into timeslots, and limit the changes in channel allocations to fall on timeslot boundaries. The discretization also helps with switch programming. Switches have to be reprogrammed only on timeslot boundaries. With discretization the control processor can be triggered at every timeslot boundary to check whether any circuits need reconfiguration. Without discretization, an interrupt based mechanism has to be implemented, which is more complex.

In the following experiment, we assume that 30% of calls specify an R_{max} of 1 channel, 30% specify 2 channels, and the remaining 40% specify 4 channels. Link capacity is assumed to be 100 channels. We study the impact of the discretization unit T_d at four sample points: 0.05, 0.5, 1, and 2 seconds.

Fig. 4.4 shows a comparison of the throughput for different values of T_d . The plots are categorized according to the value of R_{max} . The rationale of doing this is that throughputs (especially at low loads) are naturally limited by their R_{max} values. Therefore, comparing throughputs for files with different values of R_{max} is inappropriate.

Fig. 4.4 shows that the throughput decreases as T_d increases. This is because, with larger T_d , there are more unused time ranges. Fig. 4.4 also shows that the impact of time discretization is

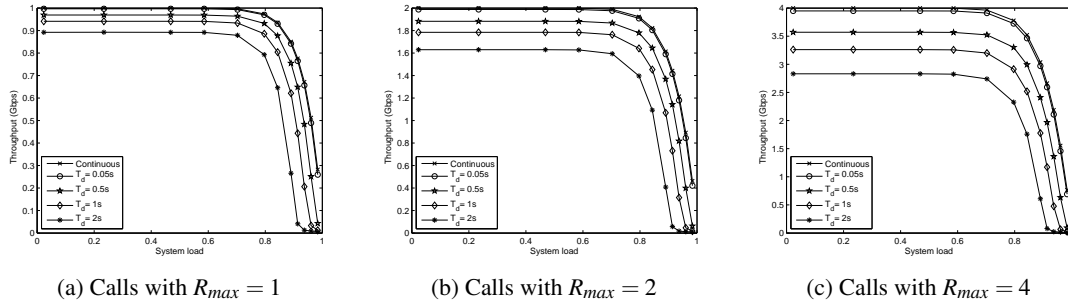


Figure 4.4: Throughput comparison for different values of T_d (0.05, 0.5, 1, and 2 sec).

more significant for calls specifying larger values of R_{max} because these calls need relatively shorter duration to complete their file transfers.

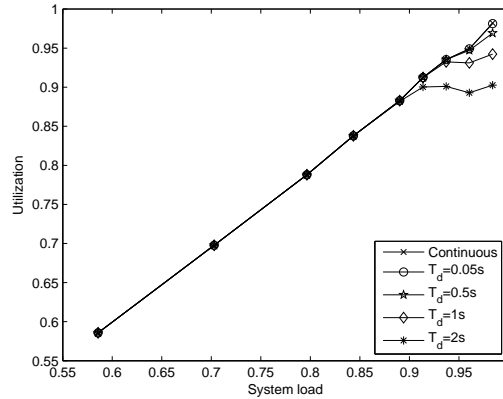


Figure 4.5: Utilization comparison for different values of T_d (0.05, 0.5, 1, and 2 sec).

Fig. 4.5 compares utilization for different values of T_d . Utilization drops only under high load conditions. To understand the impact on utilization, we derive an upper bound for a discretization-related utilization penalty. Consider an implementation of VBDS in which the discretization unit is T_d seconds. A file transfer, which takes T seconds under a continuous-time VBDS, will need $\lceil T/T_d \rceil$ timeslots to complete its transfer under a discrete-time VBDS. The difference $\lceil T/T_d \rceil \times T_d - T$ represents a loss in “transfer capacity.” Let us denote the sizes of a sequence of files as $S^i, i = 1, 2, \dots$. Therefore an upper bound on the loss in transfer capacity (in bits) for the i^{th} file is $\chi R_{max}^i T_d$. Therefore the overall loss in transfer capacity for the first n files, expressed as a fraction

of the total number of bits transferred, is upper bounded by $(\sum_{i=1}^n \chi R_{max}^i T_d) / (\sum_{i=1}^n S^i)$. If all S^i , $i = 1, 2, \dots$, have identical distributions, and all R_{max}^i , $i = 1, 2, \dots$, also have identical distributions, we can compute an upper bound for the discretization-related utilization penalty as

$$\lim_{n \rightarrow \infty} \frac{\sum_{i=1}^n \chi R_{max}^i T_d}{\sum_{i=1}^n S^i} = \frac{\chi E(R_{max}) T_d}{E(S)}. \quad (4.4)$$

For example, in our experiment, where the probabilities that calls specifying 1, 2 and 4 channels are 30%, 30% and 40%, respectively, and the per-channel bandwidth, χ , is 1Gbps, we can calculate an upper bound for the system utilization loss using (4.4). If T_d equals 0.05 sec, the utilization penalty is less than 0.534%. However, this value increases as T_d increases. If T_d equals 2 sec, the upper bound for the utilization penalty increases to 21.3%.

Fig. 4.6 shows the advantage of larger discretization units when considering the average number of change points in $\gamma(t)$. We see that the reduction in the number of change points in $\gamma(t)$ is significant when the load is high. This reduction is expected to be more prominent in multi-link scenarios.

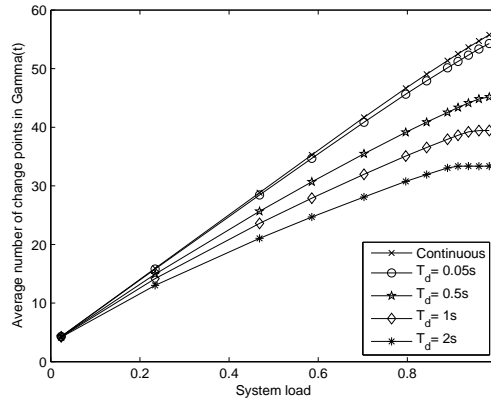


Figure 4.6: Number of change points in $\gamma(t)$ for different values of T_d .

4.4.5 Impact of switch programming time

As noted in Section 4.3.1, unlike in the fixed-bandwidth allocation mode where the switch is only programmed at the start and end of a call, a VBDS allocation could require circuit switches to be

reprogrammed multiple times within a transfer. With electronic TDM switches, where switch programming times are in the order of nanoseconds [68], the impact of reprogramming on utilization will be less significant than in networks with optical Micro-Electro-Mechanical Switches (MEMS), in which switch programming times are in the order of milliseconds. We carry out a series of experiments to measure the impact of the switch programming time on the performance of VBDS. We found that the impact is small. This is especially true when all reservation request set T_r , the desired start time, to “immediate.” The reason is that when all calls request an immediate start, once a call acquires a channel, it can keep the channel until it finishes the transfer. Therefore, the number of channels allocated to a call will never decrease during the lifetime of the call. The number of time switches need to be reprogrammed within the lifetime of a call is bounded by the maximum number of channels assigned to the call. For example, if a $500MB$ file request specifies an R_{max} of 4 channels, the most optimistic transmission time is $1s$, while the switch programming penalty is utmost $4 * 5ms = 20ms$ (assume the switch programming time is $5ms$). The ratio of the switch programming penalty to the total transmission time is small (0.02) even in this worst-case scenario. We recognize that switch programming times will have a larger impact if calls specify a future start time because the channels allocated to a call C may have to be released for calls that were scheduled before C .

4.4.6 Comparison of VBDS with FBDS and PS

The primary objective of this simulation study is to compare the performance of VBDS with that of two alternative schemes: (1) a packet-switched (PS) system, and (2) FBDS (the greedy scheme that schedules each file transfer request to start as soon as possible, using a fixed bandwidth of R_{max}). The rationale for this comparison is to illustrate that although VBDS is a circuit-based resource-sharing scheme, it achieves file-transfer throughput close to that of packet switching. As pointed out before, standard circuit switching using FBDS will result in significantly lower throughput than packet switching because ongoing file transfers cannot exploit bandwidth released by completed file transfers. However, the varying-bandwidth nature of VBDS mitigates this loss of throughput by exploiting the bandwidth that becomes available after the start of a transfer.

The PS system is an idealized system in that the buffers are assumed to be infinitely large. Files are divided into packets of 1500 bytes. Packets arrive at a constant rate equal to R_{max} divided by the packet length.

For the input parameters, we choose $\alpha = 1.1$, $k = 500MB$, and $p = 1TB$. We allow three values for R_{max} , 1, 2, and 4 channels, with corresponding probabilities of 0.3, 0.3, and 0.4, respectively. The link capacity, C , is 4 channels with a per-channel rate of $1Gbps$. T_d and T_p are set to $50ms$ and $5ns$, respectively.

4.4.6.1 Throughput results

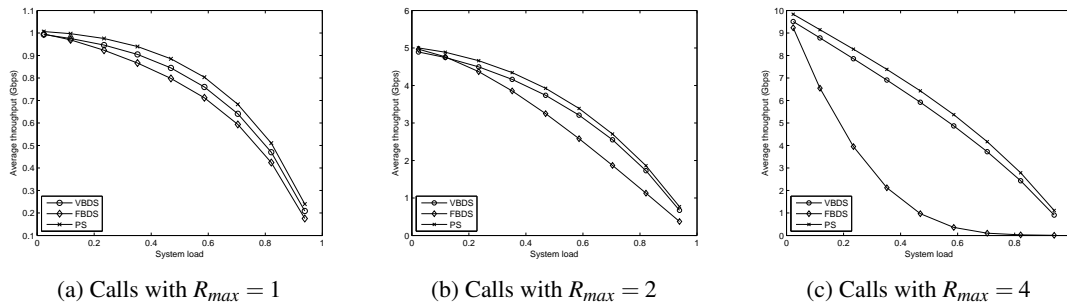


Figure 4.7: A comparison of FBDS, PS, and VBDS throughputs.

Fig. 4.7 plots the throughput (in $Gbps$) as a function of system load for VBDS, FBDS, and PS. The throughput of the VBDS scheme is significantly higher than that of the FBDS scheme in the entire load region, and almost the same as that of the PS scheme. This serves to illustrate our main point that by taking into account file sizes and allocating a varying-bandwidth schedule for each transfer, we can eliminate the problem associated with circuit-switched networks.

4.4.6.2 Normalized delay results

Fig. 4.8 plots normalized delay as a function of system load for VBDS, FBDS and PS. The results are similar to the results of the throughput comparison: VBDS achieves lower delay than FBDS, but similar performance as the PS scheme.

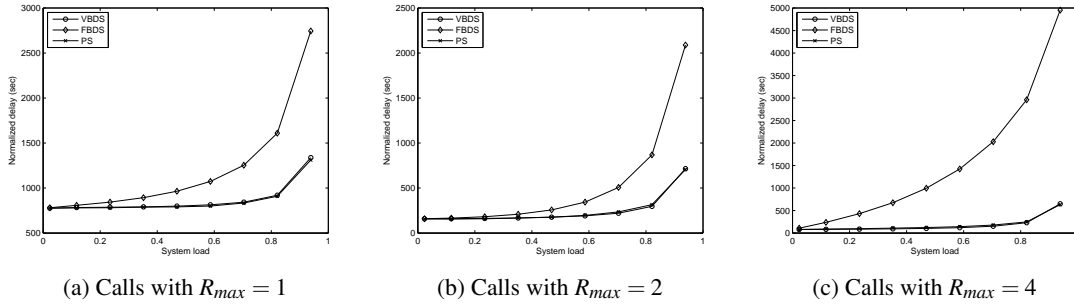


Figure 4.8: A comparison of FBDS, PS, and VBDS normalized delays.

Next we derive a lower bound for the normalized delay in the VBDS scheme. When the system load is low, files experience almost no waiting delay, and the file transfer delay approximately equals the file size divided by R_{max} . Let D^i denote the file transfer delay of the i^{th} file. The normalized delay is calculated as $(\sum_{i=1}^n S^i D^i) / (\sum_{i=1}^n S^i)$. Substituting D^i by $\frac{S^i}{R_{max}\lambda}$, we get $(\sum_{i=1}^n S^i \frac{S^i}{R_{max}\lambda}) / (\sum_{i=1}^n S^i)$, which converges to $\frac{E(S^2)/E(S)}{R_{max}\lambda}$ as $n \rightarrow \infty$. Given the file sizes in the experiment follow a bounded Pareto distribution with $k = 500MB$, $p = 1TB$, and $\alpha = 1.1$, $E(S^2)/E(S) = 779.95Gb$. Therefore, for $R_{max} = 1, 2, 4$, a lower bounds for the normalized delay are 779.94, 155.99, and 77.99 seconds, respectively, as shown in Fig. 4.8.

4.4.6.3 Improvements to files of different sizes

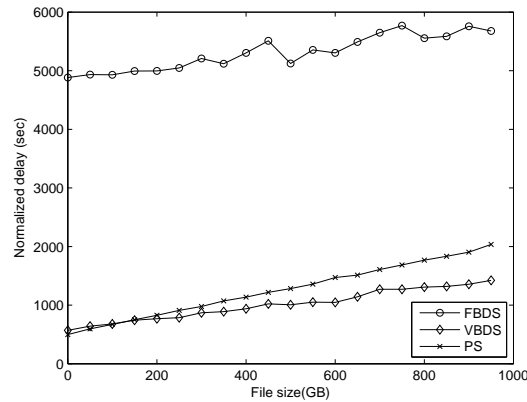


Figure 4.9: Comparison of the improvements to files of different size ($\rho = 0.94, R_{max} = 4$).

In order to study the performance of VBDS for files of different sizes, we focus on requests specifying a R_{max} of 4 channels, divide them into 20 bins based on file sizes, and plot normalized delays for files in each bin. From the results shown in Fig. 4.9, we observe that large files experience lower normalized delays with VBDS than with PS. In PS, new requests are automatically admitted causing a degradation in service for ongoing transfers. This degradation is greater for large files. Therefore, VBDS outperforms PS for large files.

4.5 Conclusions

We designed a book-ahead mechanism called Varying-Bandwidth Delayed Start (VBDS) for data-type requests in connection-oriented networks. By having end host applications specify file sizes in their reservation requests, VBDS is able to make a Time-Range-Channel vector allocation for transfers. This approach overcomes a well-known drawback of using circuits for file transfers in which a fixed-bandwidth allocation mode fails to allow users to take advantage of bandwidth that becomes available subsequent to the start of a transfer. We demonstrate through simulations that with VBDS we can improve performance over fixed-bandwidth schemes significantly, making it indistinguishable from packet switching. Further, we see that VBDS favors large files to moderate-sized files when compared to packet switching. Sensitivity of results to switch programming time and the time-discretization unit show that these practical considerations do not have a significant impact on the performance of VBDS.

Chapter 5

Immediate-Request Bandwidth-Sharing Mode

In this chapter, we describe our deployment of an experimental wide-area GMPLS network [58] and the implementation of software packages to support the immediate-request bandwidth-sharing mode in this network [23]. The network is called **CHEETAH**, which stands for “Circuit-switched High-speed End-to-End Transport Architecture.” The CHEETAH network consists of “Ethernet-SONET circuit-based gateways.” An “Ethernet-SONET circuit-based gateway” is a network node that has Ethernet interface cards and SONET interface cards, and can be programmed to crossconnect any port on an Ethernet interface card to an equivalent-rate time-division multiplexed SONET signal on any port of the SONET interface cards. Technologies to transparently carry Ethernet frames within SONET frames have been defined and implemented in Ethernet-SONET circuit-based gateways. Given that most end host network interface cards (NICs) are Ethernet based, these circuit-based gateways are critical to realizing our goal of connecting distant end hosts with wide-area dedicated circuits. An end-to-end circuit consists of Ethernet segments at the edges mapped to SONET circuits (carrying the Ethernet signals transparently) through metro-/wide-area segments. We chose this combination of Ethernet and SONET for our experimental testbed to ease technology transfer to production networks in which Ethernet dominates LANs and SONET dominates MANs/WANs.

By using GbE and 10GbE interfaces in hosts and gateways, and SONET based circuit-switched service in the WAN, CHEETAH meets both the high-speed and predictable service requirements

of scientific applications. To meet the dynamic bandwidth-sharing requirement, CHEETAH is built with switches and gateways that implement GMPLS control-plane protocols.

We describe **several interesting problems** we encountered in the design and implementation of the CHEETAH network, and our solutions to these problems. Our design philosophy is to select solutions that will allow for the scalability of the CHEETAH network and its interconnection to other GMPLS networks. These goals lead us to recommend the use of IPv6 static public IP addresses for most of the interfaces in a GMPLS network, the use of Domain Name System (DNS) to perform wide-area IP-to-MAC address resolution, the use of dynamic updates to IP and Address Resolution Protocol (ARP) tables at end hosts after the setup/release of wide-area Ethernet-SONET-Ethernet circuits, the use of the Internet as the control-plane network, and the use of IPsec to secure the control plane.

To bring the benefits of CHEETAH service to scientists without requiring their deliberate participation in the invocation of this service, we implement a software package called **CHEETAH software** for Linux end hosts, and provide a **CHEETAH-API (Application Programming Interface)** for application programmers. For example, by integrating this API into an FTP server, such as vsftpd [69], a scientist using vsftpd would enjoy the high-speed and predictable CHEETAH service while being unaware that the vsftpd program has dynamically invoked the setup of a dedicated end-to-end CHEETAH circuit prior to executing data transfers. The end-host CHEETAH software package consists of many components, primary among which is an RSVP-TE module. This module initiates the setup and teardown of end-to-end circuits across the CHEETAH network in response to requests from applications.

Finally, we describe **signaling experiments** that we conducted across the CHEETAH network and provide measurements of end-to-end circuit setup delay for different types of circuits. A typical number for a wide-area SONET circuit that traverses two switches is 166ms.

The rest of this chapter is organized as follows. Section 5.1 reviews GMPLS control-plane protocols, and summarizes the CHEETAH concept and related work. Section 5.2 describes the wide-area CHEETAH experimental network, including details on IP addressing and control-plane security. We describe the end-host CHEETAH software package in Section 5.3. Section 5.4 de-

scribes the signaling performance experiments conducted on the CHEETAH network to obtain call setup delay measurements. We conclude the chapter in Section 5.5.

5.1 Background and Related Work

5.1.1 GMPLS control-plane protocols

There are three major components in GMPLS control-plane protocols: RSVP-TE signaling protocol, Open Shortest Path First - Traffic Engineering (OSPF-TE) routing protocol [4], and Link Management Protocol (LMP) [70]. These three protocols are designed to be implemented in a control processor associated with each network switch to provide corresponding functionality for its switch. Each of these protocols provides an increasing degree of automation, and correspondingly a decreasing dependence upon manual network administration.

The main purpose of the LMP module is to automatically establish and manage the control channels between adjacent nodes, to discover and verify data-plane connectivity, and to correlate data-plane link properties. The purpose of the OSPF-TE routing protocol software module located at a switch is to enable the switch to send topology, reachability and loading conditions of its interfaces to other switches, and receive corresponding information from other switches. The purpose of the RSVP-TE signaling engine at a switch is to manage the bandwidth of all interfaces on the switch, and to program the data-plane switch hardware enabling it to forward demultiplexed incoming user bits or packets as and when they arrive. A typical RSVP-TE engine implemented in switches executes five steps when it receives a connection setup request (i.e., an RSVP-TE *Path* message): signaling message parsing, route determination, connection admission control, data-plane configuration (i.e., switch fabric configuration), and signaling message construction.

5.1.2 CHEETAH Concept

The CHEETAH architecture, first proposed in [58], is illustrated in Fig. 5.1. End hosts are equipped with two Ethernet NICs. The primary NICs (NIC1) in the end hosts are connected to the public Internet through the usual LAN Ethernet switches and/or IP routers, while the secondary NICs

(NIC2) are connected to Ethernet ports on Ethernet-SONET circuit gateways. Ethernet-SONET circuit gateways, in turn, are connected to wide-area SONET circuit-switched networks, in which both circuit gateways and pure SONET switches are equipped with GMPLS protocols to support call-by-call dynamic bandwidth sharing. Dynamic bandwidth sharing through a distributed mechanism, such as the one offered by GMPLS control-plane protocols, is critical to achieving scalability, an important factor for any network as noted in [71]. End-to-end CHEETAH circuits (as shown with dashed lines in Fig. 5.1) are set up dynamically between end hosts with RSVP-TE signaling messages being processed at each intermediate gateway/switch in a hop-by-hop manner (*Path* messages 1-5 in the forward direction and *Resv* messages 6-10 in the reverse direction).

CHEETAH is designed as an “**add-on**” service to existing Internet connectivity, leveraging the services of the latter. This design brings the following benefits: (i) The connection to the Internet allows a CHEETAH end host to communicate with other non-CHEETAH hosts on the Internet while it is in a communication session with another CHEETAH end host through a dedicated circuit; (ii) Signaling messages can be transmitted through the primary NICs and the Internet eliminating the need for a dedicated signaling channel; (iii) For applications such as file transfers that can use both the Internet and the CHEETAH paths, end-host software can selectively choose to request CHEETAH circuits only when the Internet path is estimated to provide a lower service quality than the CHEETAH circuit, and further “fall back” to the Internet path if the CHEETAH circuit setup attempt fails due to an unavailability of circuit resources on the CHEETAH network.

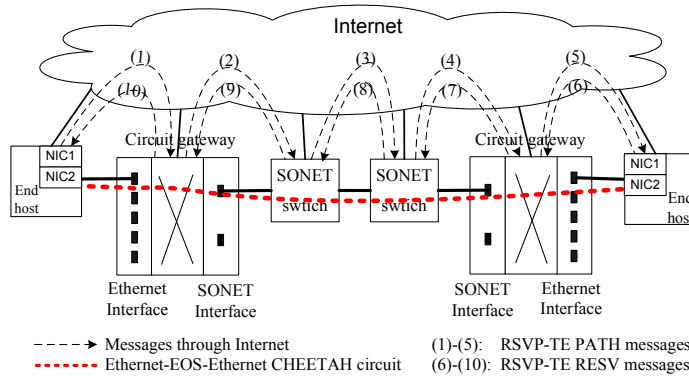


Figure 5.1: CHEETAH concept.

The trigger for CHEETAH circuit setup is provided by end host applications. If source code is available for an application, we propose modifying it to make use of CHEETAH circuits as and when appropriate. If source code is unavailable, users are provided a program called a circuitrequestor to first set up a dedicated host-to-host circuit before initiating the application. As triggers for circuit setup come from applications running on end hosts, automatic large flow detection for the purpose of initiating circuit setup as was proposed for ATM switched virtual circuits in [72] and for circuits in [73] is not required in CHEETAH.

5.1.3 Related work

Other optical connection-oriented testbeds being deployed in support of eScience applications include CANARIE's CA*net 4 [14], OMNInet [74], SURFnet [16], UKLight [15], DOE's UltraScience net [75], NSF DRAGON [76] and NSF UltraLight [77] networks. These networks differ in their choice of technology, between SONET/SDH, WDM and SDM, and in their control-plane solutions. For example, UltraScience net uses SONET switches, while the DRAGON network uses WDM switches. Similarly, on the control plane, CA*net 4 implements User Controlled LightPath (UCLP) software [78], while DRAGON is based on GMPLS control-plane protocols.

In spite of these differences, the goal of most of the above-mentioned testbeds is to enable the dynamic creation of long-held high-speed circuits that are reserved in advance for specific scientific projects. These circuits are often used to interconnect routers or Ethernet switches in user/application-specific network topologies. The CHEETAH project differs from these testbeds primarily in its goal to create large-scale networks, and hence emphasizes the bandwidth sharing aspect of GMPLS networks. With end-host CHEETAH software and the CHEETAH-API integrated into end-host applications, the goal is to encourage bandwidth-sharing for short-duration calls that are triggered automatically by application software such as vsftpd and web servers. Circuits extend between end hosts rather than IP routers or Ethernet switches, and are established immediately upon request rather than in advance.

5.2 CHEETAH Wide-area Network

This section describes the wide-area CHEETAH experimental network, and several interesting problems that we encountered relative to the design of the control-plane network, IP addressing, address resolution and control-plane security. We describe our solutions to these various problems.

5.2.1 Network switches and hosts

We selected the Sycamore SN16000 Intelligent Optical Switch for the primary network nodes in the CHEETAH network. Our reasons for doing so were **two-fold**. **First**, the SN16000 switch implements a standardized data-plane solution for Ethernet-SONET mapping called Generic Framing Procedure (GFP) [79]. In addition, it implements Virtual Concatenation, which is also a standardized mechanism, to select the SONET rate to be as closely matched to the Ethernet rate as possible [80]. For example, a 1Gb/s Ethernet signal is mapped to a 21-OC-1 virtually concatenated SONET signal instead of an OC-48 (2.5Gb/s) signal, which would have been the appropriate level to use in the original SONET hierarchy.

Second, the Sycamore SN16000 switch has a robust implementation of GMPLS control-plane protocols, RSVP-TE and OSPF-TE, which includes the major GMPLS control-plane-related IETF RFCs, RFC 3473, 3477, 3945, 3946 for RSVP-TE, and RFC 3630 for OSPF-TE. While these specifications specify procedures for homogeneous circuits (e.g., pure-SONET, where the incoming and outgoing ports used on the circuit are both SONET), they do not support hybrid circuits, i.e., Ethernet-SONET circuits. To establish such circuits, most equipment require administrators to use network management software. Standardization is underway to extend RSVP-TE to handle such hybrid circuits. Meanwhile, Sycamore provided us a version of their control-plane software, which implements a proprietary solution for RSVP-TE control of such hybrid circuits. The built-in RSVP-TE software at the SN16000 switches makes these switches easy to use. We simply configure parameters at each switch to specify the data and control-plane addresses of the remote ends. Our end-host RSVP-TE software inter-operates with the built-in RSVP-TE engines of these switches.

The **end hosts** on the CHEETAH network are general-purpose Linux PCs equipped with two

Ethernet NICs, the primary NIC (which can be copper or optical) for connectivity to the Internet, and the secondary NIC, an optical 1GbE or 10GbE card, to connect to an SN16000 switch.

5.2.2 Connectivity

5.2.2.1 Data-plane

Fig. 5.2 shows the data-plane topology of the wide-area CHEETAH experimental network. Three CHEETAH Point of Presences (PoPs) equipped with SN16000 switches are located at MCNC in Research Triangle Park (RTP), NC, Southern Crossroads (SOX)/Southern Light Rail (SLR) in Atlanta, GA, and Oak Ridge National Laboratory (ORNL) in Oak Ridge, TN. The three SN16000 switches are interconnected by long-distance OC-192 SONET circuits.

Each of the three SN16000 switches is equipped with GbE interface cards (each with ten ports), which are used to connect enterprise networks and/or end hosts. Fig. 5.2 shows that currently, four university campuses, University of Virginia (UvA), City University of New York (CUNY), North Carolina State University (NCSU), Georgia Institute of Technology (GaTech) are enterprises connected to the CHEETAH PoPs. The access links are created by provisioning Virtual Local Area Networks (VLANs) or MultiProtocol Label Switch (MPLS) connections across regional networks and the Hybrid Optical and Packet Infrastructure (HOPI) network [81]. A few end hosts, wukong in MCNC, zelda1, zelda2, zelda3 in SOX/SLR, and zelda4 and zelda5 in ORNL, are physically

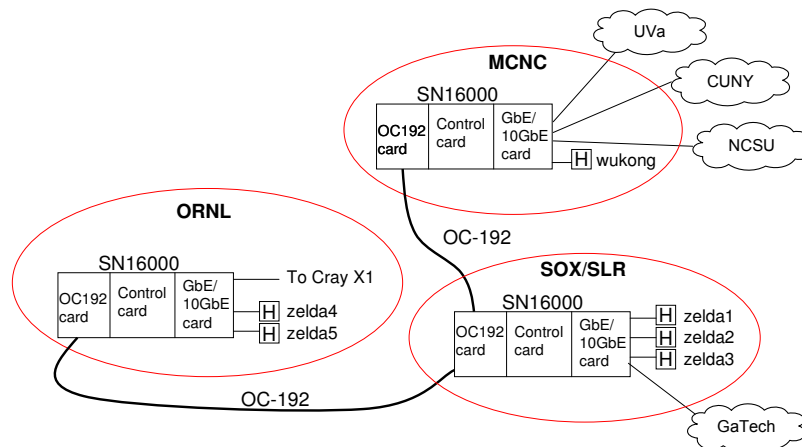


Figure 5.2: Wide-area CHEETAH testbed topology (data-plane).

located at the three CHEETAH PoPs and directly connected to the SN16000 switches for experimental research.

5.2.2.2 Control-plane

The SN16000 switch supports two types of interfaces for transmitting control-plane messages: (i) in-band Data Communication Channels (DCC) embedded within SONET data-plane interfaces, and (ii) out-of-band Ethernet control ports in the control cards of the SN16000s as shown in Fig. 5.2. We choose the out-of-band signaling option in the CHEETAH network for switch-to-switch control-plane communication simply because we needed the ability to capture control-plane messages for experimental reasons. The Ethernet control ports of the switches are connected to the Internet.

Control-plane communication between end hosts and switches are also carried on the Internet through the primary NICs on the end hosts.

5.2.3 Addressing

Given the shortage of IPv4 addresses, enterprises often use private and/or dynamic IP addresses with techniques such as Network Address Translation (NAT) and Dynamic Host Configuration Protocol (DHCP). In this section, we consider the question of whether private and/or dynamic IP addresses can be assigned to the data-plane and control-plane interfaces in GMPLS networks.

5.2.3.1 Data-plane addressing

End hosts in a GMPLS network are comparable to servers on the Internet in that they can be “called” by other clients. A “calling” end host needs to be able to find the IP address of the “called” end host, which implies that the address needs to be *static*. Further, the address needs to be globally unique, which means it has to be *public* if the GMPLS network is to be scalable with subnetworks managed by multiple autonomous-system administrators. As scalability is an important goal of the CHEETAH project, we assign the secondary NICs (data-plane NICs) static public IP addresses. When a process running on a CHEETAH end host, H_1 , wants to initiate a circuit setup to another CHEETAH end host, H_2 , it will look up the domain name of the secondary NIC of host H_2 , find its

static public IP address and use this as the destination address in the *Session* object of the RSVP-TE *Path* message.

In GMPLS networks, the same addressing scheme needs to be used at both ends of a link. For example, both ends need to be “numbered,” or both ends need to be “unnumbered.” A numbered interface is assigned its own IP address. An unnumbered interface is assigned an interface identifier (ID), which is used in conjunction with the single IP address assigned to the node on which the interface is located [82]. As there is no necessity to address data-plane interfaces of switch-to-switch links, we use the unnumbered scheme for these links in the CHEETAH network (e.g., see the link between the SOX/SLR and MCNC SN16000s in Fig. 5.3). However, since the data-plane interfaces of end hosts need to be assigned IP addresses, the corresponding switch interfaces are also numbered (e.g., see the link between the wukong and the MCNC SN16000 in Fig. 5.3). For reasons that will become apparent in the next section, switch interfaces connected to end hosts need static public IP address assignments.

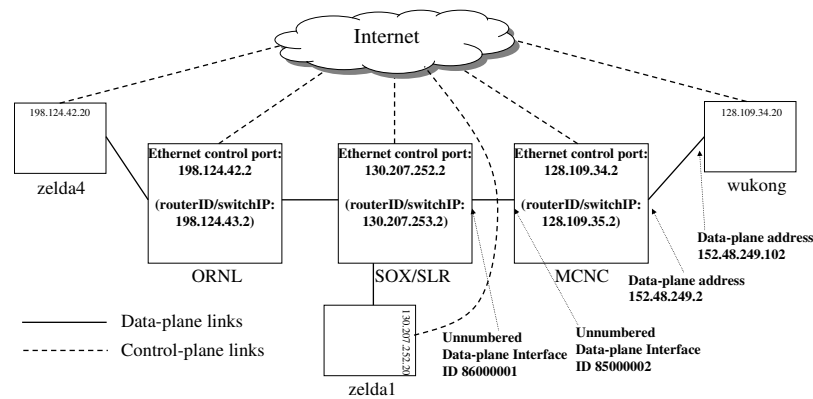


Figure 5.3: IP addressing in the CHEETAH network.

5.2.3.2 Control-plane addressing

As we described in section 5.2.2.2, end hosts send their control-plane messages via their primary NICs. We assign *static public* IP addresses to the control ports of all the end hosts and switches in CHEETAH, as illustrated in Fig. 5.3. The reason that these addresses need to be *static* is that Traffic-Engineered (TE) links need to be configured at the GMPLS switches to provide information

Table 5.1: TE-link configuration at the MCNC Sycamore SN16000 switch.

Link	Local data-plane IP address / interface ID	Remote end data-plane IP address / interface ID	Remote end control-plane address	Capacity
Link to SOX/SLR SN16000	85000002	86000001	130.207.253.2	OC-192
Link to end host wukong	152.48.249.2	152.48.249.102	128.109.34.20	1Gb/s

needed by GMPLS signaling engines. Link configuration includes specification of the data-plane addresses (local and remote) and the remote control-plane address. For example, we show the configuration information for two TE-links at the MCNC SN16000 switch in Table 5.1. The first link is between switches and hence uses unnumbered addressing, i.e., interface IDs, for its data-plane interfaces as described in the previous section. The second link is from the switch to an end host and is hence assigned an IP address for both ends of the data-plane interface. When a call setup request (an RSVP-TE *Path* message) arrives at a switch, it needs the control-plane address of the next-hop switch or control-plane address of the destination host toward which it should route the setup procedure. Given the TE-link configuration process is static, in that it is executed once at initialization time, we need to make the control-plane IP addresses and data-plane numbered IP addresses static. If these addresses are obtained dynamically using DHCP, a dynamic TE-link database update mechanism is needed to inform the neighboring switch or end host. As these procedures are undefined, we use static addresses.

Having understood the reason for requiring control-plane addresses to be *static*, we next examine whether they can be private instead of public. If the GMPLS network under design is a small-scale single-domain network, then control-plane ports can be assigned private addresses. However, given the CHEETAH project goals of creating multiple-domain scalable GMPLS networks, addresses in the control plane will need to be assigned by different administrators. Hence for global uniqueness, these addresses need to be public. The only exception is the control-plane interfaces of internal switches within a single provider's network, which can be assigned static pri-

vate IP addresses. However, we need static public IP addresses for the control ports of all border switches [83].

One final point to note in Fig. 5.3 is that besides needing to assign IP addresses to Ethernet control port(s) in SN16000 switches, a RouterID/SwitchIP address allocation is required for each switch. The RouterID in a GMPLS switch, such as the SN16000, is similar to the RouterID used in IP routers for routing protocol exchanges. The SwitchIP is a similar address used in RSVP-TE signaling messages. Just as the RouterID should be a public static “routable” address in the Internet to allow routing protocol messages to be routed via any of the router’s interfaces, so does the SwitchIP. As recommended by Sycamore, we use a single address for both the RouterID and SwitchIP in the CHEETAH network.

Given the shortage of public IPv4 addresses, we recommend the use of IPv6 to create scalable GMPLS networks. However, due to limitations of the current software implementation in the SN16000, we use IPv4 static public addresses in the CHEETAH network.

5.2.3.3 Handling the effects of our addressing solution

Our addressing solution requires dynamic updates to the IP routing table and ARP table at the end hosts as circuits are established and released. We explain the need for this action below.

IP routing table update: We explain the need for IP routing table updates at the end hosts with an example. Say a CHEETAH circuit is established between an end host at UVa to an end host in the NCSU campus through the CHEETAH network (see Fig. 5.2). Using our recommendation of assigning static public IP addresses (see Section 5.2.3.1), we assign the UVa end host data-plane interface an address from UVa’s 128.143 Class-B address space allocation, and similarly, we assign the NCSU end host data-plane interface an address from NCSU’s 152.1 Class-B address space allocation. Once the circuit is set up, IP datagrams generated by an application running at the UVa end host destined for the NCSU end host must be routed directly on to this Ethernet-SONET-Ethernet circuit. But the default configuration of the IP routing table at the UVa end host will indicate that packets destined to the NCSU host interface address be directed to a default gateway (an IP router) since the NCSU interface address is not in the same subnet as UVa’s interface address.

The routing problem can be solved by adding a host-specific entry to the IP routing table at the two end hosts after the circuit is established. At the UVa end host, the entry would indicate that the NCSU end host interface is directly reachable on the secondary NIC and vice versa. When the circuit is released, the entries at both end hosts should be deleted.

ARP table update: Continuing with our UVa to NCSU example, having updated the IP routing table at the UVa host to indicate that the next-hop IP router to use for the NCSU host interface address is the interface itself, we must now contend with needing the MAC address of the NCSU host interface. The typical solution is to use ARP. However, using ARP, an address resolution mechanism designed for local-area networks, across a wide-area network, has two drawbacks. First, it incurs a round-trip propagation delay, adding to the overhead of circuit setup delay. In the CHEETAH project, one of our goals is to reduce call setup delay to enable the use of circuits for short-duration calls for increased resource sharing. Second, ARP, being a broadcast mechanism, could lead to broadcast storms if there are Ethernet switches on the end-to-end circuits. Ideally each of these switches should be programmed with a port-mapped untagged or tagged VLAN corresponding to the circuit, but in case of configuration errors, a broadcast storm could be triggered across the wide area.

If ARP is to be avoided as a solution to this wide-area address resolution problem, what then is a more suitable technique? Here, we turn to the other ubiquitous address resolution mechanism used in the Internet today, i.e., the DNS. This system is designed for large-scale networks and seems ideal to provide support for global-scale IP address to MAC address translations that will be needed if GMPLS networks create wide-area Ethernet-SONET/WDM-Ethernet circuits. We propose using the TXT resource record type of DNS to store MAC addresses corresponding to domain names along with the A-type resource record, which stores IP addresses corresponding to domain names.

As a domain name lookup is invariably required to find the IP address of the remote host (which could incur a wide-area round-trip delay for infrequently accessed hosts), obtaining the MAC address simultaneously would save the additional round-trip delay incurred if ARP was used after the circuit is established. Thus, we propose using the TXT resource records to store MAC addresses of CHEETAH end hosts in the existing DNS hierarchy.

Once the MAC address is obtained, the signaling software at the end host configures the ARP table adding an explicit entry mapping the IP address of the remote end of the CHEETAH circuit to the obtained MAC address after the circuit is established. When user data starts flowing, an ARP table lookup will directly provide the destination MAC address allowing for a quick Ethernet header/trailer encapsulation of the IP datagram being sent to the distant host.

5.2.4 Control-plane design

In Section 5.2.2.2, we stated that the Internet is used to transport all control-plane messages in CHEETAH. This is accomplished by connecting the primary NICs of end hosts (which are effectively the control-plane ports of end hosts) and the Ethernet ports of the SN16000 control cards to the Internet. In this section, we describe our solutions to two problems that arise in the control-plane design of GMPLS networks.

First, with the use of an out-of-band control-plane network, i.e., the Internet, two GMPLS switches that are data-plane neighbors may not necessarily be control-plane neighbors. In other words, two switches interconnected by a data-plane interface may be interconnected on a path consisting of one or more IP routers on the Internet. This causes problems for the automatic neighbor discovery process in OSPF-TE.

Second, given the security problems faced on the Internet, there is significant risk in using the Internet as the control-plane network of a GMPLS network. Both RSVP-TE and OSPF-TE control-plane protocols are used to carry out critical functions. A malicious user could cause significant harm by sending RSVP-TE messages to tie up bandwidth. Similarly a malicious user could send OSPF-TE messages that create routing loops or other such problems. To prevent such attacks, we need a solution to secure the control-plane ports of a GMPLS network.

We describe our solutions to these two problems below.

5.2.4.1 Enabling OSPF-TE automatic neighbor discovery

Automatic neighbor discovery occurs by the OSPF-TE implementation at a switch broadcasting a *Hello* message to all its neighbors. If two GMPLS switches are data-plane neighbors, we need to

ensure that they are control-plane neighbors as well. We do this by provisioning IP-in-IP tunnels between neighboring GMPLS switches through the Ethernet control ports. A tunnel is viewed as an interface by the switch software. All control-plane messages sent to the broadcast IP address, such as the OSPF-TE *Hello* message, will be routed to all tunnels and interfaces on which OSPF-TE is enabled.

The outer datagram header of the OSPF *Hello* message sent on the IP-in-IP tunnel will carry the SN16000 Ethernet control port IP addresses as source and destination (since these are used in the tunnel-creation commands), and the inner datagram header will carry the sending switch's RouterID and the broadcast IP address as source and destination, respectively. The packets will be routed through the Internet using the outer datagram header and upon reaching the destination, it will be delivered to the OSPF-TE software at the neighboring switch.

When two neighboring SN16000s have exchanged *Hello* messages, the OSPF-TE software in each switch learns the neighbor's RouterID, and automatically adds an IP routing table entry mapping this address to the tunnel interface. Subsequent OSPF-TE messages, such as Link State Updates, which are addressed to the RouterID of a neighbor, will be routed through the IP-in-IP tunnel interface. As noted in Section 5.2.3.2, the SwitchIP address used by RSVP-TE messages is the same as the RouterID. Thus, both OSPF-TE and RSVP-TE messages exchanged between two switches will carry two IP headers when traversing the Internet, the inner one carrying the RouterID/SwitchIP addresses as source and destination (except for the OSPF Hello message, whose inner header will carry the broadcast IP address as destination), and the outer one carrying the Ethernet control-port IP addresses of the switches as source and destination.

As in connectionless IP networks, we do not expect end hosts to run OSPF-TE in the CHEETAH network. Therefore, no IP-in-IP tunnels are required between end hosts and switches for routing protocol messages.

5.2.4.2 Security

Since RSVP-TE and OSPF-TE run directly over IP, we choose IPsec [84] to secure the control plane rather than Secure Shell (SSH) [85], Secure Sockets Layer (SSL) [86], or Transport Layer Security

(TLS) [87].

Open-source IPsec software such as Linux Openswan [88], or Cisco/Juniper IPsec clients can be used at the end hosts. Since the SN16000 switch software does not support IPsec, we use an external security device, the Juniper NS-5XT [89], through which tunnel-mode IPsec is configured. Openswan, which runs on the end hosts, interoperates with IPsec software in the Juniper NS-5XT. As shown in Fig. 5.4, the Ethernet control ports of the three SN16000s are connected to the trusted (secured) interface on an NS-5XT device each.

We establish IPsec tunnels between the NS-5XTs at the two switches configuring them to add an outer IP header with an Encapsulating Security Payload (ESP) header to IP datagrams received from its trusted interface (the link to the switch). The Security Policy Database (SPD) is configured to allow only those packets whose destination and source IP addresses match the Ethernet control-port addresses and RouterID/Switch IP of the two adjacent switches through this IPsec tunnel.

When an OSPF-TE or RSVP-TE message is sent on an IP-in-IP tunnel set up between adjacent switches (as described in the last subsection), the payload is already encapsulated in two IP headers. When the NS-5XT receives these IP-in-IP datagrams, it determines which security association and routing entry, if any, the packet matches (by consulting the security policy database and the routing table), and then applies the security services indicated in the SA database for that association. For these IP packets, NS-5XT encrypts the whole IP packet and adds an outer IP header whose des-

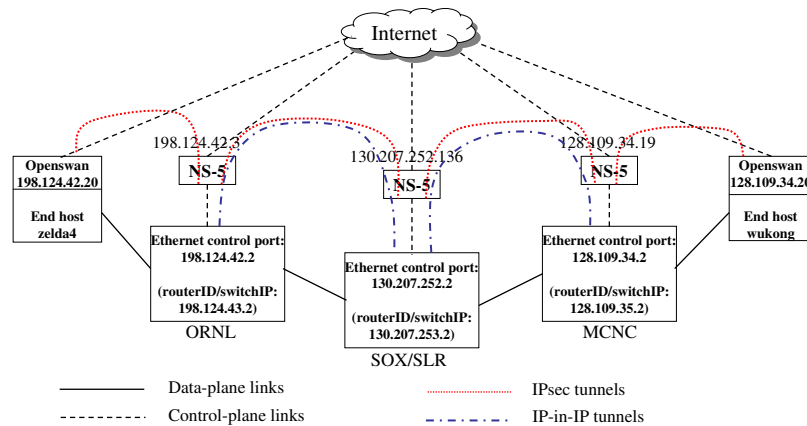


Figure 5.4: Creating IPsec tunnels on the CHEETAH network control plane.

termination and source addresses are the NS-5XT untrusted (Internet) interfaces' IP addresses. Thus, there are two tunnels in the CHEETAH control-plane network between any two SN16000s, an inner IP-to-IP tunnel described in the last subsection, and an outer IPsec tunnel. Similarly we configure IPsec tunnels between Linux end hosts running Openswan and the SN16000 switches.

With the ESP header, IPsec authentication capability is provided in addition to encrypting messages. Each switch authenticates the host from which it receives RSVP-TE messages, and its neighboring switches from which it receives RSVP-TE and OSPF-TE messages. As this authentication is host-based, if an intruder is able to gain access to a CHEETAH end host, then this host can be made to issue RSVP-TE messages indiscriminately to tie up network bandwidth. To solve this problem, we plan to use the RSVP-TE Integrity object [90], which will be implemented in the next release of the SN16000 software, to support user-level authentication.

Given our proposed use of static public IP addresses for scalability reasons, it becomes important to configure firewalls to prevent unwanted access to the control interfaces of the switches and end hosts. The Juniper NS-5XTs have firewall capability, which we use in the CHEETAH network. At the end host, we use the Linux iptables firewall software.

Finally, Denial of Service (DoS) attacks are possible to the RSVP-TE software at the switches. The NS-5XT device provides some DoS protection but it is not extensive. In summary, through the use of Openswan clients on CHEETAH end hosts and Juniper NS-5XT devices, we are able to provide a minimal level of security for the control plane.

5.3 End-Host CHEETAH Software

To support the immediate-request mode of bandwidth sharing in CHEETAH, end hosts need to be equipped with new software to initiate and accept dynamic circuit requests. We describe the design and implementation of this end-host CHEETAH software in this section. The software architecture is shown in Fig. 5.5. We provide a brief overview of CHEETAH software at end hosts, and then focus on the RSVP-TE module. Readers are referred to [91] for more details about the other modules.

The first component of the CHEETAH software shown in the end hosts in Fig. 5.5 is the **Optical Connectivity Service (OCS) client**. The purpose of this module is as follows. As described in Section 5.1.2, CHEETAH is designed as an add-on service to the existing Internet connectivity. This allows a CHEETAH end host to both communicate via the Internet with end hosts that are not connected to the CHEETAH network, as well as communicate with CHEETAH end hosts on dedicated circuits. To allow an end host to determine whether or not the correspondent end host with which it wants to communicate is on the CHEETAH network, we propose using DNS. Again, exploiting the TXT resource record capability of DNS, we propose adding a text string such as “on-cheetah-network” in this resource record. Thus a DNS query for the TXT resource record of a domain name will yield an answer indicating whether the remote host is on the cheetah network. The calling end host performs this lookup via an Optical Connectivity Service (OCS) client module.

The second component of end-host CHEETAH software shown in Fig. 5.5 is the **routing decision** module. Recall the choice of two paths available to CHEETAH end hosts as described in Section 5.1.2. Using measurements collected about the state of the two networks, this module answers queries on which network to use for specific data transfers.

The third component of end-host CHEETAH software shown in Fig. 5.5 is the **RSVP-TE** module. This module is used to format and send RSVP-TE messages requesting the setup or release of circuits. It also manages the bandwidth of the host-to-switch link (the secondary NIC) and configures the IP and ARP tables as described in Section 5.2.3.3.

In addition to the CHEETAH software module, Fig. 5.5 shows that CHEETAH end hosts run

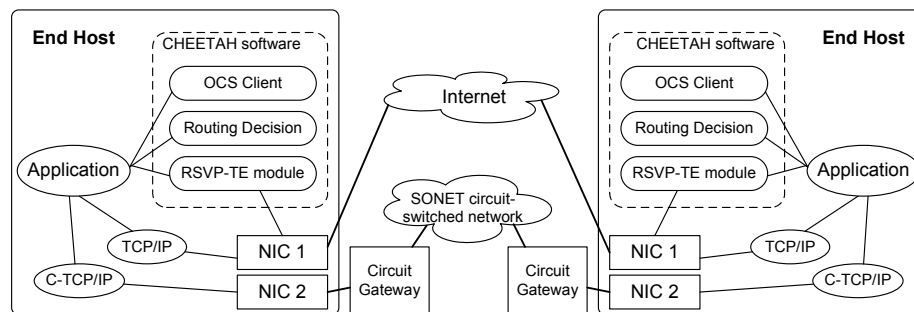


Figure 5.5: End-host CHEETAH software architecture.

a module labeled **Circuit-TCP (C-TCP)/IP** [63]. C-TCP is actually a modified version of the TCP code in the kernel with changes made to match the nature of dedicated end-to-end circuits. Since CHEETAH is built as an add-on service to basic Internet communication, our modifications are triggered only for TCP connections established on CHEETAH circuits through NIC2. For connections passing through the primary NIC, NIC1, standard TCP code will be executed. For details on the C-TCP modifications, see [63].

We will now describe more details about the RSVP-TE module as this is my contribution to the CHEETAH software package.

5.3.1 Functional description of the RSVP-TE module

Recall from Section 5.1.1 that switches perform the following five steps upon receiving a *Path* message: message parsing, route lookup, connection admission control, data-plane configuration, and message construction. Similar functions are needed in the **end-host RSVP-TE module**. The **message parsing and construction** steps are the same on end hosts as in the switches. Let us focus on the remaining three steps.

We treat **route lookup** at an end host in the same manner as is currently in practice with IP packet forwarding, in that end hosts are simply programmed with a default gateway address and all IP packets, destined to nodes outside the local network, are sent to this gateway. Similarly, the RSVP-TE module at the end host sends all *Path* messages to its circuit gateway. The circuit gateway then either performs source route computation to reach the specified destination and adds the computed source route in an Explicit Route Object (ERO) to the *Path* message, or simply determines the next-hop switch toward which to direct the call setup. This mode of operation, in which the end client does not include an ERO, is called the GMPLS overlay model [92], in contrast to the GMPLS peer model where the end client is a router that computes the ERO itself.

Connection admission control (CAC) on end hosts is required because multiple applications running on an end host can independently ask the RSVP-TE module of the end-host CHEETAH software to initiate requests for circuits to remote end hosts. The CAC sub-module of the RSVP-TE module checks to see whether or not the secondary NIC is already tied up in a circuit. In the current

implementation, the secondary NIC can be part of only one circuit. In later implementations, we plan to support VLAN multiplexing, which will allow for multiple circuits (to the same or different destinations) to be supported on the secondary NIC. The CAC sub-module will then need to be upgraded to ensure that the total bandwidth reserved across the multiple circuits does not exceed the NIC2 bandwidth.

The **data-plane configuration** needed at an end host is comparable to that needed at a switch. In the latter, it consists of configuring the switch fabric to properly forward user data from an input interface to a corresponding output interface. An equivalent packet forwarding function at end hosts is to move outgoing data from applications to the NIC and in the reverse direction from the NIC to applications. As described in Section 5.2.3.3, this requires configuring the IP routing table and ARP table.

5.3.2 RSVP-TE module design and implementation

The end-host RSVP-TE module we developed for Linux hosts is based on RSVP-TE KOM [93]/DRAGON VLSR [76] code. We reused the RSVP-TE message parsing and construction functions and added software for CAC and data-plane configuration. As shown in Fig. 5.6, the software is divided into three parts: `rsvpd`, `sig_proc` (signaling procedures), and `bwlib`. `Rsvpd` is a daemon that sends and receives RSVP-TE messages, and maintains RSVP-TE sessions. `Sig_proc` is another daemon, which (i) listens for circuit setup requests from local-host applications, performs connection admission control for the host's secondary NIC, and initiates RSVP-TE signaling if the request

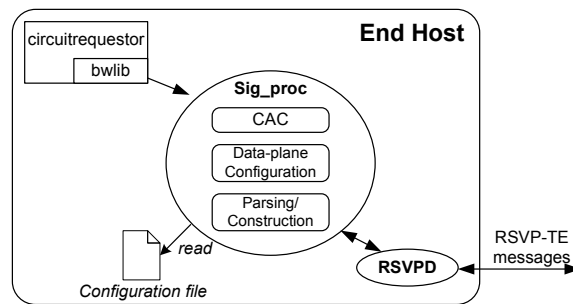


Figure 5.6: RSVP-TE module architecture.

is admitted, (ii) listens for circuit setup requests passed to it by rsvpd for incoming calls and processes them. Sig_proc has a configuration file, which defines the control-plane IP address of the end host, the control-port IP address of the edge switch, and the data-plane link information. Bwlib is a library that is integrated into applications to enable the application to communicate with sig_proc to request the setup and release of circuits. For example, in our end-host RSVP-TE software package, we include an example application integrated with bwlib called circuitrequestor, which allows users to request circuits manually. This tool is useful when application source code is unavailable for integration with bwlib.

Rsvpd and sig_proc are configured to automatically start up when the end host system is booted. When started, sig_proc loads configuration information from the configuration file, binds itself to a specific port, and listens for circuit-related requests from local applications through bwlib. It also registers a default session with rsvpd to handle *Path* messages for calls initiated by remote end hosts.

On the sender side, when an application needs a circuit, bwlib sends a request, with destination domain name and desired bandwidth, to sig_proc. Sig_proc then performs CAC and if successful, it forks a child process to handle this request. The newly-forked process initiates circuit setup by constructing a *Path* message and asking the rsvpd daemon to send it to the edge switch. When the sender host receives the *Resv* message confirming successful circuit setup, sig_proc updates the IP routing table and ARP table to add corresponding entries for the receiver.

On the receiver side, when sig_proc receives a *Path* message from rsvpd, it forks a child process to handle the request. The newly-forked process creates a new session in rsvpd for this request and performs the following steps: (i) updates the CAC table to indicate that the data-plane NIC is now reserved, (ii) sends back a *Resv* message to the edge switch, (iii) updates the IP routing table and ARP table by adding entries for the far-end host, and (iv) handles follow-up messages for this session.

5.4 Call Setup Delay Measurements

Given that dynamic bandwidth sharing in connection-oriented switches is controlled by the signaling engine, the request handling performance of the signaling engine is critical to the scaling of networks. The faster the response times of signaling engines, the lower the cost to an application to release and re-acquire bandwidth as and when needed, and the higher the link utilization. Therefore we conducted experiments to measure circuit setup delays.

5.4.1 Experimental setup

Measurements are taken for the circuit setup procedure between host `zelda1` and host `wukong` (see Fig. 5.2). Both `zelda1` and `wukong` are Dell PowerEdge servers running the Red Hat Linux operating system. As seen from Fig. 5.2, `zelda1` is connected to the SOX/SLR SN16000 (`sn16k-atl`) and `wukong` is connected to the MCNC SN16000 (`sn16k-nc`). The two switches are interconnected by a wide-area OC-192 circuit. The RSVP-TE module on `zelda1` initiates the call. For this circuit, RSVP-TE *Path* message processing is required at the `sn16k-atl` and `sn16k-nc` switches, and at `wukong`, as illustrated in Fig. 5.1. The primary NICs at the hosts on which signaling messages are carried and the SN16000 control ports are 100Mb/s Ethernet.

Measurements are obtained for three types of circuits: OC-1 SONET circuit, OC-3 SONET circuit, and 1Gb/s Ethernet-over-SONET (EoS) circuit. The experiment was repeated ten times for each circuit type, and the average values are reported. The variance was very small since the Internet path is lightly loaded.

Performance data collected includes end-to-end circuit setup delay and per-switch signaling message processing delay. **Circuit setup delay** is defined as the time interval from the instant at which an end host initiates a circuit setup to the instant at which it receives a circuit setup success confirmation from the network. In the experiment, we recorded the time difference between the instant that function `bwrequest` (the function in the `bwlib` library that is invoked by applications to request circuits) is called and the time that the function returns, which indicates that the circuit was set up successfully.

There are two types of **per-switch signaling message processing delays**. The first is “per-switch *Path* message processing delay,” which is defined as the time interval from the instant at which an SN16000 receives a *Path* message from the previous hop to the instant at which it sends out a *Path* message to the next hop. Similarly we define “per-switch *Resv* message processing delay” as the time interval from the instant at which an SN16000 switch receives a *Resv* message to the instant at which it sends out a *Resv* message. To obtain per-switch signaling processing delays, we captured RSVP-TE messages using Ethereal [94] and recorded the time stamps of incoming and outgoing messages.

Besides per-switch signaling message processing delays, the end-to-end circuit setup delay consists of other two components: processing delays incurred at the two end hosts, and propagation plus emission delay of signaling messages. We also measured these delays.

5.4.2 Experimental results and analysis

Table 5.2 shows measured data for different circuit setup procedures between host zelda1 and wukong. The first column shows the circuit type. The second column shows the end-to-end circuit setup delay. The third and the fourth columns show the per-switch signaling message processing delays observed at the sn16k-nc for the *Path* and *Resv* messages, respectively. Finally, we shown the round-trip signaling message propagation plus emission delays.

Table 5.2: Signaling delays incurred in setting up a circuit between zelda1 and wukong across the CHEETAH network

Circuit type	End-to-end circuit setup delay (s)	Processing delay for <i>Path</i> messages(s) at the sn16k-nc	Processing delay for <i>Resv</i> messages(s) at the sn16k-nc
OC-1	0.166103	0.091119	0.008689
OC-3	0.165450	0.090852	0.008650
1Gb/s EoS	1.645673	4.907113	0.008697
Round-trip signaling message propagation plus emission delay between sn16k-atl and sn16k-nc: 0.025s			

We can see that end-to-end circuit setup delays for an OC-1 circuit and an OC-3 circuit are both

around 166ms. The reason that these two delay values are similar is that the procedures for setting up these two pure-SONET circuits are the same.

On the other hand, the end-to-end circuit setup delay for a 1Gb/s EoS circuit is much higher (around 1.65s). This is because 7 OC-3 circuits are established one after another between the two switches before the Ethernet segments are mapped to this 7-OC3 circuit. It is purely an artifact of the current release of the SN16000 software whereby virtually concatenated signals, while supported in the data plane are not yet supported in the control-plane. We expect to receive a software upgrade that would reduce EoS circuit setup delays to the same order of magnitude as pure-SONET circuits.

The *Path* message processing delay measured at the sn16k-nc for the 1Gb/s EoS circuit can be validated by summing 14 per-switch *Path* message processing delays (one at each switch for each OC-3), 7 times round-trip propagation plus emission delays, and 14 per-switch *Resv* message processing delays (one at each switch for each OC-3). An approximation of *Path* message processing delay for a 1Gb/s EoS circuit is $14 \times (0.091 + 0.0087) + 7 \times 0.025 = 1.57s$. Roughly speaking, this explains the 1.65s *Path* message processing delay seen at the sn16k-nc for the 1Gb/s EoS circuit.

We use these measured end-to-end call setup delays in our analytical/simulation models of file transfers on the CHEETAH network. For example, in [58], where we originally proposed the CHEETAH concept, we used analytical models to determine the conditions under which a dedicated circuit offered a delay advantage over a TCP/IP path for a file transfer. In that study, we used a low call setup processing delay of $4\mu s$ based on our hardware-accelerated implementation of RSVP-TE [95]. This led us to draw conclusions that for TCP connections across a WAN, e.g., connections with a round-trip time (RTT) of 50ms, if the bottleneck link rate is the same as the circuit rate, then from a delay comparison perspective, it is always appropriate to attempt a circuit setup because if resources are available and setup succeeds, the file transfer will incur a much shorter delay. However, from a utilization perspective, file transfer delay should be much larger than call setup delay. If we assume a factor of 10, then for a 100Mbps circuit across a 50ms-RTT path, we concluded that a circuit setup attempt is warranted for files over 6MB if we assume our hardware-accelerated signaling implementation. On the other hand, with our newly measured value of 1.65 seconds for call setup delay across a network of off-the-shelf switches, this minimum

file size increases to 200MB. These measurements are useful for implementations of the routing-decision module.

5.5 Conclusions

We implemented the immediate-request mode of bandwidth sharing in the CHEETAH testbed, which is based on the GMPLS architecture. The wide-area CHEETAH network deployment consists of three Sycamore SN16000 switches interconnected by wide-area OC192 circuits, and Linux end hosts connected to the SN16000s via GbE and 10GbE links. End hosts are equipped with second NICs for this connectivity, thus leaving intact their primary NICs for Internet access. Sycamore SN16000 switches have built-in GMPLS control-plane software to support the immediate-request mode. We developed an end-host CHEETAH software package for Linux end host users to initiate circuit setup requests, and a CHEETAH API for usage by application programmers. Applications running on end hosts, such as file-transfer applications, can be upgraded to include the CHEETAH API for a transparent invocation of the CHEETAH service, when appropriate. The end-host CHEETAH software package includes an RSVP-TE module to initiate the setup and release of end-to-end circuits on behalf of end-host applications. Wide-area signaling experiments conducted on the CHEETAH network show that the end-to-end circuit setup delay across a 25ms round-trip propagation-delay path that traverses two SONET switches is around 166ms.

To create scalable multi-domain GMPLS networks, we recommend the use of an Internet-complementary network architecture, the use of IPv6 static public IP addresses for GMPLS interfaces, the use of the Internet as the control-plane network, the use of IPsec and RSVP-TE Integrity object to secure the control-plane, the use of dynamic updates of IP routing and ARP tables at end hosts as circuits are established and released, and the use of the DNS TXT record to store MAC addresses and an indicator of GMPLS-network connectivity for end hosts.

Chapter 6

Conclusions and Future Work

6.1 An Integrated View

In this section, we provide an integrated view of the bandwidth-sharing mechanisms described in the main chapters of this dissertation, starting with existing mechanisms that are in use today.

Communication services offered on networks today are of three types: (i) IP services, (ii) telephone services, and (iii) leased-line services. *IP services* are offered on connectionless, packet-switched networks. In these networks, a user cannot reserve bandwidth for any single flow; instead the effective data rate of any single transfer is affected by the data rates of other ongoing transfers on shared links of its end-to-end path. *Telephone services* are offered in connection-oriented, circuit-switched networks. The bandwidth-sharing mechanism in these networks is an immediate-request call blocking mode. *Leased-line services* are offered on high-speed connection-oriented networks. Sharing of bandwidth for these services is primarily along spatial lines, with limited temporal sharing. For example, on an OC192 link, a provider may lease out 48 OC3 circuits, but these circuits are held for long durations (on the order of years), effectively eliminating temporal sharing.

The goal of this research was to design bandwidth-sharing techniques that result in an efficient utilization of high-speed connection-oriented networks. Of the three services offered in existing networks, as listed above, only the latter two are offered on connection-oriented networks. The leased-line service clearly results in poor utilization owing to its lack of temporal sharing, making it an inefficient service. The immediate-request call blocking mode of bandwidth sharing, used to

offer telephone services, can be used in high-speed connection-oriented networks. We concluded that to achieve efficient utilization with a low call blocking probability, the per-call bandwidth required by applications has to be a small-to-moderate fraction of the link capacity. For example, we showed that if this ratio is on the order of 10, then to achieve 80% utilization, call blocking probability will be 23.6%. We observe that there are applications needing a high fraction of link capacity. For example, participants in large scientific projects require the transfers of very large files (e.g., terabyte-sized files). These transfers require high per-call bandwidth to decrease transfer delays. In today's environment, computer clusters with disk arrays can source and sink data at multiple *Gbps* rates. However, the highest-rate data links deployed today are typically 10Gbps. This makes the per-call-bandwidth-to-link-capacity ratio less than 10. This motivated us to design new bandwidth-sharing mechanisms for high-speed connection-oriented networks.

We designed two new bandwidth-sharing mechanisms, both of which are advance-reservation mechanisms: VBDS for data-type requests and BA-*n*/BA-First for session-type requests. Data-type requests are designed for file transfers. We proposed the VBDS technique, which allows circuit-switched networks to perform as well as packet-switched networks for moderate-to-large-sized files, thus overcoming the problem with fixed bandwidth allocation that once a circuit is setup, the file transfer cannot take advantage of bandwidth freed as other transfers complete. Thus VBDS achieves high utilization while simultaneously offering users low delay. For example, in a scenario we simulated (in which the link is divided into 4 channels with a per-channel rate of 1*Gbps*, file transfer requests specify maximum bandwidth of 1, 2, and 4 channels with corresponding probabilities of 0.3, 0.3, and 0.4, respectively, and file sizes are distributed according a bounded Pareto distribution with shape parameter $\alpha = 1.1$, lower bound $k = 500MB$ and upper bound $p = 100GB$), 90% utilization was achieved with a mean normalized delay of 490 seconds for calls specifying a maximum rate of 2 channels. Contrast this with 1500 seconds mean normalized delay with FBDS in the same scenario.

The BA-*n*/BA-First mechanisms are designed for session-type requests, in which users specify a fixed bandwidth and duration. By allowing users to make advance reservations, which means a network scheduler is deployed to manage resources over an advance-reservation window of time,

efficient utilization can be achieved with a low attendant call blocking probability even when the per-call bandwidth is a large fraction of link capacity. For example, in a network in which per-call bandwidth is $(1/10th)$ of the link capacity, 95% utilization can be achieved with a call-blocking probability of only 1% if the bandwidth is shared in the BA-First mode. Contrast these numbers with those mentioned above for the IR scheme.

The reason that we need both BA-n/BA-First and VBDS book-ahead bandwidth-sharing mechanisms is as follows. In BA-n/BA-First, a fixed bandwidth level is indicated, which means that if a file-transfer user divides file size by bandwidth to determine duration, and then sends a BA-First request with a bandwidth and duration indication, the scheduler will return an FBDS schedule. As we just discussed, mean delay will be 100% or more than if VBDS is used at the same utilization. Thus for file transfers, VBDS is the better BA scheme. On the other hand, for applications such as remote visualization and remote instrument control, in which scientists synchronously interact with a remote computer or instrument, scientists will need the flexibility to indicate acceptable call initiation times. Also, for these applications, they need to specify a certain bandwidth level and duration. The VBDS scheme is unsuitable for such applications.

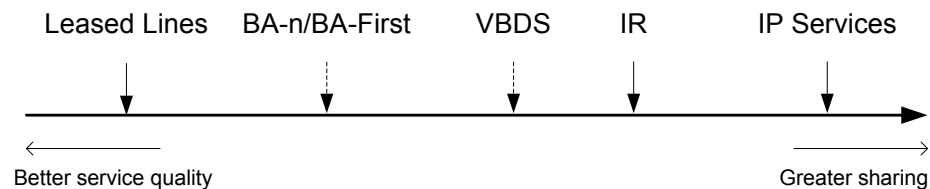


Figure 6.1: Different bandwidth-sharing mechanisms.

We summarize the above discussion using a pictorial representation in Figure 6.1. At the ends of this spectrum are IP services and leased-line services. IP services offer an extremely “fine-grained” bandwidth sharing mechanism in which efficient utilization is the main goal. There is no attendant requirement on user-performance metrics such as bandwidth. At the other end of the spectrum is leased-line services in which the sole focus is on user-performance metrics such as bandwidth, with an attendant degradation of utilization. The immediate-request call blocking mode provides efficient utilization with an attendant low call blocking probability only if the number of channels into which link capacity is divided is moderate-to-large. These three services, IP, leased lines, and IR,

are existing services. The two new schemes we proposed are VBDS and BA-n/BA-First. With these schemes, new applications that require the sharing of a link with just a few channels (on the order of 10) for longer durations (shorter than leased-line durations but longer than IR-call durations), can be supported with efficient utilization along with low mean delays (file transfers) or low call blocking probabilities (session-type requests). The exact number of channels and value of duration that warrant a BA scheme depend on the desired utilization and mean delay/call blocking probability. Our analytical and simulation models are essentially tools to allow a network administrator to make such determinations.

We conclude that high-speed connection-oriented networks should support a combination of IR, VBDS, BA-n/BA-First and leased-line services. IR is required to support applications, such as video-telephony or transfers of moderate-sized files. For the latter, if the circuit rate matches the maximum rate that can be supported as determined by disk and other limitations of general-purpose web servers, the fixed bandwidth allocation disadvantage of connection-oriented networks when compared to packet-switched networks is eliminated. Allowing users to request bandwidth on-demand, as-needed, is an attractive characteristic of this mode. However, with this mode, users can only request a small-to-moderate fraction of a link's capacity. VBDS is suitable for large file transfers, which will experience a significant reduction in delay with high-bandwidth allocations. BA-n/BA-First is needed to support applications that need a high bandwidth allocation for fixed durations without a degradation in call blocking probability. Finally, leased-line services will be needed for their role in serving as "wires" between switches to create networks that offer any or all of the other services shown in Fig. 6.1 (such as IP, IR, VBDS and BA-n/BA-First).

As a complement to the analytical and simulation studies used in the design of our two proposed BA schemes, we undertake an implementation and experimental study of the IR bandwidth-sharing mode in high-speed connection-oriented networks. The state-of-the-art in industry to support this mode of bandwidth sharing in high-speed optical networks is as follows. Control-plane protocols have been standardized by the IETF, and vendors have implemented these protocols in high-speed optical circuit switches. What had not yet been demonstrated when we started this work in 2004 is a network implementation using these control-plane-equipped circuit switches. Also, there

was no implementation of a signaling protocol client for use in end hosts to enable the creation of end-to-end high-speed circuits. The scientific community needed such circuits between cluster computers and/or storage depots. To fill these gaps, we deployed an experimental wide-area connection-oriented network called CHEETAH, and designed and implemented a software package to run on Linux end hosts connected to the CHEETAH network. Among other functions, this software package included a signaling module to enable end users and applications to dynamically initiate requests for dedicated end-to-end circuits for immediate use, and receive/respond to requests when initiated by remote end hosts. We ran experiments of IR mode call setups and releases and used these experiments to obtain measurements for end-to-end circuit setup delays as well as per-switch signaling message processing delays. This work demonstrated the readiness of off-the-shelf switches for actual service offerings, as is now available on Internet2.

6.2 Future Work

This work can be extended in the following directions:

1. Extend our bandwidth-sharing mechanisms to networks in which multiple route options are considered in the reservation phase. In our study of the performance of bandwidth-sharing mechanisms in multi-link scenarios, we assumed a linear topology. In real-world networks, topologies are more complex, and the availability of multiple route options should be exploited. Theoretical studies often separate routing and multiplexing problems; however, the interaction between these two aspects is clearly important in real-world networks.

2. Our bandwidth-sharing mechanisms were primarily designed for single-link systems. Our implementation is a centralized scheduler. However, such a scheduler is impractical as service providers do not share network topology information with each other. Therefore, to support these bandwidth-sharing mechanisms across networks in multiple administrator domains, we need a distributed implementation of these mechanisms.

3. As discussed in Section 2.4.2, we used expert intuition to validate our analytical and simulation models because our bandwidth-sharing mechanisms are in the first stage of their life-cycles,

i.e., the initial design phase. As networks, such as Internet2, ESnet4, deploy switches and software to offer services based on these bandwidth-sharing modes, these models should be validated against real-system measurements as and when they become available. This is a long-term future-work item as it will take a few years to build the necessary user base on these networks.

Bibliography

- [1] Internet2 network map. [Online]. Available: <http://www.internet2.edu/pubs/networkmap.pdf>
- [2] E. Mannie, "Generalized multi-Protocol label switching (GMPLS) architecture," IETF RFC 3945, Oct. 2004.
- [3] L. Berger, "Generalized multi-protocol label switching (GMPLS) signaling - resource reservation protocol-traffic engineering (RSVP-TE) extensions," IETF RFC 3473, Jan. 2003.
- [4] D. Katz, K. Kompella, and D. Yeung, "Traffic engineering (TE) extensions to OSPF version 2," IETF RFC 3630, Sep. 2003.
- [5] M. Veeraraghavan, X. Fang, and X. Zheng, "On the suitability of applications for GMPLS networks," in *Proc. of Globecom 2006*, San Francisco, CA, Nov. 2006.
- [6] LHC - the Large Hadron Collider. [Online]. Available: <http://lhc.web.cern.ch/lhc/>
- [7] A. Naik, H. J. Siegel, and E. K. P. Chong, "Dynamic resource allocation for classes of prioritized session and data requests in preemptive heterogeneous networks," in *Proc. of the 2001 International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, Jun. 2001.
- [8] M. Schwartz, *Telecommunication networks: protocols, modeling and analysis*. Boston, MA: Addison-Wesley, 1986.
- [9] P. Hagelin, U. Krishnamoorthy, J. Heritage, and O. Solgaard, "Scalable optical cross-connect switch using micromachined mirrors," *IEEE Photonics Technology Letters*, vol. 12, pp. 882–884, Jan. 2000.

- [10] B. Pesach, G. Bartal, E. Refaeli, A. J. Agranat, J. Krupnik, and D. Sadot, "Free-space optical cross-connect switch by use of electroholography," *Applied Optics*, vol. 39, pp. 746–758, Feb. 2000.
- [11] Background information on the Terascale Supernova Initiative (TSI). [Online]. Available: <http://www.phy.ornl.gov/tsi/pages/background.html>
- [12] Internet2 dynamic circuit network. [Online]. Available: <http://www.internet2.edu/network/dc/>
- [13] ESnet. [Online]. Available: <http://www.es.net/>
- [14] CANARIE's CA*net 4. [Online]. Available: <http://www.canarie.ca/canet4/index.html>
- [15] UKLight. [Online]. Available: <http://www.uklight.ac.uk/>
- [16] SURFnet. [Online]. Available: <http://www.surfnet.nl/info/en/home.jsp>
- [17] Japan Gigabit Network 2. [Online]. Available: <http://www.jgn.nict.go.jp/english/index.html>
- [18] ESnet On-demand Secure Circuits and Advance Reservation System (OSCARS). [Online]. Available: <http://www.es.net/oscars/index.html>
- [19] X. Zheng, A. P. Mudambi, and M. Veeraraghavan, "FRTP: Fixed rate transport protocol - a modified version of SABUL for end-to-end circuits," in *Pathnets Workshop, held in conjunction with Broadnets 2004*, San Jose, CA, Oct. 2004.
- [20] X. Zhu and M. Veeraraghavan, "Analysis and design of book-ahead bandwidth-sharing mechanisms," *Accepted by IEEE Transactions on Communications*, 2007.
- [21] X. Zhu, M. E. McGinley, T. Li, and M. Veeraraghavan, "An analytical model for a book-ahead bandwidth scheduler," in *Proc. of IEEE Globecom 2007*, Washington, DC, Nov. 2007.
- [22] X. Zhu, X. Zheng, and M. Veeraraghavan, "Experiences in implementing an experimental wide-area GMPLS network," *IEEE Journal on Selected Areas in Communication*, vol. 25, pp. 82–92, Apr. 2007.

- [23] X. Zhu, X. Zheng, M. Veeraraghavan, Z. Li, Q. Song, I. Habib, and N. S. V. Rao, "Implementation of a GMPLS-based network with end host initiated signaling," in *Proc. of IEEE ICC 2006*, Istanbul, Turkey, Jun. 2006.
- [24] L. C. Wolf and R. Steinmetz, "Concepts for resource reservation in advance," *Multimedia Tools Appl.*, vol. 4, no. 3, pp. 255–278, 1997.
- [25] D. Ferrari, A. Gupta, and G. Ventre, "Distributed advance reservation of real-time connections," in *Proc. of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, NH, Apr. 1995.
- [26] L. Yuan, C. Tham, and A. L. Ananda, "A probing approach for effective distributed resource reservation," in *Second International Workshop on Quality of Service in Multiservice IP Networks*, Milano, Italy, Feb. 2003.
- [27] A. G. Greenberg, R. Srikant, and W. Whitt, "Resource sharing for book-ahead and instantaneous-request calls," *IEEE/ACM Transactions on Networking*, vol. 7, no. 1, pp. 10–22, 1999.
- [28] E. G. Coffman, P. Jelenkovic, and B. Poonen, "Reservation probabilities," *Advances in Performance Analysis*, vol. 2, no. 2, pp. 129–158, 1998.
- [29] J. T. Virtamo, "A model of reservation systems," *IEEE Transactions on Communications*, vol. 40, pp. 109–118, 1992.
- [30] O. Schelén and S. Pink, "Resource sharing in advance reservation agents," *Journal of High Speed Networks*, vol. 7, 1998.
- [31] C. Bouras and K. Stamos, "An adaptive admission control algorithm for bandwidth brokers," in *3rd IEEE International Symposium on Network Computing and Applications (NCA04)*, Cambridge, MA, Aug. 2004.
- [32] R. Guérin and A. Orda, "Networks with advance reservations: The routing perspective," in *Proc. of IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000.

- [33] L.-O. Burchard, "On the performance of computer networks with advance reservation mechanisms," in *Proc. of the 11th International Conference on Networks*, Sydney, Australia, Sep. 2003.
- [34] V. Pason and S. Floyd, "Wide area traffic: The failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, Jun. 1995.
- [35] K. S. Trivedi, *Probability and statistics with reliability, queuing, and computer science applications*. Hoboken, NJ: Wiley, 2001.
- [36] R. Yates and D. Goodman, *Probability and Stochastic Processes*. Hoboken, NJ: Wiley, 2004.
- [37] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. New York, NY: Wiley-Interscience, 1991.
- [38] D. K. Pace and J. Sheehan, "Subject matter expert (SME)/peer use in M&S V&V," in *Proc. of the Foundations*, Laurel, MD, Oct. 2002.
- [39] D. Manfield and T. Downs, "On the one-moment analysis of telephone traffic networks," *IEEE Transactions on Communication*, vol. 27, pp. 1169–1174, Aug. 1979.
- [40] J. Heidemann, K. Mills, and S. Kumar, "Expanding confidence in network simulation," *IEEE Network Magazine*, vol. 15, no. 5, pp. 58–63, Sep./Oct. 2001.
- [41] D. P. Bertsekas and R. G. Gallager, *Data Networks*. Englewood Cliffs, N.J.: Prentice Hall, 1992.
- [42] L. Massoulié and J. Roberts, "Bandwidth sharing: Objectives and algorithms," in *Proc. of IEEE Infocom*, New York, NY, Mar. 1999.
- [43] A. Tang, J. Wang, and S. H. Low, "Is fair allocation always inefficient," in *Proc. of IEEE Infocom*, Hong Kong, Mar. 2004.
- [44] R. G. Gallager, *Discrete Stochastic Processes*. Dutch: Kluwer Academic Publishers, 1996.

- [45] H. B. Newman, M. H. Ellisman, and J. A. Orcutt, "Data-intensive e-science frontier research," *Communications of ACM*, vol. 46, pp. 68–77, Nov. 2003.
- [46] H. Lee, "Scheduling file transfers on a circuit-switched network," in *Ph.D. Dissertation*, New York, NY, May 2004.
- [47] M. Veeraraghavan, H. Lee, E. K. P. Chong, and H. Li, "A varying-bandwidth list scheduling heuristic for file transfers," in *Proc. of IEEE ICC2004*, Paris, France, Jun. 2004.
- [48] R. Grobler, "Signaling and scheduling for efficient bulk data transfer in circuit-switched networks," in *Ph.D. Dissertation*, New York, NY, Aug. 2000.
- [49] E. G. Coffman, M. R. Garey, D. S. Johnson, and A. S. Lapaugh, "Scheduling file transfers," *SIAM Journal of Computing*, vol. 14, pp. 744–780, Aug. 1985.
- [50] T. Erlebach and K. Jansen, "Off-line and on-line call scheduling in stars and trees," in *Proc. of the 23rd International workshop on graph-theoretic concepts in computer science*, Berlin, Germany, Jun. 1997.
- [51] S. Martello and D. Vigo, "Exact solution of the two-dimensional finite bin packing problem," *Management Science*, vol. 44, pp. 388–399, 1998.
- [52] E. E. Bischoff and M. D. Marriott, "A comparative evaluation of heuristics for container loading," *European Journal of Operations Research*, vol. 44, pp. 267–276, 1990.
- [53] M. Gehring, K. Menscher, and M. Meyer, "A computer based heuristic for packing pooled shipment containers," *European Journal of Operations Research*, vol. 44, pp. 277–288, 1990.
- [54] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," *Operations Research*, vol. 48, pp. 256–267, 2000.
- [55] M. Pinedo, *Scheduling: Theory, algorithms and systems*. Upper Saddle River, NJ: Prentice-Hall, 1995.

- [56] W. Feng and P. Tinnakornsrisuphap, "The failure of TCP in high-performance computational grids," SC00: High-Performance Networking and Computing Conference, pp. 37–48, 2000.
- [57] —, "The adverse impact of the TCP congestion-control mechanism in heterogeneous computing systems," in *Proc. of the International Conference on Parallel Processing*, Toronto, Canada, Aug. 2000.
- [58] M. Veeraraghavan, X. Zheng, H. Lee, M. Gardner, and W. Feng, "CHEETAH: Circuit-switched high-speed end-to-end transport architecture," in *Proc. of Opticomm 2003*, Dallas, TX, Oct. 2003.
- [59] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," in *Proc. of IEEE Infocom*, Hong Kong, Mar. 2004.
- [60] S. Floyd, "HighSpeed TCP for large congestion windows," Feb. 2003. [Online]. Available: <http://www.icir.org/floyd/hstcp.html>
- [61] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *Computer Communication Review* 32(2), Apr. 2003.
- [62] C. Jin, D. X. Wei, and S. H. Low, "Fast TCP: Motivation, architecture, algorithms, performance," in *Proceedings of IEEE INFOCOM*, Mar. 2004.
- [63] A. P. Mudambi, X. Zheng, and M. Veeraraghavan, "A transport protocol for dedicated end-to-end circuits," in *Proc. of IEEE ICC 2006*, Istanbul, Turkey, Jun. 2006.
- [64] M. Veeraraghavan, X. Zheng, W. Feng, H. Lee, E. Chong, and H. Li, "Scheduling and transport for file transfers on high-speed optical circuits," in *Second International Workshop on Protocols for Fast Long-Distance Networks (PFLDN2004)*, Chicago, IL, Feb. 2004.
- [65] J. Postel and J. Reynolds, "File transfer protocol (FTP)," IETF RFC 959, Oct. 1985.
- [66] W. Allcock, J. Bresnahan, R. Kettimuthu, and M. Link, "The adverse impact of the TCP congestion-control mechanism in heterogeneous computing systems," in *Proc. of the ACM/IEEE Supercomputing 2005*, Seattle, WA, Nov. 2005.

- [67] M. Crovella, M. Harchol-Balter, and C. D. Murta, "Task assignment in a distributed system: Improving performance by unbalancing load (extended abstract)," in *Measurement and Modeling of Computer Systems*, 1998, pp. 268–269.
- [68] H. Wang, R. Karri, M. Veeraraghavan, and T. Li, "Hardware-accelerated implementation of the RSVP-TE signaling protocol," in *Proc. of IEEE ICC2004*, Paris, France, 2004.
- [69] vsftpd. [Online]. Available: <http://vsftpd.beasts.org/>
- [70] J. Lang, "Link management protocol (LMP)," IETF RFC 4204, Oct. 2005.
- [71] B. Metcalfe, "Metcalfe's Law: A network becomes more valuable as it reaches more users," *Infoworld*, no. 17, Oct. 1995.
- [72] P. Newman, T. Lyon, and G. Minshall, "Flow labelled IP: A connectionless approach to ATM," in *Proc. of IEEE Infocom*, Jun. 1996, pp. 1251–1260.
- [73] P. Molinero-Fernandez and N. Mckeown, "TCP switching: Exposing circuits to IP," *IEEE Micro*, vol. 22, no. 1, 2002.
- [74] OMNInet. [Online]. Available: <http://www.icaire.org/omninet/>
- [75] N. S. V. Rao, W. R. Wing, S. M. Carter, and Q. Wu, "Ultrascale net: Network testbed for large-scale science applications," *IEEE Commun. Mag.*, vol. 43, no. 11, pp. 12–17, Nov. 2005.
- [76] Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON). [Online]. Available: <http://dragon.east.isi.edu/>
- [77] UltraLight: An ultrascale information system for data intensive research. [Online]. Available: <http://ultralight.caltech.edu/>
- [78] User-controlled lightpaths (UCLP). [Online]. Available: <http://www.canarie.ca/canet4/uclp/>
- [79] *Generic Framing Procedure (GFP)*, ITU-T Std. G.7041/Y.1303, Dec. 2001.

- [80] *Synchronous Optical Network (SONET) - Payload Mappings*, American National Standards Institute Std. T1.105.02, 1998.
- [81] The hybrid optical and packet infrastructure project (HOPI). [Online]. Available: <http://networks.internet2.edu/hopi/>
- [82] K. Kompella and Y. Rekhter, "Signalling unnumbered links in resource reservation protocol - traffic engineering (RSVP-TE)," IETF RFC 3477, Jan. 2003.
- [83] M. Veeraraghavan, X. Zheng, and X. Zhu. Addressing and secure control-plane network design in GMPLS networks. [Online]. Available: <http://cheetah.cs.virginia.edu/documents/dcn/dcn-design.pdf>
- [84] S. Kent and R. Atkinson, "Security architecture for the internet protocol," IETF RFC 2401, Nov. 1998.
- [85] T. Ylonen and C. Lonvick, "The secure shell (SSH) protocol architecture," IETF RFC 4251, Jan. 2006.
- [86] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL protocol version 3.0. [Online]. Available: <http://wp.netscape.com/eng/ssl3/draft302.txt>
- [87] T. Dierks and C. Allen, "The TLS protocol version 1.0," IETF RFC 2246, Jan. 1999.
- [88] Openswan. [Online]. Available: <http://www.openswan.org/>
- [89] Netscreen-5xt. [Online]. Available: {http://www.juniper.net/products/integrated/ns_5series.html}
- [90] F. Baker, B. Lindell, and M. Talwar, "RSVP cryptographic authentication," IETF RFC 2747, Jan. 2000.
- [91] X. Zheng and M. Veeraraghavan. CHEETAH end-host software design document. [Online]. Available: <http://cheetah.cs.virginia.edu/software/software-document.pdf>

- [92] G. Swallow, J. Drake, H. Ishimatsu, and Y. Rekhter, "Generalize Multiprotocol Label Switching (GMPLS) User-Network Interface (UNI): Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Support for the Overlay Model," IETF RFC 4208, Oct. 2005.
- [93] R. Steinmetz. KOM RSVP engine. [Online]. Available: <http://www.kom.tu-darmstadt.de/en/downloads/software/kom-rsvp-engine/>
- [94] Ethereal. [Online]. Available: <http://www.ethereal.com/>
- [95] H. Wang, M. Veeraraghavan, R. Karri, and T. Li, "Design of a high-performance RSVP-TE signaling hardware accelerator," *IEEE Journal on Selected Areas in Communication*, vol. 23, no. 8, pp. 1588–1595, Aug. 2005.