

On Providing Fair Circuit/Virtual-Circuit Networking Services

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 Engineering

by

Mark E. McGinley

May 2012

© Copyright by
Mark E. McGinley
All rights reserved
May 2012

APPROVAL SHEET

This dissertation is submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Computer Engineering

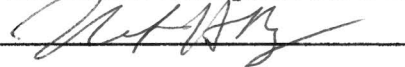


Mark McGinley

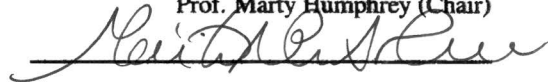
This dissertation has been read and approved by the examining committee:



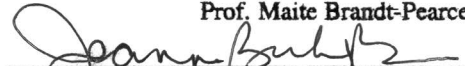
Prof. Malathi Veeraraghavan (Advisor)



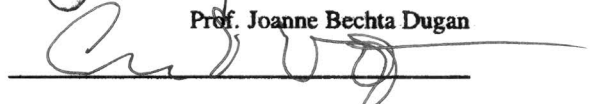
Prof. Marty Humphrey (Chair)



Prof. Maite Brandt-Pearce

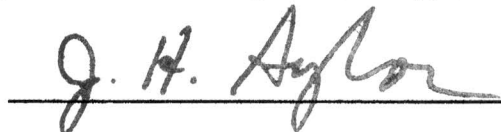


Prof. Joanne Bechta Dugan



Prof. Kamin Whitehouse

Accepted for the School of Engineering and Applied Science:



Dean, School of Engineering and Applied Science

May 2012

Abstract

It has become clear that widespread collaboration in the scientific community, which can increasingly be characterized by geographically distributed and large-scale projects, requires predictable network service [1]. Predictable network service needed by applications such as large file transfers, remote visualization, and remote instrumentation can only be offered on connection-oriented networks. Circuit-switched and virtual-circuit networks offer such connection-oriented services.

This dissertation addresses four issues that inhibit wider adoption of connection-oriented service. From a theoretical perspective, we address reservation systems and queueing systems. An important consideration for a reservation system with multiple classes, as are necessary to provide a level of service matched with a specific use, is fairness. First, we present a novel three-step scheduling algorithm that finds the optimal solution to a multi-class, fairness-considering bin packing problem, and demonstrate how it can be used to ensure fairness in a reservation system. Second, with many types of reservation systems across different disciplines, we recognize a need for a uniform way to describe reservation systems, and so present a novel *general reservation system model (GRSM)*, and analyze several examples of commonly encountered reservation and queueing systems to understand why certain systems belonging to the same category of examples use reservation systems while others use queueing systems.

From an experimental perspective, we address performance and deployment issues with connection-oriented service. First, data-plane performance is paramount to being able to provide satisfactory connection-oriented service. To address this, we have performed an in-depth experimental study of the data-plane issues that arise in circuits, as well as comparing existing transport protocols to a novel implementation designed for circuits. Finally, to provide connection-oriented

service to end hosts, we present a solution that allows an off-the-shelf physical switch to be virtualized into multiple logical switches. Thus, connection-oriented service could be provided on the same existing substrate as connectionless service with no new infrastructure expenditures.

Acknowledgments

Foremost, my sincerest thank you to Prof. Malathi Veeraraghavan, my advisor, who has unceasingly guided me in the completion of this work. It has been an incredibly valuable experience to benefit from her knowledge, experience, and generosity, and a pleasure to work with someone who brings an energy and intellectual curiosity to every discussion.

I thank the postdoctoral fellows and graduate students, Tao Li, Xiangfei Zhu, Xiuduan Fang, Helali Bhuiyan, Xuan Zheng, and Zhenzhen Yan, with whom I have had the honor of working and from whom I have learned a great deal.

I would like to thank Professor Marty Humphrey, Professor Maite Brandt-Pearce, Professor Joanne Bechta Dugan, Professor Alfred Weaver, and Professor Kamin Whitehouse for serving on my defense committee and providing guidance towards the completion of my work.

I would especially like to thank my wife Anne, my parents Michael and Susan McGinley, my wife's parents John and Nancy Hines, and all of our family members who have supported my effort over the years.

This work was carried out under the sponsorship of National Science Foundation grants 1038058 and 1116081 and the Department of Energy grant DE-SC002350. I would like to thank the NSF, DOE, and American taxpayers for funding this research.

Contents

Acknowledgments	vi
1 Introduction	1
1.1 Background	1
1.2 Motivation	3
1.3 Problem statement	5
1.4 Hypothesis formulation	5
1.5 Related work	6
1.6 Dissertation organization	7
1.7 Key contributions	7
2 On Reservation Systems and Queueing Systems	10
2.1 Introduction	10
2.1.1 Key differences between queueing systems and reservation systems	10
2.1.2 Advantages and disadvantages of the two types of systems	11
2.2 Related work	12
2.3 A General Reservation System Model (GRSM)	12
2.3.1 Request	14
2.3.2 Response	14
2.3.3 Resource usage	15
2.4 Examples of reservation and queueing systems	15
2.5 Solving using a Queueing Model	18

2.5.1	M/M/m/ ∞ Queueing Model	18
2.5.1.1	Effect of competition	21
2.5.1.2	Server pooling	22
2.5.2	USST Requests with Single Options	22
2.6	Solving with Discrete Time Markov Chain (DTMC) models	25
2.6.1	Synchronized Server Model	25
2.6.1.1	Performance metrics	26
2.6.1.2	Numerical results	28
2.6.2	Server Vacations Model	30
2.6.2.1	Performance metrics	32
2.6.2.2	Numerical results	35
2.7	Multiple Options Simulation	37
2.8	Conclusions	37
3	Fairness in Multi-Class Book-Ahead Scheduling	39
3.1	Introduction	39
3.1.1	Problem definition	40
3.1.2	Related work	40
3.2	System Model	41
3.2.1	Non-homogeneous Continuous-Time Markov Chain model	42
3.2.2	Discrete-time Markov Chain model	43
3.3	Scheduling Algorithm	43
3.4	Model Solution	50
3.5	Evaluation Approach	52
3.6	Results	53
3.6.1	Sensitivity analysis on system variables m and K	53
3.6.2	Sensitivity analysis on system variable r	55
3.6.3	The impact of the fairness weight ω	56

3.6.4	Relaxing the fairness objective by the optimization variable ϵ	56
3.6.5	Mis-matched capacity demand	56
3.6.6	Comparison of Multi-class BA-First to FCFS	59
3.7	Conclusions	61
4	An in-depth cross-layer experimental study of transport protocols over circuits	62
4.1	Introduction	62
4.2	Background	63
4.2.1	Dynamic circuit service deployment	64
4.2.2	Equipment used in dynamic circuit networks	65
4.2.3	Mismatched-rate circuits and Layer-2 flow control	65
4.2.4	Impact of TCP congestion control on switch buffer occupancy	66
4.2.5	TCP-Ethernet layer interaction within hosts	66
4.3	Prior work	67
4.4	New CTCP Implementation and BWdetail	67
4.5	Experiments	68
4.5.1	Experimental Setup	68
4.5.2	Findings on cross-layer interactions	69
4.5.3	Results	69
4.5.4	Discussion	74
4.6	Conclusions	75
5	On Virtualizing Ethernet Switches	76
5.1	Introduction	76
5.2	Motivation and Related Work	77
5.2.1	Initial drivers	77
5.2.2	Broader motivation and related work	78
5.3	Approach for Virtualization of an off-the-shelf Ethernet Switch	80
5.3.1	Virtualization architecture overview	80

<i>Contents</i>	x
5.3.2 Data-plane support for sliver isolation	81
5.3.3 Slice Administration Controller (SAC)	82
5.4 Implementation and Measurements	84
5.4.1 SAC Design	85
5.4.1.1 Resource control	85
5.4.1.2 Access Transparency	85
5.4.1.3 Security	85
5.4.1.4 Performance	86
5.4.1.5 Reliability	86
5.4.2 Implementation	86
5.4.2.1 Setup	86
5.4.2.2 Operation	87
5.4.2.3 Security	87
5.4.3 Measurements	88
5.5 Conclusions	88
6 Conclusions and Future Work	90
6.1 Conclusions	90
6.2 Future Work	92
Bibliography	93

List of Figures

1.1	A classification of switches	1
1.2	Dimensions of Qos	3
2.1	Reservation and service timelines	13
2.2	M/M/m/ ∞ queueing system; β is ratio of mean waiting time to mean service time .	20
2.3	Solution to requests with a single option.	23
2.4	The comparison of queueing systems with different lengths of time between service intervals τ	29
2.5	The comparison of queueing systems with different vacation settings.	36
3.1	An illustration of the BA-First scheduler for multiple classes.	41
3.2	Optimization example; MRT: Mean Response Time.	50
3.3	Sensitivity analysis on system variables m and K	54
3.4	Sensitivity analysis on system variable r	55
3.5	The impact of the fairness weight ω	57
3.6	Relaxing the fairness objective by the optimization variable ϵ	58
3.7	Mis-matched capacity demand	59
3.8	Comparison of Multi-class BA-First to FCFS	60
4.1	Internal structure of a Gigabit/sec Ethernet (GigE) line card in an Ethernet-SONET MSPP	64
4.2	Experimental Setup	68
4.3	Circuit Rate = OC3 (155 Mbps)	69

4.4	Reno over OC3 - 1GB Transfer	71
4.5	BIC over OC3 - 1GB Transfer	72
4.6	CTCP over OC3 - 1GB Transfer	73
5.1	Virtualization Architecture	82

List of Tables

2.1	Categories with reservation and queueing systems	16
2.2	Characterization of reservation categories	17
5.1	Configuration database for one time interval	84
5.2	Delay for various commands	88

List of Abbreviations

ATCA	Advanced Telecommunications Computing Architecture
ATM	Asynchronous Transfer Mode
BDP	Bandwidth-Delay Product
BIC TCP	Binary Increase Congestion control TCP
CBP	call blocking probability
CBS	Committed Burst Size
CDF	Cumulative Distribution Funtion
CERN	European Organization for Nuclear Research
CHEETAH	Circuit-switched High-speed End-to-End Transport ArcHitecture
CIR	Committed Information Rate
CLI	Command-Line Interface
CTCP	Circuit-TCP
CTMC	continuous-time Markov chain
cwnd	congestion window
CWR	Congestion Window Reduced
DMA	Direct Memory Access
DMV	Department of Motor Vehicles
DoE	Department of Energy
DRAGON	Dynamic Resource Allocation via GMPLS Optical Networks
DTMC	discrete-time Markov chain
EST	Earliest Start Time

FCFS	first-come first-served
FPC	Flexible Packet Classification
FPGA	field-programmable gate array
GB	gigabyte
Gbps	gigabit per second
GENI	Global Environment for Network Innovations
GFP	Generic Framing Procedure
GigE	Gigabit Ethernet
GRSM	general reservation system model
HOPI	Hybrid Optical Packet Infrastructure
IEEE	Institute of Electrical and Electronic Engineers
ION	Interoperable Ondemand Network
IP	Internet Protocol
ITU	International Telecommunication Union
KB	kilobyte
LAN	Local Area Network
MB	megabyte
Mbps	megabit per second
MIB	Management Information Base
MPLS	Multiprotocol Label Switching
MRT	mean response time
MSPP	Multi-Service Provisioning Platform
MSPP	Multi-Service Provisioning Platform
NIC	Network Interface Card
NSF	National Science Foundation
OC	Optical Carrier
OS	Operating System
OSCARS	On-Demand Secure Circuits and Advance Reservation System

PMF	Probability Mass Function
POTS	plain old telephone service
QoS	Quality of Service
REN	research-and-education network
RTT	round-trip time
RWA	routing and wavelength assignment
SAC	Slice Administration Controller
SAC	Slice Administration Controller
SDCS	scheduled dynamic circuit service
SNMP	Simple Network Management Protocol
SONET	Synchronous Optical Networking
SSH2	Secure Shell 2
TCP	Transmission Control Protocol
TCP	Transmission Control Protocol
USST	User-Specified Start Time
VC	virtual circuit
VCG	virtual-concatenation group
VINI	Virtual Network Infrastructure
VLAN	Virtual LAN
VLAN	Virtual Local Area Network
WDM	Wavelength Division Multiplexing
WFQ	Weighted Fair Queueing
WRED	Weighted Random Early Discard

Chapter 1

Introduction

1.1 Background

In communication services, link bandwidth is a resource that typically must be shared in some manner. Communication service offerings can be categorized by the method of sharing as either connection-oriented or connectionless service. In connectionless service, such as IP service, as new data flows start and finish with no admission control actions, the TCP congestion control bandwidth-sharing mechanism [2] was introduced to allow flows such as file transfers to dynamically adjust sending rates so that all ongoing flows sharing a path receive equal shares of the bandwidth (referred to as proportional fairness). In contrast, admission control is a key component of bandwidth sharing in connection-oriented networks. Control-plane software such as OSCARS [3] and DRAGON [4] perform admission control, handling on-demand and advance-reservation requests for circuits/virtual-circuits (VCs).

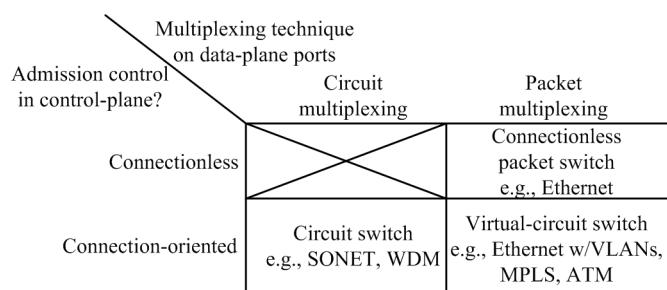


Figure 1.1: A classification of switches

Connection-oriented networks can be created using two types of network switches: packet switches and circuit switches, as shown in Fig. 1.1. The key difference between these two switches is the multiplexing scheme used on the links of the switch. In a circuit switch, the multiplexing scheme used on its links is position-based, be it time, frequency, or space. To support circuit multiplexing in the data plane, an admission-control mechanism is required in the control plane. Admission control is typically used to limit the number of customers sharing a resource by keeping track of the aggregate number of admitted customers. In circuit multiplexing, in addition to tracking aggregate bandwidth allocation, specific positions need to be assigned to a flow when the call is admitted, in order that the user's process can transmit data at the assigned position (e.g., at an assigned timeslot).

In a packet switch, the multiplexing scheme used on its links is packet-based multiplexing in which information carried in the packet header is used to demultiplex packets belonging to a particular flow. In order to use packet switches for connection-oriented services *with guaranteed bandwidth*, an admission-control mechanism is necessary. The admission control mechanism in connection-oriented packet switches, henceforth referred to as *virtual-circuit switches*, is similar to that used in circuit switches. Clearly, the aggregate bandwidth check is required. The step analogous to assigning positions in a circuit switch is the allocation of "labels" or "virtual-circuit identifiers" in the packet's header. Labels and virtual-circuit identifiers are used to demultiplex packets belonging to the same flow. Since packet switches have buffers, the admission control algorithm should not only consider aggregate bandwidth but also buffer space when admitting flows. In addition to admission control, data-plane functions such as policing and scheduling are necessary to support guaranteed bandwidth. The three functions can be visualized on orthogonal axes, as shown in Fig. 1.2. Policing enforces that incoming traffic does not exceed the resources allocated to it (by dropping packets to make the traffic conform), and scheduling influences the egress traffic with functions such as shaping outgoing traffic and using queueing disciplines.

In order to provide service that meets customer demand for specific bandwidth and delay requirements without costing as much as a leased line, research-and-education network (REN) providers as well as commercial network providers have begun offering scheduled dynamic circuit

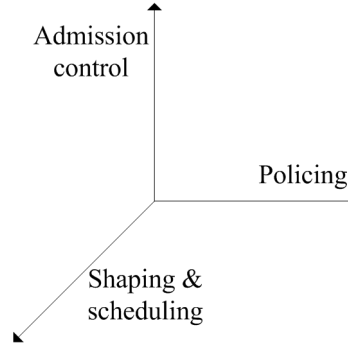


Figure 1.2: Dimensions of Qos

service (SDCS). In SDCS, a customer contracts for an access link to the circuit-switched network, and then requests circuit service of a specific rate and duration only when required. In that manner, SDCS is much like plain old telephone service (POTS), where a monthly fee is charged for access to the telephone network and circuits are requested as an additionally charged action. There are several key differences, though, between SDCS and POTS, and the most important for this discussion is the number of channels into which a link is divided. For example, a T3 line consists of 672 individual voice channels, while an SDCS request could be to support a high-bandwidth application's OC-12 service requirement across an OC-192 link, representing one-sixteenth of the link. As the number of servers m increases, a higher utilization is achievable with decreasing blocking probability. As the number of servers decreases, the utilization falls and the call blocking probability sharply rises. Therefore, for the high number of channels into which links are divided for POTS, a queueing system can operate with low blocking but still achieve the high utilization necessary for profitability. For high-bandwidth circuit service as described above to be able to provide both low blocking probability and high utilization, a reservation system is required, and hence reservations are required as in SDCS.

1.2 Motivation

The motivation for this work stems from a recognition that widespread collaboration in the scientific community, which can increasingly be characterized by geographically distributed and large-scale

projects, requires predictable network service [1]. Predictable network service needed by applications such as large file transfers, remote visualization, and remote instrumentation can only be offered on connection-oriented networks. Circuit-switched and virtual-circuit networks offer such connection-oriented services.

Implemented software, such as OSCARS [3] and DRAGON [4] as used in Internet2's ION network [5], acts as centralized scheduler and performs the control-plane actions required to instantiate a circuit. The theoretical component of bandwidth sharing for connection-oriented networks has been studied extensively, including by our research group [6], but there are two additional aspects further developed in the scope of this work. First, we present a novel *general reservation system model (GRSM)* to be able to describe reservation systems in a uniform way, and second, we address providing *fairness* in scheduling service. Together those comprise the theoretical component of this work.

Though circuits are used in the providers' networks, a packet-switched infrastructure is typically involved on both ends of the connection. Multi-Service Provisioning Platforms (MSPPs), which can include both packet-switched and circuit-switched line cards, function as gateways between the customer's packet-switched infrastructure and the service providers circuit-switched network. A virtual circuit can be established across the customer's packet-switched infrastructure to provide connection-oriented service to the MSPP, and again on the other side of connection, to ensure end-to-end connection-oriented service. The various factors involved in the end-to-end connection, such as mismatched rates between the customer's packet-switched infrastructure and the providers circuit service, can cause performance problems, which are explored as part of the experimental component. The scope of this work includes addressing the cross-layer issues present in extending the connection-oriented service into the customer's packet-switched infrastructure, and includes our method for providing both connection-oriented and connectionless service on the customer's packet-switched infrastructure.

1.3 Problem statement

The hypothesis of this research is that *connection-oriented service can be provided with a fairness-considering advance-reservation system on both circuit-switched and packet-switched infrastructures.*

There are two main components in this work:

1. Theoretical component: the development of both a general reservation system model and analysis of using queueing and reservation systems in general, and an analytical model for a multi-class advance-reservation scheduling system to achieve fairness and,
2. Experimental component: a cross-layer study of transport protocols on circuits as well as a novel method to offer both connection-oriented and connectionless service on a single substrate of packet switches.

A combination of analytical, simulation, and experimental methods are used throughout this work.

1.4 Hypothesis formulation

Theoretical component: As our research group has developed scheduling algorithms, we have continually encountered the need to compare reservation and queueing systems. There has not been a comprehensive treatment on the question of *how to choose whether a system should use reservation scheduling or simply queueing*, and so we have sought to answer that question. In doing so, we recognized the need for a uniform way to describe the various reservation systems we were investigating. To accomplish this, we developed a *general reservation system model (GRSM)*.

Our interest in furthering the analytical work on scheduling led us to pursue the issue of fairness in multi-class advance-reservation scheduling systems. We hypothesize that an optimization can be performed to provide fair scheduling while taking into account mean waiting time and blocking probability, and that we can allow the fairness to be tuned by the system operator to improve performance metrics by relaxing fairness constraints.

Experimental component: The data-plane performance is paramount to being able to provide satisfactory connection-oriented service. To address this, we have performed an in-depth experimental study of the data-plane issues that arise in circuits, as well as comparing existing transport protocols to a novel implementation designed for circuits.

Further, in the customer’s packet-switched infrastructure, packet switches between the data source and sink that are heavily loaded must be configured to provide connection-oriented service¹. Typically, this configuration is performed by the control-plane software. Administrators controlling the network would be wary of granting the control-plane software unfettered access to existing switches. We hypothesize that a solution can be created to enable the existing substrate of switches to be used for connectionless traffic as well as virtual circuits, such that no new infrastructure expenditures would be necessary to provide connection-oriented service.

1.5 Related work

The scope of this work includes four unaddressed issues. In terms of the discussion on reservations and queueing, there is clearly deep research within specific topics, e.g., variations on customer behavior regarding queue abandonments [7], but in general, a lack of the identification of commonalities across disciplines. More specifically, on advance reservation scheduling, most of the work involves the use of simulations, as the state-space of the models hinders analytical approaches. Further, none considers the fairness issue of advance reservations across multiple classes. Relevant to our experimental component, high-speed transport protocols (e.g., HighSpeed TCP [8]) were not designed specifically for circuits. Circuit-TCP [9] was designed for circuits, but the cross-layer interactions that are part of this work were not considered. Finally, while there are other hardware-based or software-based solutions to offer connection-oriented and connectionless service on a packet-switched infrastructure, our approach capitalizes on existing infrastructure without any data-plane performance sacrifice. Subsequent to the publication of our work, OpenFlow [10] has risen in prominence, with commercial and educational participation.

¹In practice, data-plane functions do not need to be instantiated on switches that are lightly loaded, since the rate guarantees would likely be met simply by an excess of capacity.

1.6 Dissertation organization

This dissertation is organized into six chapters. The motivation for this work, relevant background and related work, and a summary of this work's key contributions are provided in this chapter.

In Chapter 2, these differences between reservation systems and queueing systems are characterized, and a *general reservation system model (GRSM)* is proposed. Under four different sets of assumptions, the GRSM is reduced to analytically tractable models for which solutions are provided, and solved with simulations under a fifth set of assumptions. Several examples of commonly encountered reservation and queueing systems are identified and analyzed to understand why certain systems belonging to the same category of examples use reservation systems while others use queueing systems.

In Chapter 3, a *multi-class advance-reservation scheduling system* is developed for use in circuit/VC networks. An important consideration for a scheduling system with multiple classes is fairness. Our scheduling algorithm applies to any bin packing problem or advance-reservation scenario where a flexible class boundary is allowed.

In Chapter 4, an in-depth experimental investigation is performed to gain insights into the complex interactions between the TCP layer, ON/OFF flow control at the Ethernet layer, and switch buffer sizes (the Ethernet line cards have buffers). Using a novel tool and transport protocol designed for circuits, these interactions are characterized.

In Chapter 5, we propose an approach for virtualizing off-the-shelf Ethernet switches that have built-in support for creating isolated bandwidth partitions on their data-plane interfaces.

Finally, this work's contributions are summarized, future work is discussed and the dissertation is concluded in Chapter 6.

1.7 Key contributions

The key contributions are as follows.

1. A *general reservation system model (GRSM)* is proposed to uniformly describe reservation systems. The differences between reservation systems and queueing systems are character-

ized, and categories of reservation systems and queueing systems are analyzed to understand the factors that influence why reservation systems and queueing systems may be used by different systems in the same category. This work has been submitted in a paper to *Manufacturing and Service Operations Management* [11].

2. We present a novel three-step scheduling algorithm that finds the optimal solution to a multi-class, fairness-considering bin packing problem. This model extends our previous single-class book-ahead *BA-First* algorithm into a multi-class algorithm, and provide an analytical discrete-time Markov chain (DTMC) model. The system model is solved using analytical techniques (for small-scale systems), and simulations (for small and larger scale systems), and sensitivity analysis is performed to characterize the algorithm's performance and explore the tradeoff between fairness, mean response time, and utilization. This work has been submitted in a paper to *IEEE Transactions on Networking* [12].
3. An interesting new dynamic was found between flow control at the data-link layer and congestion control at the transport layer. With this in-depth characterization, it is clear that automated mechanisms are necessary to not only configure end-host TCP send and receive buffer sizes as is required for high throughput across IP-routed networks, but for circuit networks, additionally, Ethernet-layer output queue (called *qdisc* in Linux) size needs to be set, along with flow-control related parameters within switches. This work is published in a paper in the *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN2010)* [13].
4. An approach to virtualize off-the-shelf Ethernet switches is presented. Our solution is to implement two software modules that are run external to the switches, a *slice scheduler*, which acts as a reservation scheduler, and a *Slice Administration Controller (SAC)*. We applied our approach to virtualizing a specific Ethernet switch, i.e., the Force10 E600 model. We describe our implementation, and show how a slowdown of 3% to 26%, based on the type of administrative command issued, is experienced when using the SAC. This work is published in a

paper in the *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN2008)* [14].

Chapter 2

On Reservation Systems and Queueing Systems

2.1 Introduction

Resource-sharing mechanisms, in which a set of servers are shared spatially and temporally (i.e., simultaneously by multiple customers), can be classified into the broad categories of *reservation systems* and *queueing systems*. The word “queue” is defined in Webster’s dictionary as a “waiting line of persons or vehicles.” The word “reserve” is defined as “to retain or hold over to a future time or place.” What are the key differences between a reservation system and a queueing system? When should one or the other type of system be implemented?

This paper examines these questions to better characterize the two types of systems. Reservation systems considered here are restricted to online systems, thus excluding batch scheduling systems. The difference is that in online systems, each reservation request is processed immediately and a response is generated, while requests are batched and processed together allowing for optimization in batch systems.

2.1.1 Key differences between queueing systems and reservation systems

1. In addition to the resources being accessed (also referred to as “servers”), reservation systems include a control entity called a *reservation scheduler*, which is not present in queueing systems.

2. In reservation systems, customers are required to specify anticipated service durations in their reservation requests. This is required because a reservation scheduler needs to know when in-service and previously scheduled jobs are going to depart in order to be able to assign a future start time for a new reservation request. In contrast, in queueing systems, customers do not declare their anticipated service durations prior to service.
3. In reservation systems, customers can request advance reservations. This is not a requirement for reservation systems, but given that a reservation scheduler is present, this feature can be supported readily. In other words, a reservation scheduler can support (i) immediate requests and return an earliest start time at which the requested resources are available (we refer to these as *Earliest Start Time (EST)* requests), as well as (ii) advance-reservation requests where users specify a set of start times acceptable to them based on other considerations (we refer to these as *User-Specified Start Time (USST)* requests). In contrast, queueing systems can only support immediate requests as there is no scheduler to accept requests for service in advance.

2.1.2 Advantages and disadvantages of the two types of systems

Queueing systems avoid the costs of having to maintain reservation schedulers, and the need to design penalties for no-shows and cancellations without customer alienation. Furthermore, they offer users more flexibility as customers are not required to specify service durations before the start of service. A disadvantage of a queueing system, however, is that the customer has to wait in a queue between arrival instant and service-start time, while in a reservation system, customers do not need to wait in a queue. Instead, once assigned a service start time, the customer can arrive just before that instant. A second disadvantage of queueing systems is that customers cannot make advance requests as there is no controller to accept such requests.

The **objective of this work is threefold**: (i) to develop a general model for reservation systems by considering several examples of current reservation systems, (ii) to provide analytical solutions to a subset of specific models derived from the general reservation model under certain sets of assumptions, and (iii) to use these solutions to analyze example reservation systems to determine the conditions under which a reservation system is advisable instead of a simpler queueing system.

After reviewing related work in Section 2.2, Section 2.3 describes a general reservation system model (GSRM). Examples of reservation and queueing systems are presented in Section 2.4. For two sets of assumptions, the GSRM is solved using queueing models as presented in Section 2.5. Under other sets of assumptions, the GSRM is solved with Markov chain techniques, as described in Section 2.6, or with simulations, as described in Section 2.7. The paper is concluded in Section 2.8.

2.2 Related work

Queueing and reservations systems are diversely applied and ubiquitous. Correspondingly, the relevant body of work ranges from general queueing theory and scheduling algorithms to specific concerns such as variations on customer behavior regarding queue abandonments [7]. Typically, researchers from various fields do not communicate across disciplines in a way that enables similarities to be recognized. For example, scheduling algorithms as well as theory being developed in the context of high performance computing resource management systems [15] may have practical overlap with that of real-time systems [16] but the discovery of such is hindered by a lack of a uniform way to frame reservation problems. Similar to the Kendall notation [17] for queueing theory, we contribute toward creating a general reservation system model. This model and its notation is developed in the following section.

2.3 A General Reservation System Model (GSRM)

A reservation system model is developed in this section to characterize different types of reservation systems used in everyday life. We refer to this model as a *general single-resource reservation model*. Fig. 2.1 shows two timelines, a reservation timeline and a service timeline. There is a periodicity in both timelines, with T_r and T_s being the **reservation period** and **service period**, respectively. Within a service period, **service intervals** are interspersed with **idle intervals**. Similarly, a reservation period also has **reservation intervals** interspersed with idle intervals (as reservation schedulers could be human agents who take breaks). Within a reservation period T_r , reservations can be accepted during reservation intervals for service within the **reservation window**, which is a

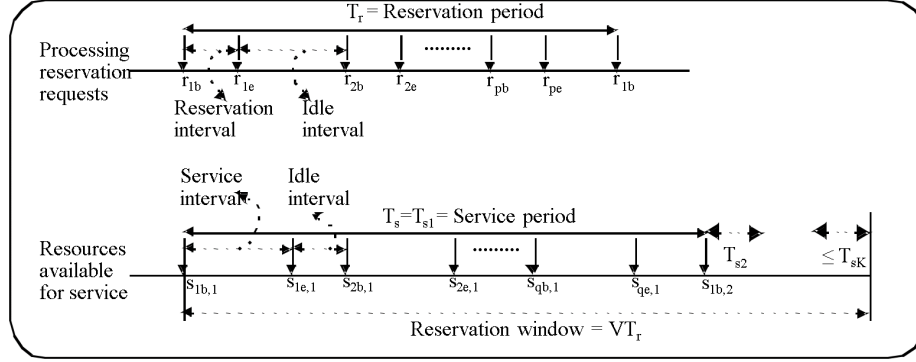


Figure 2.1: Reservation and service timelines

multiple of T_r , shown as VT_r in Fig. 2.1. This may or may not equal an integral multiple of service periods, as shown in Fig. 2.1. The last service period T_{sK} may extend into the next reservation window. For example, a current airline's reservation system advances the reservation window at the end of each day, making T_r equal to 1 day, and it accepts reservations for up to 337 days in advance. The service period is 1 week since the same pattern of flights repeats each week with some exceptions.

Reservation requests are accepted for processing in reservation intervals: (r_{ib}, r_{ie}) , $1 \leq i \leq p$, where p is the number of reservation intervals within a reservation period T_r , while the resources are reserved for use within service intervals: $(s_{ib,j}, s_{ie,j})$, $1 \leq i \leq q$, $1 \leq j \leq K$, where b and e stand for “beginning” and “end,” respectively, q is the number of service intervals within a service period T_s , and K is the number of service periods within a reservation window (as stated earlier, the last service period may not fit entirely in the reservation window).

A service interval consists of one or more service timeslots, where a **service timeslot** is the minimum service reservation duration. The number of resources available for sharing in each service interval is denoted M_{ij} , $1 \leq i \leq q$, $1 \leq j \leq J$, and J is the number of **server groups**.

As an example, consider airline seats on a particular airline between two cities as the shared resource. There could be several flights each week between these two cities. In this example, each service period, T_s , is 1 week. Each flight occupies one service interval. The number of seats available in each class of service (e.g., business and economy) on each flight corresponds to M_{ij} , $1 \leq i \leq q$, $1 \leq j \leq 2$. Most airlines have web-based systems that process reservations anytime,

which means there are no idle intervals in the reservation timeline.

2.3.1 Request

A reservation request from a user is characterized by $\{U \triangleq \{h, n, j, T_{req}\}, t_{arr}\}$, where h is the requested duration (holding time), n is the requested number of resources, j , $1 \leq j \leq J$, is the server group requested, and T_{req} , an optional parameter, is a set of acceptable start times. t_{arr} denotes the arrival time of the reservation request. Some systems could deploy multiple **request classes** per server group, in which case, instead of specifying h and n , a request may simply specify the request class. Thus, if server group j , $1 \leq j \leq J$ supports L_j request classes, a user request could be characterized as $\{U \triangleq \{l, j, T_{req}\}, t_{arr}\}$, where $1 \leq l \leq L_j$. Requests are of two types: *User-Specified Start Time (USST)* and *Earliest Start Time (EST)*. For USST requests, T_{req} is present, and the scheduler interprets this parameter as follows: if the requested resources are available starting at any one of the specified start times in T_{req} , then the request is granted. If not, the request is rejected. For EST requests, T_{req} is absent, and the scheduler interprets the request as follows: schedule the request for the earliest possible start time at which the requested resources become available. A performance metric for both types of calls, USST and EST, is call blocking probability, and an additional metric for EST calls is mean waiting time.

2.3.2 Response

In response, the scheduler returns $t_{allocated}$, i.e., the allocated start time at which the requested resources will be available to the user. For a given t_{arr} within the period $[0, T_r]$, the allocated service-start time for the request is such that $t_{arr} \leq t_{allocated} \leq s_{ie,j}$ for some i , $1 \leq i \leq q$, and some j , $1 \leq j \leq K$, and $(s_{ie,j} - t_{allocated}) \geq h$. This assumes that the requested service holding time is not split between multiple service intervals. This can be relaxed in future models. Further, for USST requests, $t_{allocated} \in T_{req}$. Lastly, $t_{allocated}$ necessarily falls on service timeslot boundaries.

2.3.3 Resource usage

Finally, t_{actual} represents the actual time at which service starts for a particular user request. Ideally $t_{actual} = t_{allocated}$. However, in many practical systems, $t_{actual} > t_{allocated}$; for example, a flight's departure is delayed or a physician is running late for an appointment. In some instances, $t_{actual} < t_{allocated}$, for example, if a patient arrives early for a physician appointment and the physician completes the previous appointment before anticipated. However, this variability does not influence the reservation scheduling process itself, in terms of allocated start times, as well as accept/reject decisions. It does not impact the rate of blocked requests in the short term. A long-term analysis of $t_{allocated}$ and t_{actual} times can be carried out to determine if the class definitions, service intervals, and service periods need to be modified. For example, it could lead to a revision of the service timeslot duration and/or a better estimator of required service holding time in systems such as physicians' offices.

More sophisticated reservation systems can be designed in which users provide several parameters characterizing the distribution of service duration (or at least first and second moments). This will allow schedulers to implement algorithms in which users are provided probabilistic characterizations of their $t_{allocated}$ times, which could result in better server utilization through statistical multiplexing. However, we will see in Section 2.4 that this level of sophistication is not implemented in practice. For example, a physician's administrative assistant simply estimates the required appointment period by asking the patient the reasons for scheduling a visit.

2.4 Examples of reservation and queueing systems

One-time events, such as sporting events or concerts, typically use reservation systems if the number of resources (e.g., seats) is large, and queueing systems if the number is small, e.g., a music band in a bar. If the number of resources is large, there would be long queue build-up if all customers had to stand in line to purchase tickets, which is why reservation systems are used for one-time events. The more interesting examples are of **periodic** systems, in which a set of resources is shared over multiple recurring periods. A categorization of examples of periodic systems is given in Table 2.1,

Table 2.1: Categories with reservation and queueing systems

Category	Queueing Examples	Reservation Examples
Transportation	Seats on commuter buses, subways	Seats on airlines, Amtrak-type trains
Restaurants	Fast-food	Reservations accepted or required
Human servers	Bank tellers	Physicians' offices
Equipment / physical space	Weight-training machines / public restrooms	Medical equipment / university classrooms
Government facilities	Renewing drivers' licenses	DMV commercial license
Communications bandwidth	Plain old telephone service (POTS) and packet switching	Scheduled dynamic circuit service (SDCS)

with both reservation and queueing examples for each category.

In Table 2.1, there are examples of both reservation systems and queueing systems within each category. In *transportation*, seats on flights and Amtrak-type trains are shared with a reservation system, while subway and bus seats, road lanes and street parking spots are shared in queueing mode. In the *restaurant* category, high-end restaurants accept phone reservations for several days in advance, while at the other end of the spectrum, fast-food restaurants are queueing based. In the *human servers* category, physician offices have reservation systems while customers queue up for bank tellers. The *equipment/physical space* category includes sporting equipment and facilities, university facilities, medical equipment, homeowner rental equipment, and out-of-towner needs such as hotel rooms and rental cars. While physical space such as racquetball courts, university classrooms, hotel rooms, and equipment such as medical equipment, homeowner equipment, and rental cars, are shared with reservation systems, other physical space such as public restrooms, and equipment such as weight-training machines, are shared in queueing mode. Some *government* agencies, such as the Department of Motor Vehicles, offer some services, such as drivers' licenses with a queueing system, and other services, such as commercial licenses with a reservation system. *Link bandwidth* can likewise be shared using either a reservation system, such as provided by ES-nets's dynamic circuit service [18], or as a queueing system as in the Plain Old Telephone Service (POTS) or packets in an IP router.

Table 2.2: Characterization of reservation categories

Example	Request type	EST Criteria			USST Criterion	Solution
		Service time	Vacations	Synchronized servers	Number of options	
Transportation (as soon as possible, e.g., emergency trip)	EST	1 timeslot	Yes	Yes	N/A	Synchronized Server Model, Sec. 2.6.1
Transportation (1 option, e.g., business trip)	USST	N/A	N/A	N/A	1	Requests with Single Options, Sec. 2.5.2
Transportation (many options, e.g., leisure)	USST	N/A	N/A	N/A	Many	Multiple Options Simulation, Sec. 2.7
Restaurants (1 option, e.g., special occasions)	USST	N/A	N/A	N/A	1	Requests with Single Options, Sec. 2.5.2
Restaurants (many options, e.g., high-end)	USST	N/A	N/A	N/A	Many	Multiple Options Simulation, Sec. 2.7
Human servers (vacations forgiven)	EST	Random	No	No	N/A	M/M/m/ ∞ Queueing Model, Sec. 2.5.1
Human servers (vacations included)	EST	1 timeslot	Yes	No	N/A	Server Vacations Model, Sec. 2.6.2
Equipment/physical space (e.g., car rental, hotel room for a particular day)	USST	N/A	N/A	N/A	1	Requests with Single Options, Sec. 2.5.2
Equipment/physical space (flexible users)	USST	N/A	N/A	N/A	Many	Multiple Options Simulation, Sec. 2.7
Government facilities, e.g., DMV commercial license	EST	Random	No	No	N/A	M/M/m/ ∞ Queueing Model, Sec. 2.5.1
Communications bandwidth (e.g., SDCS)	USST	N/A	N/A	N/A	1	Requests with Single Options, Sec. 2.5.2

Shown in Table 2.2 are the key characteristics of the reservation system examples from Table 2.1. The first distinction is which type of requests the system accepts: *Earliest Start Time (EST)* or *User-Specified Start Time (USST)*, as described in Section 2.1. For EST systems, there are three classifying criteria: the service time distribution, whether or not the service timeline has idle intervals (i.e., servers taking vacations), and whether or not the servers are synchronized (e.g., as with an airplane in which the seats are servers, and all servers simultaneously begin and end service on a flight). For USST systems, the criterion is whether one option or many options are specified in the request. The reservation system examples in Table 2.1 fit one of the groups according to this set of criteria, and models are solved for each group as presented in the following sections.

In all examples of queueing systems in Table 2.1, there is no scheduler to accept reservations for servers a priori. Instead a customer must simply wait in line until a server becomes available. For all examples of reservation systems, there is a scheduler and customers are required to provide it the required service duration (or at least an estimate of it) in the reservation request. The question

of **how to choose between these two modes** is answered in the following sections.

2.5 Solving using a Queueing Model

2.5.1 M/M/m/ ∞ Queueing Model

This section considers the first of three solution methods used to solve the reservation and queueing systems described. For a certain set of assumptions, the GRSM described in Section 2.3 is reduced to a form equivalent to an M/M/m/ ∞ queueing model, for which an analytical solution is known.

Consider the following set of assumptions that simplify the GRSM:

1. There are no idle intervals in the reservation or service timelines.
2. One reservation period consists of a single reservation interval ($p = 1$), which in turn consists of a single service timeslot.
3. The reservation window consists of a single service period ($K = 1$) with a single service interval ($q = 1$). There are ∞ service timeslots in this single service interval, i.e., since the reservation period T_r is one service timeslot, V in Fig. 2.1 is ∞ .
4. There is only a single server group, which consists of m servers.
5. All requests are for a single server, and there is only one request class. All requests are handled in first-come, first-served mode.
6. Inter-arrival times between requests are exponentially distributed and independent.
7. All requests are of the EST type.
8. Service durations in reservation systems are discrete, since durations are specified as integral multiples of the service timeslot in reservation requests. In this simplified model, service durations are assumed to be geometrically distributed, and the service timeslot is assumed to be infinitesimally small. The latter assumption is made so that the service duration distribution is approximately exponential, which allows for the use of the M/M/m/ ∞ queueing model solution.

As shown in Table 2.2, the example categories of human servers (vacations forgiven) and government facilities fall in this group. As customers only solicit service during open hours, the idle intervals are ignored in this model, and service intervals are arranged back-to-back in one continuous service timeline. The service time for these examples may not always be exponentially distributed, but we make this simplifying assumption here for analytical tractability so as to be able to answer the question of when a reservation system is required and when a queueing system is sufficient.

The solution to the simplified GRSM is the same as that of the $M/M/m/\infty$ queueing model. The reservation window is equivalent to the buffer in the queueing system.

The analytical solution for the $M/M/m/\infty$ queueing model is well known [19]. The mean waiting time, $E[W]$, can be expressed in closed form as a function of customer arrival rate, λ , service completion rate, μ , and the number of servers m . Of interest, is the factor

$$\beta = E[W]/E[S]$$

where the mean service time, $E[S] = 1/\mu$. Per-server utilization, U , is given by

$$U = \rho/m$$

where the offered load, $\rho = \lambda/\mu$.

Fig. 2.2 plots per-server utilization, U , against the number of servers, m , for fixed values of β . These plots allow a resource owner to determine a suitable operating point. The owner of any set of resources for which there is a customer base typically has a target revenue level and a target cost level. The revenue level determines the desired offered load, ρ , and the target cost level determines the number of servers (resources), m , that the owner can deploy for simultaneous operation.

Consider a system in which the owner determines from a target revenue and cost consideration to operate with ρ value of 1.635 and with m , the number of servers, set to 2. This corresponds to the point $X = 2; Y = 0.8175$ on the lower plot of Fig. 2.2. The ratio of mean waiting time to mean service time, β , is 2 at this point of operation. This means a customer has to, on average, wait two times as long as the average time it takes for a server to serve one customer. Compare this point of

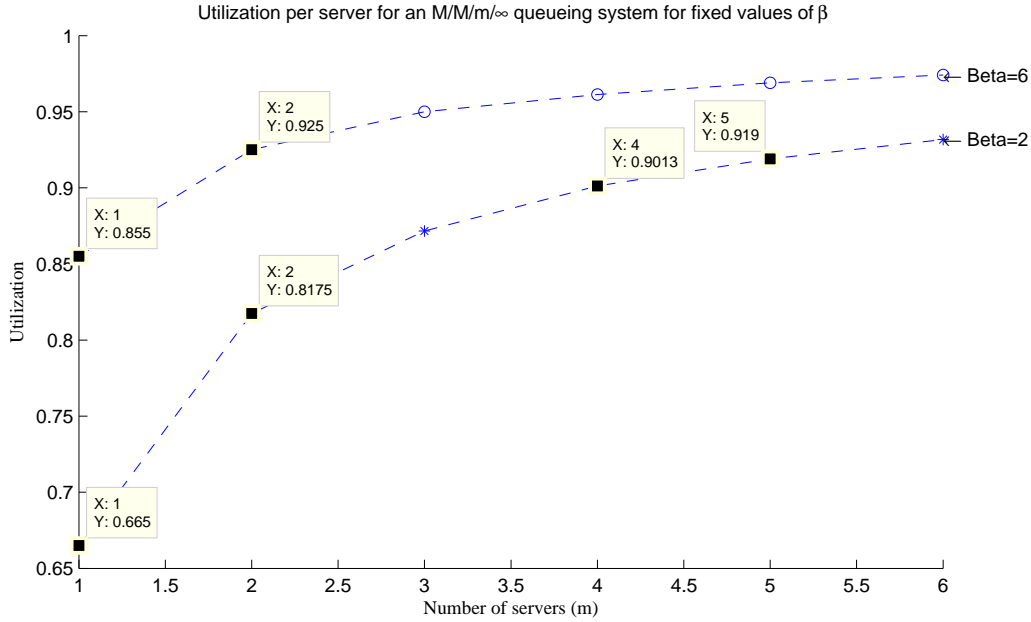


Figure 2.2: M/M/m/∞ queueing system; β is ratio of mean waiting time to mean service time

operation with the point of operation at the same value of m , i.e., 2 servers, but for a β of 6. On the upper plot of Fig. 2.2, corresponding to $\beta = 6$, the utilization per server is 0.925 in contrast to the 0.8175 for $\beta = 2$, for $m = 2$.

As examples of these two operating points, consider the *human servers* category, with physicians and bank tellers, from Section 2.4. Given the relatively higher salaries of physicians when compared to bank tellers, it is more likely that a physicians' office operates with a per-server (physician) utilization of 92.5%, while a bank teller operates at 81.75%. To operate at this higher utilization level, the owner of the physicians' office should have engineered the load ρ to be $0.925 \times 2 = 1.85$, while the load to the bank tellers is $0.8175 \times 2 = 1.635$.

Now consider the implication of the higher β for the physicians. A higher β implies a higher mean waiting time for customers. If for example, the mean service time at the physicians' office is 30 minutes, it means the mean waiting time will be 3 hours when $\beta = 6$. It is the combination of the

1. desired β , which is determined by the target (desired) offered load, ρ (a measure of revenues),

and target number of servers, m (a measure of costs), and

2. mean service time

that together determine the mean waiting time for customers. The next question is “are customers willing to wait in a queue for this length of time on average” or is the expected waiting time long enough to warrant a reservation system so that customers can carry out other activities between their times of request and allocated start times rather than wait in a queue. This is the key advantage provided by reservation systems relative to queueing systems as noted in Section 2.1. The disadvantages of reservation systems, i.e., the requirement for users to specify service durations, and the cost of running a scheduler and handling no-shows have to be borne if (i) the target β is high (which stems from a desired high per-server utilization), and (ii) mean service time is large. This explains why physicians have reservation systems while bank tellers do not.

In summary, with EST-type requests, the answer to the question of when to implement a reservation system is as follows. If the mean waiting time for customers is large at the optimal point of operation (which is determined by a consideration of revenues and costs), then a reservation system should be deployed so that customers can carry out other activities between their time of service request and start of service instead of waiting in a queue.

2.5.1.1 Effect of competition

The β value is often dictated by competition as customers will perceive differences in average wait times. It is easier for an owner to adjust the number of servers, m , than to alter offered load ρ through marketing and other indirect means. For a given ρ , say 0.9, the owner can choose to operate with just one server $m = 1$ but the corresponding β is 10. If a competitor operates with 2 servers to handle this same load of 0.9, the β value falls significantly to just 0.1. The competitor’s customers will experience far smaller waiting times forcing the first owner to add a server.

2.5.1.2 Server pooling

If it is possible to pool servers together and direct a correspondingly larger load to this server pool, then efficiencies can be realized. For example, instead of operating with 2 servers at $\beta = 6$ (with a per-server utilization of 0.925), if the owner can pool together servers and operate a system with 5 servers, then a nearly equal per-server utilization of 0.919 can be achieved with a β of just 2, as seen in Fig. 2.2, or higher per-server utilization of 0.97 can be achieved while maintaining a β of 6. This is the economies of scale advantage.

2.5.2 USST Requests with Single Options

A system in which all requests contain a single option can be modeled as an M/G/m/m queueing system. Each timeslot is modeled with a single M/G/m/m queue, the solution for which is Erlang-B. The same simplifying assumptions as in Section 2.5.1 are made with the following changes in certain assumptions:

1. Assumption 3: Instead of the infinite reservation window, here the reservation window is assumed to be finite, with V service timeslots.
2. Assumption 7: All requests are of USST type, not of EST type.
3. Assumption 8: Typically, for single-option USST systems, the requests are for a single timeslot.

As discussed in Section 2.5.1, the offered load is a measure of expected revenues and the number of servers, m , is an estimate of costs. Fig. 2.3 shows that the lower the desired blocking probability, the lower the utilization for fixed values of m and offered load (ρ). Per the Erlang-B formula, only 2 of the 4 variables, offered load, number of servers, blocking probability, and utilization, can be chosen independently. For example, if $m = 30$, and the desired call blocking probability is 5%, the only feasible per-server offered load ρ/m is approximately .83, and correspondingly, the utilization is approximately 78.5%.

Of the example reservation systems listed in Table 2.2 that fit this model, the communications bandwidth and the equipment/physical space categories are used to answer the question of *how*

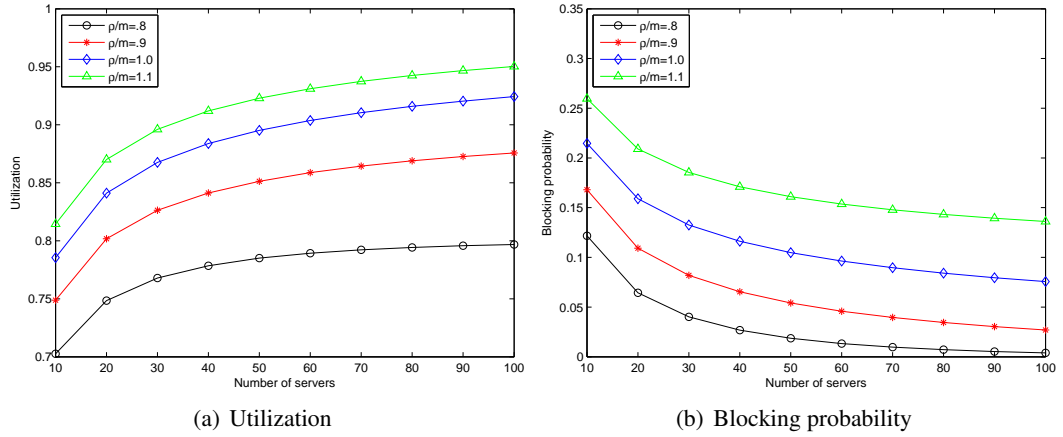


Figure 2.3: Solution to requests with a single option.

to choose between reservation and queueing systems. Research and education network (REN) providers as well as commercial network providers have begun offering scheduled dynamic circuit service (SDCS). In SDCS, a customer contracts for an access link to the circuit-switched network, and then requests circuit service of a specific rate and duration only when required. In this respect, SDCS is similar to plain old telephone service (POTS), where a monthly fee is charged for access to the telephone network and circuits are requested as an additionally charged action. However, SDCS is operated with a reservation scheduler, while POTS is a bufferless queueing system. There is no reservation scheduler in POTS, and users are not required to specify call durations when making their call requests. The reason we describe POTS as a “bufferless” queueing system is as follows. POTS is implemented as strictly a call blocking system because it needs bandwidth on multiple links of end-to-end paths, and holding resources at upstream switches while waiting for bandwidth at downstream switches will make the system under-utilized. Therefore, there are no buffers to hold calls, and calls are simply rejected if bandwidth is not available on any single link of the end-to-end path. This makes POTS a bufferless queueing system. As an aside, another example of a bufferless queueing system is street parking for the night in residential neighborhoods.

Consider the question of why SDCS designers felt the need for a reservation scheduler when POTS has been in operation without such a scheduler. The reason is because given the low bandwidth required per POTS call (i.e., 64 kbps), the number of channels into which a link is divided

can be large. For example, a T3 line can carry 672 individual voice channels. When the number of channels (“servers” in Fig. 2.3) is large, the system can be operated at a low call blocking probability without sacrificing utilization. In contrast, with SDCS, requests are expected to be a significant portion of link capacity, e.g., a 1 Gbps virtual circuit on a 10 Gbps Ethernet link, making the number of channels only 10. With this number of servers, high utilization cannot be achieved without high call blocking probability as shown in Fig. 2.3. High utilization is necessary for profitability. If users are only allowed a single option in their start time specifications, the call blocking probability will be just as high as with POTS. However, the cost of rejecting a call starting at a future time instant (which is feasible with a reservation scheduler) is smaller than in a system like POTS that does not have a scheduler to accept future start time requests. Furthermore, as Section 2.7 shows, call blocking probability falls significantly with multiple start time options.

As a second example, consider the equipment/physical space category, and specifically car rentals/hotel rooms. Users often request these facilities for a specific day/night to match their out-of-town visits, making this category fit the USST single option case. A user’s desire for guaranteed service is the dominant characteristic. If these systems are operated in a queueing mode, the customer would be severely inconvenienced if stranded without a rental car after having arrived at their destination. They would be disappointed with that provider, damaging the provider’s reputation and affecting future demand for that provider’s service. The cost to the provider to reduce blocking to very low levels would be high (both in terms of buying enough cars and maintaining a large fleet). With a larger fleet, the utilization would be lower, decreasing the provider’s return on investment. Conversely, by operating with a reservation system, the inconvenience experienced by a customer being blocked is relatively minor, as they presumably can reserve a rental car with another company in a matter of minutes with online reservations. Therefore, the provider does not need to operate at a low blocking probability, and hence can limit the size of the fleet to enable a higher utilization and return on investment. This explains the use of reservation systems instead of the simpler queueing system for this category of examples.

2.6 Solving with Discrete Time Markov Chain (DTMC) models

The second of the three solution methods to solve the reservation and queueing system models is using Discrete Time Markov Chain (DTMC) models. In each of the two following sections, a specific set of assumptions are applied to the GRSM to model example systems from Table 2.2.

2.6.1 Synchronized Server Model

Here we consider a synchronized server model, such as an airline flight, where a seat is considered a server and the servers must begin and end service simultaneously. If the system is considered at the instant before service begins, the system can be modeled as a DTMC (e.g., just before a flight departs). The system state is simply (n) , the number of customers in the queue at the instant just before the start of a service interval.

The GRSM reduces to this synchronized server model under a certain set of assumptions, which differ from those of Section 2.5.1 only as follows:

1. Assumption 1: There are idle intervals in the service timeline, but none in the reservation timeline.
2. Assumption 2: One reservation period consists of a single reservation interval ($p = 1$).
3. Assumption 3: The reservation window consists of Z/m service periods, where m is the number of servers. Each service period has one service interval and one idle interval. The service interval plus idle interval is τ .
4. Assumption 8: Service intervals consist of a single timeslot of a short duration.

The system state space S is defined as

$$S \triangleq \{s = n : 0 \leq n \leq Z\},$$

i.e., a newly arriving customer who finds Z customers already assigned for service in all available timeslots will be blocked.

The transition probability in the DTMC from state n to state n' , denoted by $p_{n,n'}$, is

$$p_{n,n'} = \begin{cases} P(n') & \text{if } n \leq m, \\ P(n' - (n - m)) & \text{if } n > m, \end{cases} \quad (2.1)$$

where $P(a)$ is defined as

$$P(a) = \begin{cases} P_A(a) & \text{if } a < Z, \\ 1 - F_A(Z - 1) & \text{if } a \geq Z, \end{cases} \quad (2.2)$$

where $P_A(a)$ and $F_A(a)$ are the Probability Mass Function (PMF) and Cumulative Distribution Function (CDF), respectively, of A , a Poisson random variable with parameter $\lambda\tau$, representing the number of customer arrivals within the interval between the beginnings of two consecutive timeslots.

2.6.1.1 Performance metrics

To characterize the performance of queueing systems with vacations, we use the following three metrics: P_B , the blocking probability, which is defined as the ratio of customers blocked to the total number of customer arrivals in a long observation interval [20], W , the mean waiting time for admitted customers, and U , the long-run utilization of the system. To calculate these three metrics, we first calculate the steady-state probabilities, denoted by vector π , of this DTMC based on the transition matrix given in Eq. (2.1) using well-known techniques [20].

2.6.1.1.1 Blocking probability In any time interval between the beginnings of two consecutive service timeslots, τ , the average number of customer arrivals is $\lambda\tau$. The number of customers blocked in this interval depends upon the state of the system at the start of the interval. We therefore use a vector ($b \triangleq b_n, n \in S$), where each element b_n denotes the blocked-request ratio in this interval if the system is in state n at the start of the interval. We define the accepted-request ratio as Q_n . We can compute b_n as follows:

$$\begin{aligned} b_n &= 1 - Q_n \\ &= 1 - \frac{\sum_{j=0}^{d_n} jP_A(j) + d_n(1 - F_A(d_n))}{\lambda\tau}, \end{aligned} \quad (2.3)$$

where $P_A(i)$ and $F_A(i)$ are the PMF and CDF of the previously defined random variable A , respectively, and d_n is the number of empty spaces in the queue in that interval, which can be calculated as

$$d_n = \begin{cases} Z & \text{if } n \leq m, \\ Z - (n - m) & \text{if } n > m. \end{cases} \quad (2.4)$$

The long-run blocking probability, P_B , can then be computed as

$$P_B = \pi b^T, \quad (2.5)$$

where the components of the vector b are the blocked-request ratios computed in Eq. (2.3).

2.6.1.1.2 Mean waiting time The waiting time of a customer consists of a fractional part, which is the delay within the arrival interval, and an integral part, which is a multiple of τ before the actual service is provided. Consider the interval between the beginnings of two arbitrary consecutive service timeslots $\tau = (t, t + \Delta t]$. Let w_n be the average waiting time conditioned on the embedded DTMC being in state n at time t . The average is calculated over the customers admitted during the interval of interest. Denote r_n and z_n as averages of the fractional parts and integral parts of waiting times, respectively. We have $w_n = r_n + z_n$. Using the same derivation as the one used in [21], we can compute r_n and z_n as follows.

$$r_n = \tau \left(1 - \frac{1}{2} F_A(d_n) - \frac{d_n + 1}{2\lambda} (1 - F_A(d_n + 1)) \right), \quad (2.6)$$

$$z_n = \frac{1}{a_n} \left(\sum_{j=1}^{d_n-1} P_A(j) v(n, j) + (1 - F_A(d_n - 1)) v(n, d_n) \right), \quad (2.7)$$

where d_n is given in Eq. (2.4), $P_A(i)$ and $F_A(i)$ are the PMF and CDF of the previously defined random variable A , respectively, a_n is the conditional expected number of customers admitted in the interval, which can be computed as

$$a_n = \sum_{j=1}^{d_n-1} j P_A(j) + d_n (1 - F_A(d_n - 1)), \quad (2.8)$$

and $v(n, j)$ is the sum of the integral parts of waiting times conditioned on the DTMC being in state n with j customers admitted in the interval, which can be computed as

$$v(n, j) = \sum_{l=1}^j \left\lfloor \frac{l + \max((n - m), 0)}{m} \right\rfloor \tau. \quad (2.9)$$

The long-run mean waiting time W is

$$W = \frac{\sum_{n \in S} \pi_n a_n w_n}{\sum_{n \in S} \pi_n a_n}. \quad (2.10)$$

2.6.1.1.3 Utilization System utilization in a time interval $(t, t + \tau]$ depends upon the number of customers waiting to be served at time t . We define the vector $(u \triangleq u_n, n \in S)$, where each element u_n is the utilization if the system is in state n in any interval. Therefore u_n can be calculated as

$$u_n = \begin{cases} \frac{n}{m} & \text{if } n < m, \\ 1 & \text{if } n \geq m. \end{cases} \quad (2.11)$$

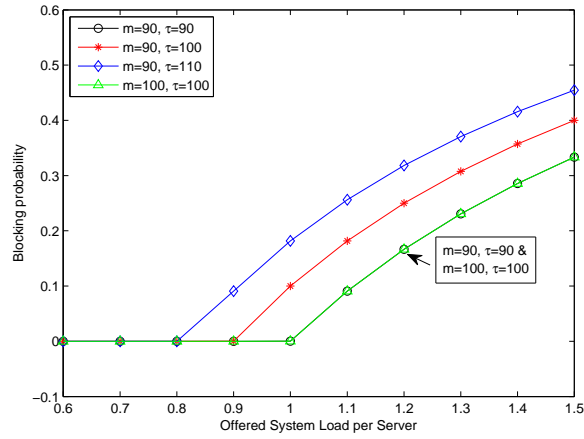
The long-run utilization can then be calculated by

$$U = \pi u^T. \quad (2.12)$$

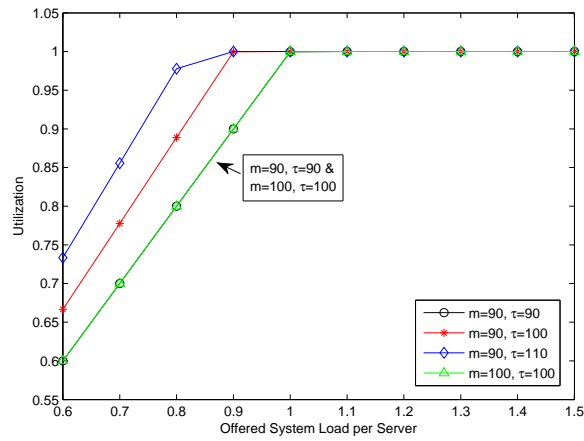
2.6.1.2 Numerical results

The impact of varying the inter-service time τ is illustrated in Fig. 2.4. As τ increases, there are more customer arrivals relative to the servers, which causes an increase in all performance metrics. The increase is also more pronounced as τ increases, as evidenced by the clearly higher slope of the utilization and mean waiting time curves.

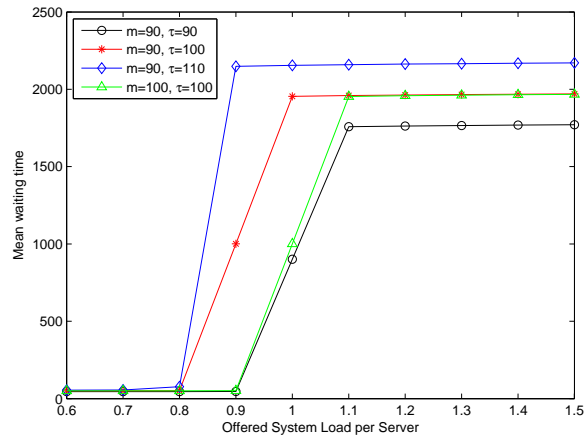
Consider the transportation category from Table 2.2. Increasing the number of seats on an airplane from $m = 90, \tau = 100$ to a larger $m = 100$ with the same $\tau = 100$ causes a decrease in blocking probability but also a decrease in utilization. Holding the number of seats constant at $m = 90$ while increasing the τ from 100 to 110 increases the blocking probability while simultaneously



(a) Blocking probability



(b) Utilization



(c) Mean waiting time

Figure 2.4: The comparison of queueing systems with different lengths of time between service intervals τ .

increasing utilization. Now consider the blocking probability and mean waiting time curves. Of the cases considered, the system with $m = 90$ and $\tau = 110$ has the highest blocking probability and at the same time, the highest mean waiting time. While blocking probability can be dropped by increasing m or decreasing τ , the latter is the better option from a consideration of mean waiting time, though both these options lower utilization.

Both utilization and mean waiting time are factors considered in the *choice of operating as a reservation system rather a queueing system*. For profitability, it is necessary to operate at high utilization. Systems with higher operating costs must aggregate customer arrivals over a longer τ or decrease the number of seats m in order to achieve the high utilization necessary for profitability. The costs per flight, such as the pilot, crew, airport fees, fuel, etc., are such that operating a larger flight is more cost effective than multiple smaller flights. Therefore, to achieve a higher utilization, a longer τ is required. Buses, conversely, are less expensive to purchase and operate (e.g., lower fuel costs, no airport fees, lower paid operators, no flight attendants, etc.). To achieve profitability, buses can operate at lower utilization, and therefore can have a lower τ . With a lower τ , mean waiting time is lower, making it acceptable to have customers wait for service as would be needed in a queueing system rather than deal with the cost and complexity of instituting a reservation system. But if the τ required to operate a large aircraft (with large m) at high utilization is high, then to save customers from having to wait in a queue (as described in Section 2.5.1), a reservation scheduler is required as with airlines.

2.6.2 Server Vacations Model

In most of the example systems listed in Section 2.4, there are idle intervals in the service timeline. To account for the idle periods as part of the waiting time, the simplifying assumptions made in Section 2.5.1, which allowed for the use of the $M/M/m/\infty$ queueing model solution, must be changed. Therefore, a different set of assumptions are considered here, which allows for solving the model with a DTMC.

1. There are no idle intervals in the reservation timeline, but there are idle intervals in the service timeline.

2. One reservation period (which is also one reservation interval) is equal to one service timeslot.
3. A service period has a single service interval followed by a single idle interval. There are N_s service timeslots, each of duration τ , in the single service interval, and the idle interval is $N_v\tau$ in length. The reservation window consists of $V/(N_s + N_v)$ service periods.
4. There is only a single server group, which consists of m servers.
5. All requests are for a single server, and there is only one request class. All requests are handled in first-come, first-served mode.
6. Inter-arrival times between requests are exponentially distributed and independent.
7. All requests are of the EST type.
8. All requests specify a service duration of one service timeslot.

The solution to this model is similar to the synchronized server solution above, so for brevity we will only point out the differences. The system state is now represented by a 2-tuple (i, n) , where i identifies which timeslot the system is about to enter, and n represents the number of customers in the system waiting to be served.

The system state space S is defined as

$$S \triangleq \{s = (i, n) : 1 \leq i \leq N_s \text{ \& } 0 \leq n \leq Z\},$$

where N_s is the number of service timeslots per service interval, and Z is $VmN_s/(N_s + N_v)$.

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

$$p_{(i,n),(j,q)} = \begin{cases} P(q) & \text{if } j = 1 + i\%N_s \\ & \text{\& } n \leq m, \\ P(q - (n - m)) & \text{if } j = 1 + i\%N_s \\ & \text{\& } n > m, \\ 0 & \text{otherwise,} \end{cases} \quad (2.13)$$

where $P(a)$ is defined as

$$P(a) = \begin{cases} P_A(a) & \text{if } a < Z, \\ 1 - F_A(Z - 1) & \text{if } a \geq Z, \end{cases} \quad (2.14)$$

where $P_A(a)$ and $F_A(a)$ are the Probability Mass Function (PMF) and Cumulative Distribution Function (CDF), respectively, of A , a Poisson random variable with parameter $\lambda\Delta t$, representing the number of customer arrivals within the interval between the beginnings of two consecutive timeslots. The length of Δt depends on whether there is a vacation interval between the two service timeslots, i.e.,

$$\Delta t = \begin{cases} \tau & \text{if } i < N_s, \\ (N_v + 1)\tau & \text{if } i = N_s. \end{cases} \quad (2.15)$$

2.6.2.1 Performance metrics

The performance is characterized using the same three metrics: the blocking probability P_B , mean waiting time W , and long-run utilization U .

2.6.2.1.1 Blocking probability The blocking probability is determined in the same way, with the only notational difference being to accommodate the different state definition. We now use a vector ($b \triangleq b_{(i,n)}, (i,n) \in S$), where each element $b_{(i,n)}$ denotes the blocked-request ratio in this interval if the system is in state (i,n) at the start of the interval. We define the accepted-request ratio as $Q_{(i,n)}$ and compute $b_{(i,n)}$ as follows:

$$\begin{aligned} b_{(i,n)} &= 1 - Q_{(i,n)} \\ &= 1 - \frac{\sum_{j=0}^{d_{(i,n)}} jP_A(j) + d_{(i,n)}(1 - F_A(d_{(i,n)}))}{\lambda\Delta t}, \end{aligned} \quad (2.16)$$

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 number of empty spaces in the queue in that interval, which can be calculated as

$$d_{(i,n)} = \begin{cases} Z & \text{if } n \leq m, \\ Z - (n - m) & \text{if } n > m. \end{cases} \quad (2.17)$$

The long-run blocking probability, P_B , can then be computed as

$$P_B = \pi b^T, \quad (2.18)$$

where the components of the vector b are the blocked-request ratios computed in Eq. (2.16).

2.6.2.1.2 Mean waiting time The waiting time of a customer consists of a fractional part, which is the delay within the arrival interval, and an integral part, which is the number of service and vacation timeslots before the actual service is provided. Until the computation of the integral part, however, the only difference is state notation.

Denote $r_{(i,n)}$ and $z_{(i,n)}$ as averages of the fractional parts and integral parts of waiting times, respectively. We have $w_{(i,n)} = r_{(i,n)} + z_{(i,n)}$. The computation is the same as above, except where the integral parts are computed.

$$r_{(i,n)} = \left(1 - \frac{1}{2}F_A(d_{(i,n)}) - \frac{d_{(i,n)} + 1}{2\lambda}(1 - F_A(d_{(i,n)} + 1)) \right) \Delta t, \quad (2.19)$$

$$z_n = \frac{1}{a_n} \left(\sum_{j=1}^{d_n-1} P_A(j)v(n, j) + (1 - F_A(d_n - 1))v(n, d_n) \right), \quad (2.20)$$

where Δt is given in Eq. (2.15), $d_{(i,n)}$ is given in Eq. (2.17), $P_A(i)$ and $F_A(i)$ are the PMF and CDF of the previously defined random variable A , respectively, $a_{(i,n)}$ is the conditional expected number of customers admitted in the interval, which can be computed as

$$a_{(i,n)} = \sum_{j=1}^{d_{(i,n)}-1} jP_A(j) + d_{(i,n)}(1 - F_A(d_{(i,n)} - 1)), \quad (2.21)$$

and $v(i, n, j)$ is the sum of the integral parts of waiting times conditioned on the DTMC being in

state (i, n) at time t for j customers admitted in the interval, which can be computed as

$$v(i, n, j) = \sum_{l=1}^j ((s(l, n) + \lfloor \frac{s(l, n) + i}{N_s} \rfloor N_v) \tau) \quad (2.22)$$

where $s(l, n)$ is the number of service timeslots the j^{th} customer needs to wait before being served, and can be calculated as

$$s(l, n) = \lfloor \frac{l + \max((n - m), 0)}{m} \rfloor \quad (2.23)$$

The long-run mean waiting time W is

$$W = \frac{\sum_{(i, n) \in S} \pi_{(i, n)} a_{(i, n)} W_{(i, n)}}{\sum_{(i, n) \in S} \pi_{(i, n)} a_{(i, n)}}. \quad (2.24)$$

2.6.2.1.3 Utilization The system utilization likewise differs to include the state notation, but also has a substantive difference due to the consideration of the period of time the servers are active. 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) . Therefore $u_{(i, n)}$ can be calculated as

$$u_{(i, n)} = \begin{cases} \frac{n}{m} & \text{if } n < m, \\ 1 & \text{if } n \geq m. \end{cases} \quad (2.25)$$

The long-run utilization while the servers are active U can then be calculated by

$$U = \pi u^T. \quad (2.26)$$

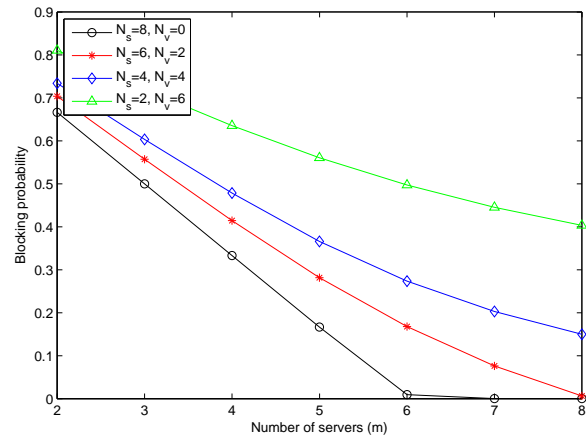
Additionally, we define a scaled long-run link utilization that is the long-run utilization scaled by the portion of time the servers are active as

$$U' = \frac{N_s}{N_s + N_v} \pi u^T. \quad (2.27)$$

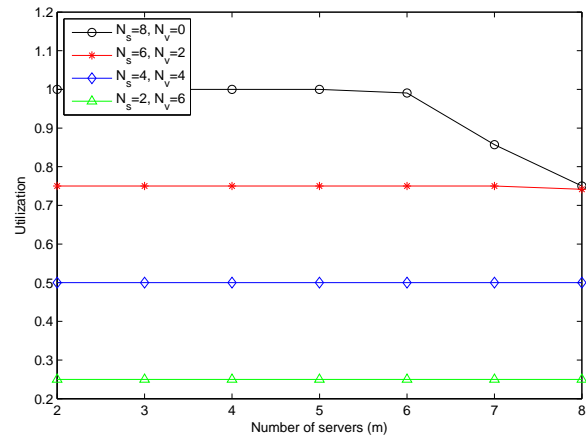
2.6.2.2 Numerical results

Consider the human services (vacations included) category from Table 2.2. In physicians' offices, the minimum service duration, which is set to be the service timeslot, is typically 15 minutes. Physicians work continuously for several service timeslots and then take a "vacation" to handle paperwork and other non-billable functions within a business day. Assume that the customer arrival rate, $\lambda = 6$. The blocking probability, scaled utilization, and mean waiting time of four combinations of doctors serving patients verses taking vacations are plotted in Fig. 2.5. The x-axis is the number of servers and one can see an appreciable decrease in scaled utilization can be seen only when increasing beyond 6 servers operating fully, after which the blocking probability and mean waiting time are relatively negligible. One interesting characteristic is the increasing difference (most noticeable for $m = 2$) in mean waiting time from $N_s = 8, N_v = 0$ to $N_s = 6, N_v = 2$ and so on. The increasing difference comes from the correspondingly increased vacation, whereby a customer in the fully serving case might be scheduled for the 5th timeslot, but for the $N_s = 2, N_v = 6$ case, the 5th effective service timeslot might actually be $2 + 6 + 2 + 6 + 1 = 17$ timeslots away. The results identify the impact of increasing or decreasing the number of doctors and the amount of service time and vacation time. For example, a 4 doctor practice going from 4 hours of seeing patients in an eight hour workday to 6 hours would result in a 6% decrease in blocking probability and an approximately one-third reduction in mean waiting time.

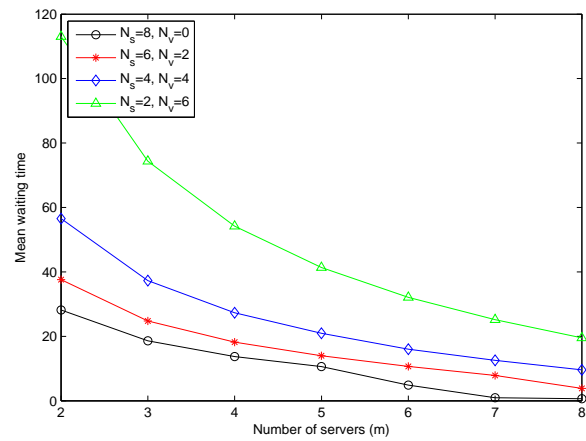
The above includes the assumptions that the demand for physicians service is high, and becoming a physician is difficult and expensive. Thus, a high overall utilization is experienced (due to high demand and low supply), and the above discussion focuses on the adjustable vacation variable, resulting in the scaled utilization. Comparing the physician's office to the queueing example in the human services category, bank tellers, a crucial distinction must be made. When the demand for bank services is high (certain portions of the day, such as around midday corresponding to lunch breaks), banks can have more bank tellers working to handle higher demand, but with some tellers working part-time. The low cost and relative ease of hiring a part-time teller means the bank can anticipate for and adjust the number of tellers to accommodate temporal increases in demand, preventing too much waiting time, and therefore operates as queueing system.



(a) Blocking probability



(b) Scaled utilization



(c) Mean waiting time

Figure 2.5: The comparison of queueing systems with different vacation settings.

2.7 Multiple Options Simulation

For systems that are USST with many options specified, there are several complications that make it difficult to solve with a DTMC. In several of the example reservation systems listed in Table 2.2, such as an airlines, the customer is shown the available resources (or a subset thereof) as part of the scheduling action; therefore, options are not independent of the state of the system. Furthermore, since requests can arrive at any point and each request may take resources that affect subsequent requests and thereby affect the destination state, the order in which requests are received is a critical characteristic. Hence, simulations are used to model USST systems with multiple options.

A relevant solution to such multiple option systems was published in [22]. It showed that for even a small number of alternatives, e.g., 3, the system performed almost as well as one in which the customer is ready to accept any available option, and, better than when only a single option is provided. For example, an SDCS system in which the customer is ready to accept any available option achieves 95% utilization with a call-blocking probability of only 1%, while with POTS, call blocking probability is 23% even when utilization is only 80%. A USST reservation system in which users specified only 1 start time option performed similar to the latter, while with 3 options, the system performed close to the former case.

2.8 Conclusions

The *general reservation system model (GRSM)* developed in this work can serve as a way to uniformly express the key set of characteristics common to all reservation systems. Several examples of reservation systems and queueing systems were used to illustrate the variations on those characteristics, such as whether a customer wants the first available timeslot or wants to specify a timeslot. This work answers the important question of when to use a reservation system instead of a queueing system, as determined by a consideration of revenues (considering load), costs (including servers), and competition. When choosing between a buffered queueing system with an Earliest Start Time (EST) reservation system, if the operational point of the system for profitability is such that the waiting time for customers is large, then a reservation system should be used. When comparing

bufferless queueing systems with User Specified Start Time (USST) reservation systems, if the operational point of the system for profitability is such that the blocking probability is high, then a reservation system should be selected. Finally, we provided five distinct solutions for simplified models drawn from the GRSM under more certain sets of assumptions into which the example systems could be divided, showing the influence of such factors as reservation window size and server vacations.

Chapter 3

Fairness in Multi-Class Book-Ahead Scheduling

3.1 Introduction

It has become clear that widespread collaboration in the scientific community, which can increasingly be characterized by geographically distributed and large-scale projects, requires predictable network service [1]. Predictable network service needed by applications such as large file transfers, remote visualization, and remote instrumentation is typically supported in circuit-switched and virtual-circuit (VC) networks [23]. Significant work in the CHEETAH project [24] focused on various problems related to using circuit-switched networks for file transfers (e.g., [9,21,22,25,26]). In this work, the focus is on an advance-reservation scheduler.

An important consideration for a scheduling system with multiple classes is fairness. The **objective** of the scheduling algorithm is to minimize mean response time while meeting a tunable fairness constraint in terms of the blocked-call ratio and response-time ratio for each class, and also maximizing the number of allocated channels across all time intervals.

The contributions of this work are three-fold:

- a novel three-step optimization-based solution to a multi-class, fairness-considering bin packing problem
- the extension of our BA-First algorithm [21] into a multi-class algorithm, and the formulation of the discrete-time Markov chain (DTMC) model

- a solution to this model of the multi-class BA-First algorithm with sensitivity analysis.

3.1.1 Problem definition

In connection-oriented networks, bandwidth can be shared using an advance-reservation scheduler. The current state-of-the-art in bandwidth scheduling lacks analytical models for multiple classes of calls. In this work, a *multi-class advance-reservation scheduling system* is developed for use in circuit/VC networks. The scheduling algorithm developed in this work applies to any bin packing problem or advance-reservation scenario where a flexible class boundary is allowed. The scheduling system is modeled with a Markov chain. Further, simulations are used to compare different scheduling algorithms that are designed to achieve fairness among classes in terms of call-blocking probability and response time.

This work extends our BA-First algorithm to accommodate multiple classes. BA-First is a single-class algorithm in that it discretizes the capacity C of a link into m equal channels with capacity C/m , and assumes that each request is for one channel and one unit of time τ , referred to as a timeslot. The system maintains a reservation window of size K , which tracks the number of channels reserved for calls in each time interval, for K future time intervals. The reservation capacity of the system is mK timeslots. BA-First fulfills a request with the earliest available timeslot, or rejects the request if a channel is unavailable in the whole K -timeslot advance-reservation window. As not all sessions require identical bandwidth, the original BA-First algorithm needs to be modified to handle multiple types of requests. For example, a remote visualization session may require 300 Mbps to display large amounts of data while a distance-learning session may only require 100 Mbps. Hence we design an advance-reservation scheme to support multiple classes.

3.1.2 Related work

There has been a significant amount of theoretical work on advance-reservation scheduling schemes, [22, 27–77], as well as [21] which this work extends. Our problem definition is unique in that we are concerned with minimizing mean response time while considering fairness experienced by calls of each class in terms of call-blocking probability and waiting time in a multi-class,

centralized scheduler. In most work, either one fairness metric or the other is considered. For example, in routing and wavelength assignment (RWA) work, the fairness metric is primarily call-blocking probability. In multiprocessor/job scheduling, the primary metric is typically the total time to complete a set of jobs (the “makespan”), and, sometimes, the waiting time experienced by jobs. However, since increasing the probability of blocking lowers the waiting time and makespan, and vice versa, this work pursues a book-ahead scheduler that considers both the blocking probability and waiting time, in addition to the objective of minimizing the mean response time.

3.2 System Model

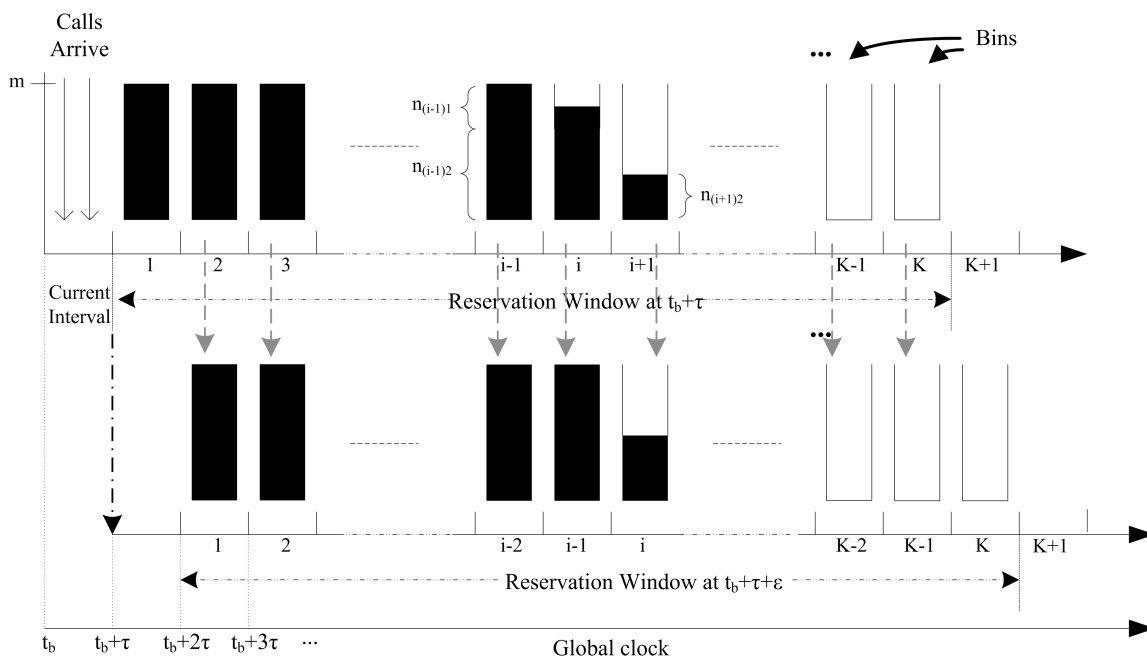


Figure 3.1: An illustration of the BA-First scheduler for multiple classes.

Link capacity C is discretized into m channels with equal bandwidth C/m . To accommodate multiple classes of calls, c_x denotes the number of channels requested by a class- x call. To gain a basic understanding of the impact of multiple classes, we limit our model to a 2-class system (i.e., $x = \{1, 2\}$). The call-arrival process for each class- x is assumed to be Poisson with rate λ_x , and

we define $\beta = \frac{\lambda_2}{\lambda_1}$. We define $r = \frac{c_2}{c_1}$, normalizing c_1 to equal one channel, and $c_2 = r$. Time is discretized into equal-length intervals of duration τ , as shown in Fig. 3.1. Each time interval can be thought of as a “bin,” such that there are K bins in the reservation window, each with capacity m . Though requests can arrive at any instant, service for calls begins and ends only at interval boundaries. This approximation is made for analytical tractability.

At each time interval boundary, the scheduling algorithms schedules calls that have arrived during the just-past interval. It is responsible for achieving fairness between the two classes in terms of call-blocking probability and mean waiting time. To accomplish this, an optimization function determines how the newly arrived calls should be scheduled. Calls that arrived in the interval but could not be accommodated are blocked. In Fig. 3.1, this scheduling algorithm is denoted as occurring between the top set of bins and the bottom set of bins, where the vertical arrows between the top and bottom show how assigned calls “move up” one bin. In the top set, the white space at the top of bin i is an example of the case when there are unallocated channels in a preceding bin, but a subsequent bin has allocated channels to only class-2 calls, shown in the bin $i + 1$. Newly arrived calls are scheduled and any new class-1 calls would fill in that gap in bin i at the top, which becomes a full bin $i - 1$ in the bottom set.

As discussed in Section , a reservation system is required when the number of channels into which a link is divided is relatively small, i.e., around 10. In such systems, the reservation window does not need to be very large to achieve low blocking probability [22]. Therefore, the systems for which this reservation scheduler would be applicable are small, and the optimization can be performed with an exhaustive search.

3.2.1 Non-homogeneous Continuous-Time Markov Chain model

The system state is described by a $2 + 2 \cdot K$ -element vector, $(n_{11}, n_{12}, n_{21}, n_{22}, \dots, n_{K1}, n_{K2}, a_1, a_2)$, where n_{ix} is the number of assigned channels, i being the index for the bin, and x being the index for the class. The elements a_1 and a_2 are the numbers of newly arrived and as yet unassigned class-1 and class-2 calls.

The state space S is defined as

$$S \triangleq \left\{ \begin{array}{l} n = (n_{11}, n_{12}, n_{21}, n_{22}, \dots, n_{K1}, n_{K2}, a_1, a_2) : \\ \forall j, 1 \leq j \leq K, n_{j1} + n_{j2}r \leq m \text{ and } a_1, a_2 \in \mathbb{Z}^+ \\ \text{and } 1 \leq j \leq K, n_{j2} \bmod r = 0 \end{array} \right\},$$

where \mathbb{Z}^+ is the set of all non-negative integers.

Given the difference in how the system transitions states within an interval and how it transitions states at interval boundaries, the system can be modeled as a non-homogeneous continuous-time Markov chain (CTMC). However, it contains an embedded time-homogeneous discrete-time Markov chain (DTMC) if only the time interval boundaries are considered.

3.2.2 Discrete-time Markov Chain model

The discrete time instants in the DTMC are the instants just before the scheduling mechanism is executed.

The transition probability from an origin state n to a destination state n' , denoted $p_{(n,n')}$, is

$$p_{(n,n')} = \begin{cases} P_{A_1}(a'_1) \cdot P_{A_2}(a'_2) & \text{if } n'_{jx} = n_{(j+1)x} + q_{jx}, \text{ where } x = 1, 2 \text{ \& } 1 \leq j \leq K \text{ \& } n_{(K+1)x} = 0, \\ 0 & \text{otherwise,} \end{cases} \quad (3.1)$$

where $P_{A_x}(a)$ is the Probability Mass Function (PMF) of A_x , a Poisson random variable with parameter $\lambda_x \tau$, representing the number of class- x call arrivals within a time interval, and q_{jx} are elements of q_n , as determined by the scheduling algorithm. The scheduling algorithm takes the origin state n and outputs a vector $q_n \triangleq (q_{11}, q_{12}, q_{21}, q_{22}, \dots, q_{K1}, q_{K2})$, $n \in S$, where q_{ix} is the number of additional channels reserved in bin i for a class- x call. The q_n values are determined by the scheduling algorithm in the next section.

3.3 Scheduling Algorithm

The scheduling algorithm can be described as a bin packing problem in which there are a series of bins, each of which may be fully or partially filled, and the algorithm must place new items (the

calls) into the bins in a way that considers both the number of the bin (lower numbered bins equating to lower mean response time) and the relative number of items that are accepted into the bins (versus rejected) to ensure a fair allocation of resources between the two classes. The state space and the various system parameters are input parameters to this bin packing optimization, which is solved independently for each state in the state space. The optimization function, $OPT(n)$, determines the division of free space in all bins to the arrived but unscheduled calls with the objective of minimizing the mean response time while meeting a tunable fairness constraint in terms of the blocked-call ratio and waiting time for each class, and maximizing the number of allocated channels across all bins.

The optimization determines the allocation vector q_n based on the origin state, $n = (n_{11}, \dots, n_{K2}, a_1, a_2)$. For simplicity, n is omitted from the notation, i.e. q_n is denoted as q with n being implicit. This omission applies to the following variables and output metrics as well, all of which are state-dependent:

- q_n^* as q^* , the most fair allocation vector
- R_n as R , the mean response time for admitted calls
- $b_{n,x}$ as b_x , the blocked-call ratio for each class
- $f_{n,b}$ as f_b , the blocked-call ratio fairness metric
- $w_{n,x}$ as w_x , the total response time for each class (the time between call arrival and the time interval boundary when the call is scheduled is ignored)
- $f_{n,w}$ as f_w , the response-time ratio fairness metric
- $f_{n,b}^*$ as f_b^* , the specific value of the blocked-call ratio fairness metric given by q^*
- $f_{n,w}^*$ as f_w^* , the specific value of the response-time ratio fairness metric given by q^*
- V_n as V , the set of candidate q vectors
- L_n as L , the maximum number of channels that can be scheduled.

The omission does not apply to the *system variables*, which are state-independent:

- C , the link capacity
- m , the number of channels with equal bandwidth $\frac{C}{m}$
- c_x , the number of channels requested by a class- x call
- r , with $r = \frac{c_2}{c_1}$ (class-1 call bandwidth is defined as a single channel)
- K , the reservation window.

Further, there are additional system variables used only in the optimization, which are also state-independent:

- ϵ , a fairness-relaxation variable to tradeoff fairness and mean response time for admitted calls
- ω , the weight used to dictate the value of each fairness metric (e.g., $\omega = 0.5$ would value the blocked-call and response-time ratio fairness metrics equally).

The overall optimization problem for a given origin state, $OPT(n)$, is divided into **three optimization subproblems**. Each optimization subproblem acts as a constraint, narrowing down the possible allocations such that the final allocation is optimal, based on the system variables.

- Subproblem 1 finds the allocation vectors that maximize the number of allocated channels across all bins.
- Subproblem 2 finds the most fair allocation vector from among those found in subproblem 1.
- Subproblem 3 finds the allocation vector with the minimum mean response time that achieves fairness within a range specified by ϵ of the fairness of the most fair allocation, as determined in subproblem 2.

In the **first subproblem**, the optimization finds the maximum number of channels that can be allocated given the bin occupancy of state n . The purpose of this step is to minimize blocked-call ratios in a way that does not give preference to class-1 calls. That is, if we simply tried to minimize the number of blocked calls, we would schedule as many class-1 calls as possible, and then schedule

as many class-2 calls as could fit, treating class-2 unfairly. Instead, we find all possible ways to schedule the arrived calls that would equally use the available capacity given the bin occupancy of state n . The objective function of the first subproblem is to find L , the maximum amount of channels that can be allocated given the bin occupancy of state n (limited by the number of arrived but unscheduled calls in n), from which the vector set V is determined. The vectors in the candidate vector set V are then evaluated in subproblems 2 and 3. Each of the vectors in V is a candidate vector because it meets the criteria of maximizing the number of allocated channels, but must be evaluated further for fairness concerns.

Given:

- The state n and the system variables listed above.

Output:

- The set of candidate vectors V defined as $V \triangleq \{v_{jx}, \text{ where } x = 1, 2 \text{ \& } 1 \leq j \leq K\}$

Subproblem 1 objective function:

$$\max L \triangleq \sum_{x=1}^2 \sum_{j=1}^K v_{jx} \quad (3.2)$$

subject to

$$n_{(j+1)x} + v_{jx} \leq m, \text{ for all } x = 1, 2 \text{ \& } 1 \leq j \leq K \text{ \& } n_{(K+1)x} = 0, \quad (3.3)$$

$$\sum_{j=1}^K v_{jx} \leq a_x \cdot c_x, \text{ where } x = 1, 2, \quad (3.4)$$

$$v_{j2} \bmod r = 0, \text{ where } 1 \leq j \leq K, \quad (3.5)$$

$$v_{j1} = 0, \text{ if } n_{jx} + v_{(j-1)x} < m, \text{ where } x = 1, 2, \text{ \& } 2 \leq j \leq K, \quad (3.6)$$

$$v_{j1} + v_{j2} = 0, \text{ if } n_{jx} + v_{(j-1)x} \leq m - r, \text{ where } x = 1, 2, \text{ \& } 2 \leq j \leq K. \quad (3.7)$$

The candidate vector set V can be formed such that every vector $v \in V$, $\sum_{x=1}^2 \sum_{j=1}^K v_{jx} = L$, and for every vector not a member of the set V , $\bar{v} \notin V$, $\sum_{x=1}^2 \sum_{j=1}^K \bar{v}_{jx} < L$. Therefore, every vector in V equally maximizes the number of channels allocated and ensures the system is work-conserving.

The constraint Eq. (3.3) ensures that the allocation does not cause m to be exceeded. The constraint Eq. (3.4) limits the vector v to the capacity requested by the actual number of calls to be scheduled. The constraint Eq. (3.5) ensures that the number of channels in each bin allocated to class-2 calls is a multiple of r . Constraints Eq. (3.6) and Eq. (3.7) ensure that calls are scheduled such that lower bins are filled before higher bins.

In the **second subproblem**, the objective function is to find the vector $q^* \in V$ that yields the most fair allocation. For each state, there is (at least one) most fair allocation, q^* , which results in best-case fairness metrics f_b^* and f_w^* . These values are state-dependent. To avoid having to provide as input absolute values for fairness metrics, which would be state-dependent, we solve this problem by finding the state-dependent best-case values and then use the same fairness-relaxation metric ϵ to allow for a mean response time/fairness tradeoff in subproblem 3.

Given:

- The set of candidate vectors V found in subproblem 1.

Outputs:

- The vector q^* that yields the most fair allocation.
- The best-case fairness metrics f_b^* and f_w^* .

Subproblem 2 objective function:

$$\min (\omega f_b^* + (1 - \omega) f_w^*) \quad (3.8)$$

subject to

$$q^* \in V. \quad (3.9)$$

For q^* , the fairness metrics, f_b^* and f_w^* , are specific (best-case) values of the blocked-call ratio fairness metric, f_b , and the response-time ratio fairness metric, f_w , respectively. The measure from [78] is used to compute the fairness metrics. With this measure, the metrics experienced by calls of each class are expressed with a value of 1 being the most fair and 0.5 being the least fair (since there are two classes), while allowing an intuitive percentage-based fairness-relaxation variable to be used.

The blocked-call ratio fairness metric, f_b , is defined as

$$f_b = \begin{cases} 1, & \text{if } a_1 = 0 \text{ and } a_2 = 0, \\ 1, & \text{if } d_1 = 1 \text{ and } d_2 = 1, \\ 0, & \text{if } d_1 = 0 \text{ or } d_2 = 0, \\ \frac{(2-d_1-d_2)^2}{2((1-d_1)^2+(1-d_2)^2)}, & \text{otherwise,} \end{cases} \quad (3.10)$$

where d_x is the accepted-call ratio for each class, defined as

$$d_x \triangleq \frac{1}{c_x \cdot a_x} \sum_{j=1}^K q_{jx}, \text{ where } x = 1, 2. \quad (3.11)$$

Likewise, the response-time ratio fairness metric, f_w , is defined as

$$f_w = \begin{cases} 1, & \text{if } a_1 = 0 \text{ and } a_2 = 0, \\ 0, & \text{if } \sum_{j=1}^K q_{j1} = 0 \text{ or } \sum_{j=1}^K q_{j2} = 0, \\ \frac{(w_1+w_2)^2}{2(w_1^2+w_2^2)}, & \text{otherwise,} \end{cases} \quad (3.12)$$

where w_x is given by

$$w_x \triangleq \frac{1}{c_x} \sum_{j=1}^K j \cdot q_{jx}, \text{ where } x = 1, 2. \quad (3.13)$$

In the *third optimization subproblem*, the objective function is to choose the vector $q \in V$ that minimizes the mean response time R for admitted calls, but will result in fairness metrics constrained by a relaxed-fairness range determined by the fairness-relaxation metric. Mean response time is the average time until a call has been completed (the time between call arrival and the time interval boundary when the call is scheduled is ignored), defined as

$$R \triangleq \frac{w_1 + w_2}{\sum_{j=1}^K q_{j1} + \sum_{j=1}^K \frac{q_{j2}}{c_2}}. \quad (3.14)$$

Given:

- The set of candidate vectors V found in subproblem 1.
- The fairness metrics f_b^* and f_w^* found in subproblem 2.

Outputs:

- The vector q that yields the minimum mean response time.
- The fairness metrics f_b and f_w , and minimum mean response time R .

Subproblem 3 objective function:

$$\min R \quad (3.15)$$

subject to

$$\omega f_b^* + (1 - \omega) f_w^* \geq \omega f_b + (1 - \omega) f_w \geq (\omega f_b^* + (1 - \omega) f_w^*) - \epsilon \quad (3.16)$$

The constraint Eq. (3.16) allows for a tradeoff between relaxing fairness and minimizing the mean response time by varying the fairness scaling factor, ϵ . For q^* , the most fair allocation, the mean response time would be R^* . The scaling factor allows a less fair solution to be found that would result in fairness metrics f_b and f_w within the range given in Eq. (3.16), for which there might be an allocation that yields $R < R^*$. By varying ϵ within the range $0 \leq \epsilon < \infty$, a tradeoff between fairness and mean response time can be explored. For $\epsilon \rightarrow 0$, the allocation will be more

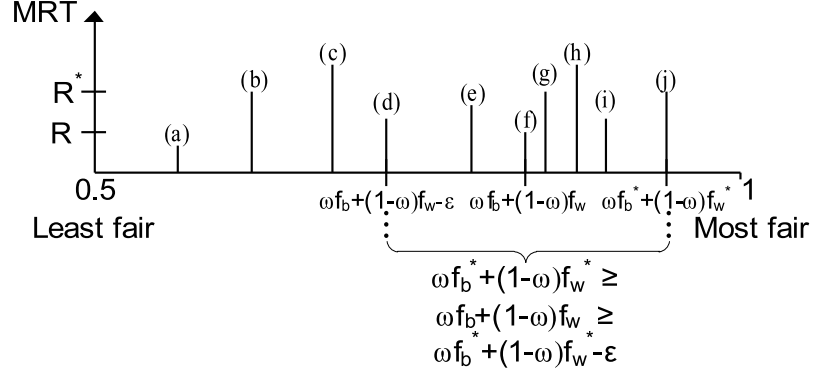


Figure 3.2: Optimization example; MRT: Mean Response Time.

fair at the possible expense of a longer mean response time, while for $\epsilon \rightarrow \infty$, the shortest mean response time will be experienced but with decreasing concern for fairness.

In summary, the output of the overall optimization is the vector $q \triangleq q_{11}, q_{12}, \dots, q_{K1}, q_{K2}$ detailing the allocation of channels to new calls, the fairness metrics and mean response time of the most fair allocation, f_b^* , f_w^* , and R^* , and the fairness metrics and mean response time of the relaxed-fairness optimization, f_b , f_w , and R .

For example, in Fig. 3.2, (a) through (j) denote vectors in set V , which are determined by subproblem 1. The vector (j) denotes the vector which is found in subproblem 2 to be the most fair allocation, with fairness f_b^* and f_w^* , and mean response time R^* . Therefore, $q^* \equiv (j)$ in Fig. 3.2. In subproblem 3, the relaxed-fairness constraint has a range given by Eq. (3.16), which allows vectors (d) through (i) to be considered. Minimizing for R as per Eq. (3.15), the optimization determines that the vector (f) has the minimum R . Therefore, $q \equiv (f)$ in Fig. 3.2.

3.4 Model Solution

The scheduling algorithm determines how newly arrived calls are placed into bins. The metrics such as b_x and f_w from above are solved for each state as described, and are now used in this section to solve the overall model. The system performance metrics are defined as:

- \mathfrak{R} , the mean response time (MRT)

- B_x , the call blocking probability for each class
- F_b , the call blocking fairness metric
- W_x , the waiting time for each class
- F_w , the response time fairness metric
- F_t , the overall fairness metric
- U , the long-run utilization of the system

In order to calculate the above metrics, the steady-state probabilities, denoted by the vector π , of the DTMC must first be calculated. As a result of the optimization process, the other parameters necessary to compute the system performance metrics are pre-determined. That is, at the moment of the state transition, the waiting time of class-2 calls, for example, is explicitly known, since, based on the state, the optimization will determine the sole possible transition. Since the states with no call arrivals should not be factored into the metrics, simply multiplying the steady-state probabilities π by the transpose of the state-dependent optimization-determined metrics is not sufficient. Instead, an $|S|$ -element vector a_e is created such that its n th element is 1 only if at least one call of either class arrived in state n , and 0 otherwise. Thus, the optimization-determined metrics are solved as follows:

- $\mathfrak{R} = \frac{\pi R^T}{\pi a_e}$
- $B_x = \frac{\pi b_x^T}{\pi a_e}$
- $F_b = \frac{\pi f_b^T}{\pi a_e}$
- $W_x = \frac{\pi w_x^T}{\pi a_e}$
- $F_w = \frac{\pi f_w^T}{\pi a_e}$
- $F_t = \frac{\pi f_t^T}{\pi a_e}$

The MRT \mathfrak{R} is used here to measure the elapsed time from the interval boundary at which the call is scheduled to the interval boundary at which the call's service is completed. Recall that requests can arrive at any instant but the scheduling action occurs only at interval boundaries (calls received between interval boundaries are held and scheduled as a batch). Therefore, there is some time experienced by each request, spent waiting to be scheduled. This time is the “cost” of doing batch processing, and is both unavoidable as well as uncontrollable by the algorithm; hence, it is not considered.

The utilization of a given state is calculated by the occupancy of the first bin divided by the number of channels, so we can define the per-state utilization $u_n = \frac{n_{11}+n_{12}}{m}$. Therefore, the utilization of the system is given by $U = \pi u_n^T$.

3.5 Evaluation Approach

There are two steps in our evaluation approach. First, we execute the scheduling algorithm for the given system parameters to obtain the q allocations for all possible states, which determines how newly arrived calls are placed into bins. Second, we solve the system model using “lookups” to find the computed q allocation for a given state. The model could be solved in two ways. For small systems, we solved it both analytically and with simulations, and for larger systems we used simulations. The two models were effectively verified through a comparison of the output from the analytical solution with the output from simulations for the same-scale systems. The results from the two models were nearly identical.

Recall from Section 3.2 that the number of newly arrived calls for each class are contained in the system state vector, making the state space infinite. For tractability, an upper limit for the number of call arrivals per class per time interval was established at $7 * m$. At that limit, there is a balance between very small probabilities of exceeding that limit (between 10^{-10} and 10^{-16} depending on β and r at the highest system load evaluated) and keeping the state space to a limited size for which steady-state probabilities could be found using Matlab on a large-memory cluster node with 32GB of memory. The state space increases to untenable levels quickly as m and K are increased.

Therefore as stated before, the model was solved analytically for a small system ($m = 4, K = 2$), and simulations were used to perform sensitivity analysis.

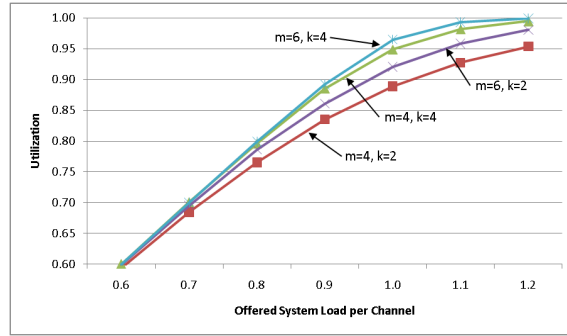
For our evaluation, we chose to run the optimization “a priori” instead of invoking it for the given state within the simulation. The number of per-class call arrivals in each interval is the only random number generated. Therefore, we pre-computed all the possible states, such that from within the simulation it required only a lookup to the optimization’s output to determine the proper transition. This reduced the processing time to run a simulation, which was still considerable despite using parallel processing on a Linux cluster. For example, for an $m = 6, K = 4$ system, each change in a variable required approximately 16 hours on a high-end compute node, and hence changing a variable over a range required 16 hours multiplied by the number of points in the range. Each experiment consisted of 80 simulation runs, and for each combination of system parameters, the experiment was repeated 500 times.

3.6 Results

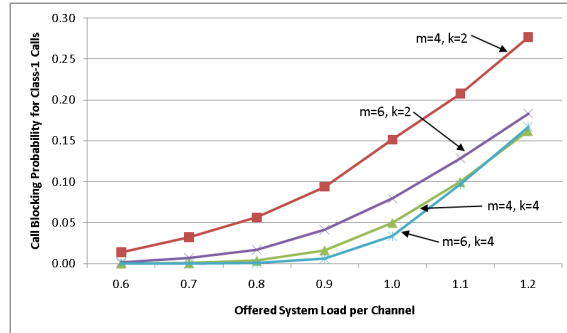
This results section evaluates the impact of the different variables, explores the tradeoffs between fairness and performance, and compares the multi-class BA-First and first-come first-served (FCFS) algorithms.

3.6.1 Sensitivity analysis on system variables m and K

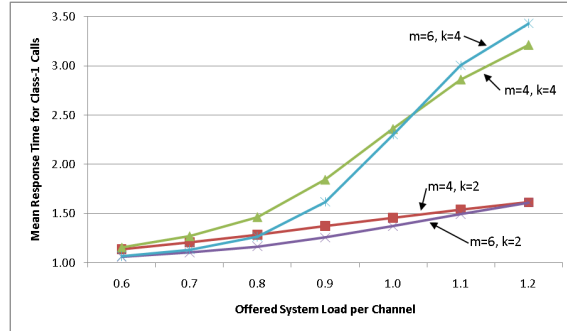
The system variables m and K are basic design parameters that, holding all other variables equal, form the fundamental tradeoff between a quicker MRT \mathfrak{R} and a lower call blocking probability. This is best demonstrated by looking at only one class. Figs. 3.3(a) and 3.3(b) show there is a somewhat counter-intuitive result in that the utilization increases as the reservation capacity of the system increases, but likewise the call blocking probability (CBP) for class-1 calls is lower for larger reservation capacity systems. Typically, a higher CBP is incurred to gain a higher utilization, and within each system this is of course true. Comparing across systems reveals that increasing the reservation capacity simultaneously increases utilization and decreases CBP. However, this is



(a) Utilization

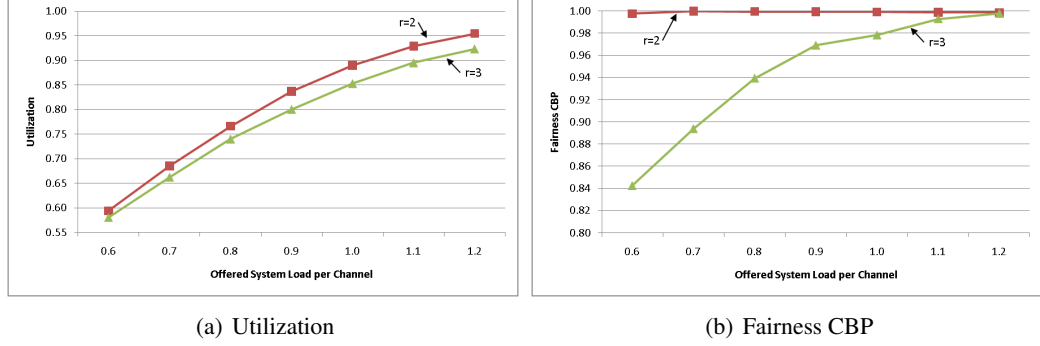


(b) Call Blocking Probability



(c) Mean Response Time

Figure 3.3: Sensitivity analysis on system variables m and K

Figure 3.4: Sensitivity analysis on system variable r

an artifact of the overall relative smallness of the reservation capacities chosen: one can note the disparity is larger between the $m = 4, K = 2$ and $m = 6, K = 2$ systems than between the $m = 4, K = 4$ and $m = 6, K = 4$ systems, even though in both comparisons the larger system is 50% larger than the smaller system.

The lower mean response time experienced in a $K = 2$ system compared to a $K = 4$ system is shown in Fig. 3.3(c). The systems with larger reservation windows incur larger mean response times but gain increased utilization. The crossover point for the $K = 4$ systems when the offered system load per channel exceeds 1 is the result of the slightly higher degree of CBP fairness being achieved by the larger system. The larger number of calls of both classes arriving in the larger system affords the optimization a greater opportunity to enforce CBP fairness, but the cost is a higher MRT.

3.6.2 Sensitivity analysis on system variable r

The graphs in Figs. 3.4(a)-3.4(b) illustrate the effect of the system variable r . For a larger r , the system has a more difficult time achieving fairness, as calls of the larger class are unable to be accommodated in situations if there are less than r channels available in a bin. The graphs are for an $m = 4, K = 2$ system, which makes the difference between $r = 2$ and $r = 3$ significant. As the load increases, however, the optimization has more calls to place and therefore more opportunity to enforce fairness, as seen in Fig. 3.4(b).

3.6.3 The impact of the fairness weight ω

For $\omega \rightarrow 0$ (favors F_w) or $\omega \rightarrow 1$ (favors F_b), the system is either very fair in terms of one metric or the other, but not both. The MRT shown in Fig. 3.5(a) reveals that either choice does not significantly impact the MRT. The graphs are for an $m = 6, K = 4, r = 2$ system. With $\omega = 1$ in Fig. 3.5(b), the optimization only considers CBP fairness. The large dip is somewhat misleading; the fairness of 1 before the dip is because there were no calls that needed to be blocked, given the large reservation capacity and low per-channel load, but with the slightly higher per-channel load of 0.7 an extremely low percent of calls needed to be blocked and at these low blocking rates it was difficult to ensure CBP fairness. For lower loads, the optimization has fewer combinations from which to choose, leading to a less fair outcome.

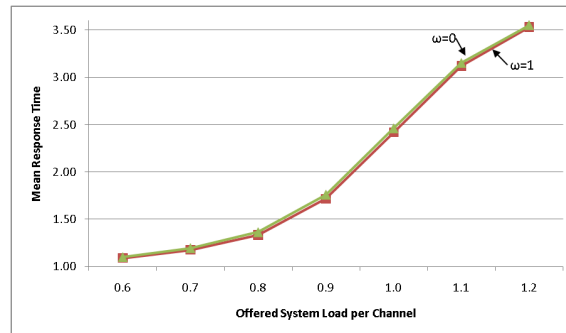
3.6.4 Relaxing the fairness objective by the optimization variable ϵ

The cost for fairness is a higher MRT \mathfrak{R} . The fairness-relaxation variable allows the optimization to schedule calls in a way that is less fair but would result in a lower MRT. In Figs. 3.6(a)-3.6(c), the system is $m = 6, K = 4, r = 2$ and $\omega = 0$ so the optimization is solely concerned with response-time fairness. By increasing ϵ , up to nearly 20% of transitions are chosen that have a lower MRT but are less fair. The decrease in fairness is evident in Fig. 3.6(b), but the corresponding decrease in MRT is very slight, amounting to only a 1% decrease in MRT at high load.

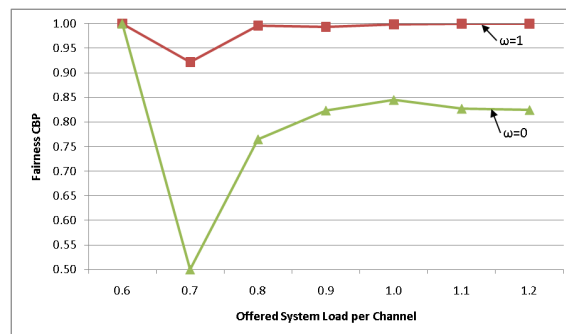
3.6.5 Mis-matched capacity demand

Under normal circumstances, over the long run, we would expect the algorithm to operate in conditions where the requested capacity by each class is fairly similar. The preceding results all follow that assumption. However, it is important to test how well the algorithm performs when there is a disparity in the requested capacity between the classes, which is shown in Fig. 3.7. The graphs are for an $m = 6, K = 4$ system.

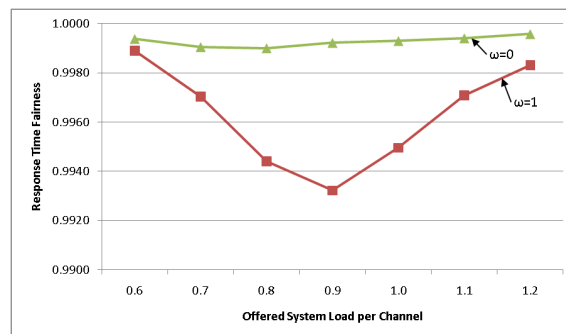
Starting with Fig. 3.7(b), at an offered system load per channel of 0.7, for a given r , the system is increasingly more fair as $\beta = \frac{\lambda_2}{\lambda_1}$ increases, showing that the more calls of class-2 there are relative



(a) Mean Response Time

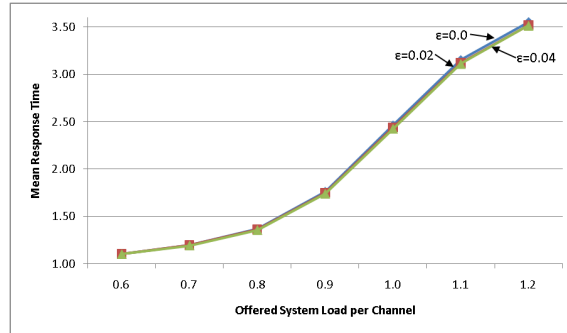


(b) Fairness CBP

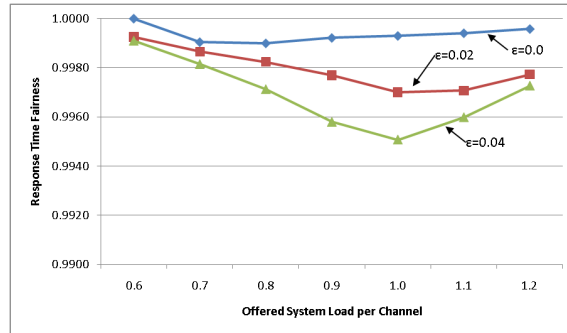


(c) Fairness Response Time

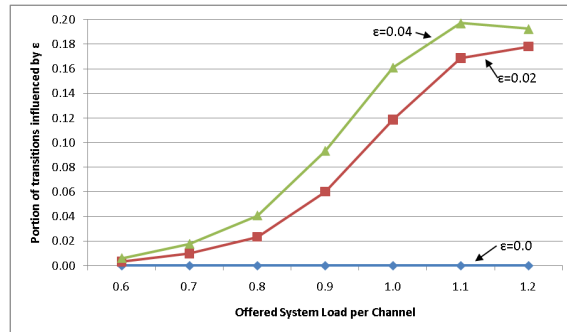
Figure 3.5: The impact of the fairness weight ω



(a) Mean Response Time



(b) Fairness Response Time

(c) Portion of transitions influenced by ϵ Figure 3.6: Relaxing the fairness objective by the optimization variable ϵ

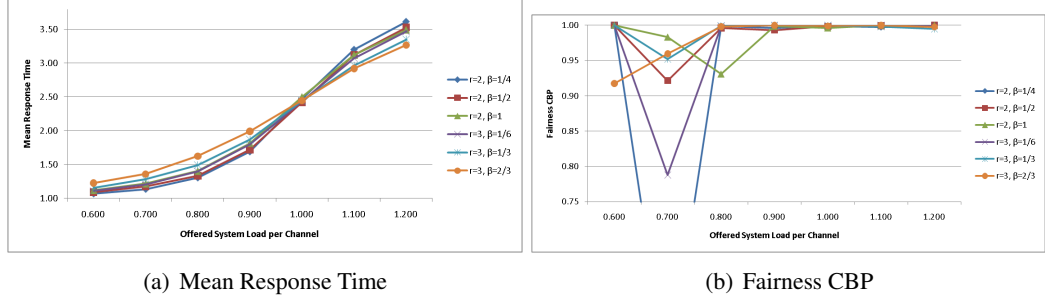


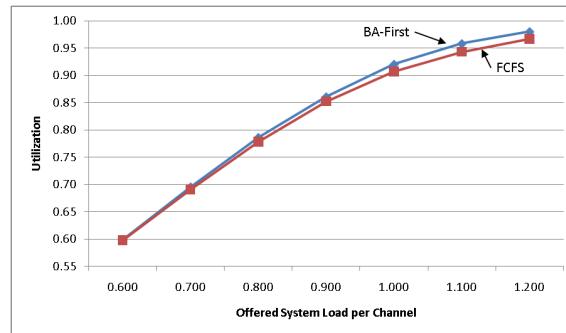
Figure 3.7: Mis-matched capacity demand

to calls of class-1, the more fair the algorithm will be. This follows rather clearly from the fact that the more availability of calls of either class, the more opportunity there is for the optimization to select a fair allocation.

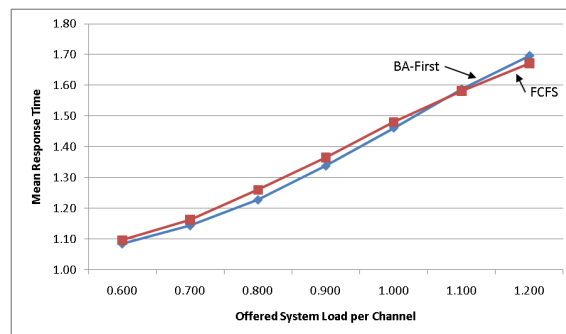
Somewhat less clear, however, is the complete crossover in order from lowest to highest mean response time that occurs at an offered system load per channel of 1.0 in Fig. 3.7(a). For a per channel load of less than 1.0, the order is, from lowest to highest, $\beta = \frac{1}{4}$ to $\beta = 1$ for $r = 2$, then $\beta = \frac{1}{6}$ to $\beta = \frac{2}{3}$ for $r = 3$. Greater than 1.0, the ordering is precisely opposite. This is directly related to the availability of calls of either class for the optimization to allocate, as well as the selection of $\omega = 1$ for this example. Less than a per channel load of 1.0, with very few blocked calls, the optimization will default to placing the higher capacity class-2 calls in higher bins. The larger r , or the more class-2 calls there are, the more likely they will be allocated to higher bins than the majority of the class-1 calls, thereby increasing the MRT. Conversely, for per channel load greater than 1.0, the system will be operating at a very high utilization, and the results are heavily influenced by the likelihood that only the final bin will have space. In that case, the more class-2 calls that are allocated, or the larger the r , the lower the MRT will be, since each class-2 call prevents multiple class-1 calls at the same high MRT from being allocated.

3.6.6 Comparison of Multi-class BA-First to FCFS

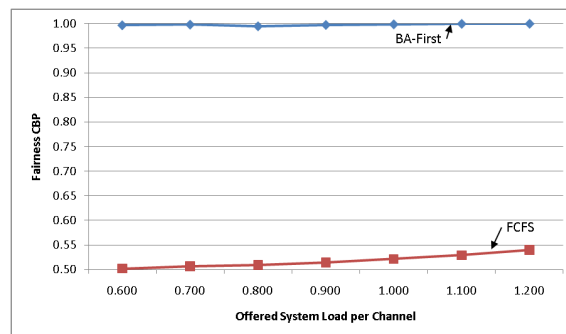
In this section the multi-class BA-First algorithm is directly compared to a non-optimizing algorithm, in which calls are simply scheduled according to the order in which they arrive (FCFS).



(a) Utilization



(b) Mean Response Time



(c) Fairness CBP

Figure 3.8: Comparison of Multi-class BA-First to FCFS

Though it is obvious the FCFS algorithm will be less fair, the performance tradeoff incurred to realize fairness is not so obvious. Fig. 3.8(a) shows the utilization as the per-channel load increases. At low load, the algorithms are nearly equal, since few calls are blocked in either case. As load increases, the ability of the multi-class BA-First algorithm to fill more available space (i.e., does not leave gaps as FCFS might) leads to a higher utilization. Along with that, though, the work-conserving aspect also causes the cross-over point at a per-channel load of 1.1 in Fig. 3.8(b). In high load, when enough calls have been received for the optimization to fully fill the reservation window with greater frequency (leading to higher utilization), that also creates a higher MRT as the last bin is more likely to be filled, compared to FCFS.

3.7 Conclusions

The scheduling algorithm developed in this work applies to any bin packing problem or advance-reservation scenario where a flexible class boundary is allowed. The algorithm determines how newly arrived calls are allocated, and those allocations for all states are used by the analytical model and simulations as solutions to the system model. Sensitivity analysis completed with the use of simulations showed how the system performs with varied inputs. We found that with more channels in a bin and an increased number of bins, the larger system affords the optimization a greater opportunity to enforce fairness, but the cost is a higher mean response time. The system has difficulty ensuring fairness when the per-channel loads are lower, as there are fewer possible choices for the algorithm to make. It enforces fairness quite well at higher loads, and the fairness can be relaxed by a tunable variable to achieve lower mean response time by decreasing the fairness. Finally, the algorithm is compared to a non-optimizing first-come first-served algorithm to illustrate the performance tradeoff incurred in achieving fairness.

Chapter 4

An in-depth cross-layer experimental study of transport protocols over circuits

4.1 Introduction

Recently, both research-and-education networks (RENs) and commercial networks have added a dynamic circuit service. With this service, users can request and obtain dedicated bandwidth for short durations (on the order of minutes to hours). RENs are motivated by eScience applications such as the high-energy physics projects enabled by the Large Hadron Collider in CERN, Geneva [79]. Commercial providers refer to this service as “bandwidth-on-demand.”

The deployed dynamic circuit networks use newly developed equipment called multi-service provisioning platforms (MSPPs), which are fundamentally SONET/WDM circuit switches with additional Ethernet line cards that implement Ethernet-to-SONET or Ethernet-to-WDM frame mapping technologies. These mappings have to be provisioned prior to data transfer, making the end-to-end communication path effectively a virtual circuit. The addition of Ethernet line cards to circuit switches has enabled the use of dedicated-bandwidth end-to-end circuits between computers. The end-to-end circuits consist of Ethernet segments between computers and MSPPs, and wide-area SONET/WDM circuits between MSPPs.

The main application for end-to-end high-speed dedicated circuits is large file transfers. Toward supporting this application, the *problem statement* of this work is to *design, implement, and test*

a *transport protocol* for use across dedicated end-to-end circuits. More specifically, the type of circuits considered here are what we term **mismatched-rate circuits**. Such circuits are created when a high-speed port, such as a GigE (gigabit per second (Gbps) Ethernet) port, is mapped on to a lower-rate circuit, e.g., a 400Mbps virtually concatenated SONET circuit. First, the MSPP technology allows for such mismatched-rate mappings. Second, the reason for provisioning such mismatched-rate circuits is as follows. For file transfers, disk-write speeds are often the bottleneck. They are typically on the order of 400Mbps per host. This means even if the link from the host to the MSPP is a GigE, by making the wide-area SONET circuit between the MSPPs just 400Mbps, we can ensure that there is no under-utilization of wide-area circuits. Therefore, the *goal of this work is to design and configure transport protocols for use on such mismatched-rate circuits*.

Our key **contribution** is a characterization of the interaction between multiple layers: TCP, switch (MSPP) buffers and Ethernet layer. Rate mismatches across circuits are handled by enabling ON/OFF flow control in the MSPP Ethernet line cards. This, in turn, causes a Layer-2 parameter, *txqueuelen*, the size of the output queue (*qdisc*), at the sending host to play an important role. The TCP congestion window is limited by the sum of *txqueuelen*, the *Pause ON* threshold set in the switch buffer, and the product of the circuit rate and round-trip propagation delay.

Using recently available features in Linux, a new tool, *BWdetail*, is developed to capture these complex interactions between (i) transport protocols, (ii) Ethernet drivers at hosts, (iii) switch buffers, and (iv) Layer-2 flow control. This in-depth study shows how TCP layer parameters, Ethernet layer parameters at hosts and switches, and switch buffer allocations should be configured to achieve better performance and fully leverage circuit infrastructure advances. Such an in-depth characterization can only be obtained with experimental methods, and not with analytical modeling and/or simulation. Therefore, this work is an **experimental study** not an abstracted modeling study.

4.2 Background

Section 4.2.1 describes newly deployed circuit networks, and more detail is provided on the MSPPs used in these networks in Section 4.2.2. The likely scenario of mismatched-rate circuits is de-

scribed in Section 4.2.3. Sections 4.2.4 and 4.2.5 provide the necessary background on interactions between TCP, switch buffers and the Ethernet layer.

4.2.1 Dynamic circuit service deployment

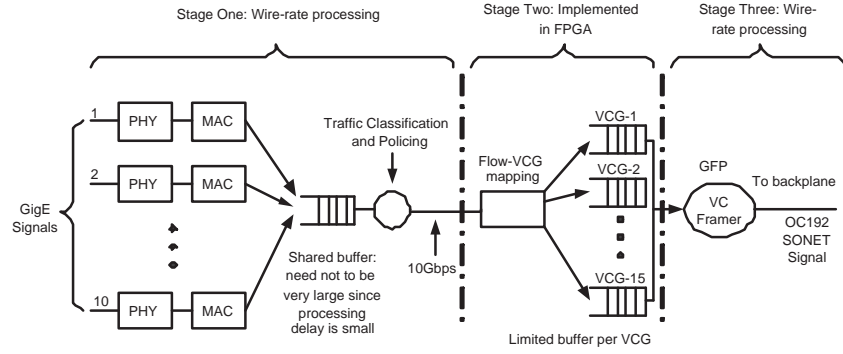


Figure 4.1: Internal structure of a Gigabit/sec Ethernet (GigE) line card in an Ethernet-SONET MSPP

Three US-wide core (backbone) REN providers, Internet2 [80], the Department of Energy's (DOE's) ESNet4 [81], and NLR [82], offer a dynamic circuit service in addition to their IP-routed service offering. Similar deployments include Europe's AutoBAHN [83], Japan's JGN2plus [84], and Canada's UCLP [85], among others. Motivated to support eScience applications that require rate guarantees, these networks provide users the ability to request on-demand high-speed circuits for short durations (on the order of minutes to hours). High-energy physics, climate studies, bio-informatics, and a number of other eScience projects are using these dynamic circuit services.

As an example, consider the Internet2 network. Its dynamic circuit network is referred to as *Interoperable Ondemand Network (ION)*. The ION consists of twenty-two Ciena CD-CI systems, which are Ethernet/SONET MSPPs. The IP-routed network consists of Juniper routers. Two wavelengths are leased by Internet2 on its entire national footprint, one to interconnect ION nodes, and the second to interconnect IP routers. Connectors, which are regional RENs, connect via separate links to these two networks. The Internet2 dynamic software suite [86] provides the control-plane software, which includes a bandwidth reservation scheduler, through which individual users or application software can request bandwidth, specifying start times, bandwidth amount, and durations.

4.2.2 Equipment used in dynamic circuit networks

Given the ubiquity of Ethernet, circuit switch vendors have added Ethernet line cards to their SONET/WDM circuit switches to create the so-called **MultiService Provisioning Platforms (MSPPs)**. The operation of an Ethernet line card designed for an MSPP is described below.

Fig. 4.1 shows the internal structure of a 10-port GigE line card in an Ethernet/SONET MSPP. It has Ethernet ports on the front end (left edge of Stage One in Fig. 4.1) and SONET (virtual) ports on the back end (right edge of Stage Three in Fig. 4.1) through which the card is connected to the backplane. The backplane thus brings SONET signals from all interface cards, both Ethernet and SONET, to the switch fabric card where the signals are cross-connected according to programmed configuration information.

Stage One classifies frames and does traffic policing on “Layer-2 flows,” which are defined as all packets entering the line card at a specific physical port, or tagged with specific virtual LAN identifiers (VLAN IDs). Virtual-concatenation groups (VCGs) are configured as shown in Stage Two of Fig. 4.1. An example VCG is a OC3c-7v, which consists of seven virtually concatenated OC3 signals, for an aggregate rate of 1.09Gbps. The “flow-VCG” mapping shown in *Stage Two* allows for the mapping of an Ethernet port or VLAN based Layer-2 flow to a VCG. As the VCG rate does not necessarily have to match the port or VLAN rate, mismatched-rate circuits can be created. *Stage Three* implements generic framing procedure (GFP) [87] in a VC framer to map Ethernet frames onto SONET frames. In Ethernet-WDM MSPPs, the ITU-T G.709 digital wrapper standard is implemented to map Ethernet frames on to wavelengths.

4.2.3 Mismatched-rate circuits and Layer-2 flow control

In a mismatched-rate circuit, Ethernet frames could arrive at a rate faster than the rate of the SONET circuit to which the incoming port is mapped. The rate mismatch will cause the VCG buffer, shown in Stage Two of Fig. 4.1, to fill. The GigE line card in the MSPP issues Layer-2 *Pause* frames to stop the sender, in accordance with IEEE 802.3x ON/OFF flow control, to prevent buffer overflows and corresponding packet losses. The switch sends *Pause* frames when the *Pause ON* threshold (typically configurable) is crossed. Within each *Pause* frame, a field specifies the time period during

which the sender should not transmit any frames. At the end of the specified time period, or after receiving a *Pause* frame with zero in the time field, the sender resumes normal transmission of frames. A switch sends the *Pause* frame with zero in the time field when the buffer size falls below the *Pause OFF* threshold.

4.2.4 Impact of TCP congestion control on switch buffer occupancy

TCP congestion control consists of two phases: slow start and congestion avoidance. In the slow-start phase, TCP congestion window $cwnd$ increases at an exponential rate. In this phase, if the size of the congestion window is x , $\frac{x}{2}$ packets will be queued in switch/router buffers. A full explanation of why this occurs is given in [88].

Similarly, during the congestion avoidance phase, there will always be $(cwnd - 2t_{prop}r)$ number of packets queued inside switch/router buffers, where t_{prop} is the one-way propagation delay and r is the bottleneck link rate. This assumes that the receive buffer is set to a large value to avoid becoming the bottleneck. As acknowledgments are received by the sender, the congestion window keeps growing by one packet per round-trip time (RTT). Eventually, for long flows, there could be packet loss once the congestion window grows beyond the *capacity* of the end-to-end path between the end hosts, where *capacity* is the sum of $2t_{prop}r$ and switch/router buffer size.

4.2.5 TCP-Ethernet layer interaction within hosts

An application writes data into the TCP *send* buffer by calling the *write()* system call. The TCP layer assembles a segment when data is available in the *send* buffer. Each segment is pushed down to the IP layer for transmission. The IP layer enqueues each packet in an output queue (*qdisc*) associated with the appropriate network interface card (NIC). The size of the *qdisc* can be modified by assigning a value to the *txqueuelen* variable associated with each NIC card. After a packet is successfully queued inside the output queue, the packet descriptor is then placed in a output ring buffer. When there are packet descriptors queued in the ring buffer, the device driver invokes the NIC direct memory access (DMA) engine to transmit packets onto the wire.

4.3 Prior work

There has been significant work developing protocols to offer high performance for data transfers, e.g., NETBLT [89], UDT [90], RAPID [91], and TCP variants such as HSTCP [8] and FAST [92]. However, none of these protocols were designed for circuits.

One protocol which was specifically designed for use on circuits is a variant of TCP called Circuit-TCP (CTCP) [9]. The key modifications to TCP are that the congestion window is set to a fixed value equal to the bandwidth-delay product of the path and that the TCP slow start and congestion avoidance algorithms are disabled. The rationale is that by reserving bandwidth and provisioning a dedicated circuit between two end hosts (or computer clusters), there can be no congestion-related losses in the data plane. Losses, if any, are due only to link errors, which is no reason to change the sending rate. In the experiments presented in [9], the effects of running CTCP on mismatched-rate circuits, interaction with Layer-2 ON/OFF flow control, and MSPP's buffering capability were not studied. These aspects are the main focus of this chapter.

4.4 New CTCP Implementation and BWdetail

A limitation of the CTCP implementation used in [9] is the need to patch the Linux kernel. To make CTCP more accessible to circuit users in the deployed dynamic circuit networks, in this work a new CTCP implementation has been developed. Starting with Linux 2.6.13, congestion control algorithms can be dynamically loaded in as kernel modules. This capability is leveraged in the new CTCP implementation. In CTCP, a configurable parameter sets the congestion window value, which remains unchanged for the duration of the connection. Thus, slow start and congestion avoidance algorithms are disabled in CTCP. Also, the TCP feature that resets the congestion window to the restart window value when idle periods are larger than one retransmission timeout [93] is disabled. By fixing the congestion window size, the number of unacknowledged packets allowable is "capped," enforcing a strict upper limit on the resources (buffer space, bandwidth, host memory, etc.) a CTCP flow will use.

To characterize the behavior of CTCP, BIC TCP and Reno TCP across circuits, a new tool called

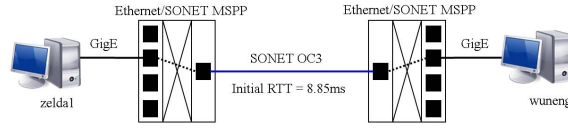


Figure 4.2: Experimental Setup

BWdetail was developed in this work [94]. BWdetail is similar to Iperf [95], in that it is used to measure throughput. BWdetail, however, additionally provides detailed information relating to the TCP stack without requiring kernel modification. BWdetail takes advantage of the exposed kernel information to record information such as the current congestion window. No kernel modifications such as Web100 are necessary to record TCP stack information.

4.5 Experiments

4.5.1 Experimental Setup

Fig. 4.2 shows the experimental setup. Two end hosts, *Zelda1* and *Wuneng*, are connected to GigE interfaces of MSPPs. A SONET circuit is provisioned between two MSPPs and the Ethernet ports are mapped to the SONET circuit. Specifically, the Ethernet/SONET MSPPs shown in Fig. 4.2 are Sycamore SN16000 systems, and the hosts are off-the-shelf Linux machines.

BWdetail is used to execute memory-to-memory data transfers of variable sizes from *Zelda1* to *Wuneng* using Reno, BIC, or CTCP as the transport protocol. The experiments are performed with a SONET circuit rate of OC3. For each of these transfers, the instantaneous throughput is recorded every 100 ms, and the *cwnd* and RTT every 5 ms. The entire transmission is also captured using *tcpdump*. The initial RTT between these two hosts, i.e., with no buffered packets, is 8.85 ms. The *txqueuelen* is set to 1000 packets. The MSPPs are configured with a *Pause ON* threshold of 1 MB and a *Pause OFF* threshold of 512 KB. Maximum frame size is 1500 bytes, which makes emission delay at OC3 rate approximately $80\mu\text{s}$, and correspondingly, the initial bandwidth-delay product (BDP), i.e., with no buffered packets, is 100 packets. The receive-side buffer is set to a large value so that the effects of congestion window growth can be isolated.

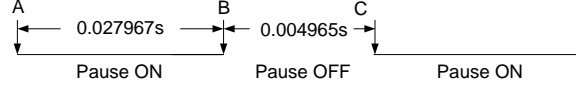


Figure 4.3: Circuit Rate = OC3 (155 Mbps)

4.5.2 Findings on cross-layer interactions

Our key finding is about the interaction between TCP congestion control, buffering within switches, and data-link layer flow control. Given the presence of ON/OFF flow control at the data-link layer, whereby *Pause* frames are sent to the sending host as the switch (MSPP) buffer fills up, there are no packet losses. Instead, when the sending end host's Ethernet layer receives a *Pause* frame, it stops transmissions. This causes the Ethernet-layer output queue (*qdisc*) to fill up because the TCP layer keeps adding packets to this queue as acknowledgments are received.

When the Ethernet-layer output queue (*qdisc*) is full, an attempt to enqueue a packet generates a local-congestion event, which is propagated upward to the TCP layer. The TCP congestion-control algorithm then enters into the Congestion Window Reduced (CWR) state, and reduces the congestion window by one every other acknowledgement (known as *rate halving*).

The *cwnd* value at which the CWR state is reached is given by:

$$cwnd_{CWR} > 2t_{prop}r + PauseON + txqueuelen \quad (4.1)$$

where t_{prop} is one-way propagation delay, r is the circuit rate, *PauseON* is the *Pause ON* threshold set in the bottleneck switch (MSPP) buffer, and *txqueuelen* is the size of the output queue (*qdisc*) at the sending host's Ethernet NIC.

4.5.3 Results

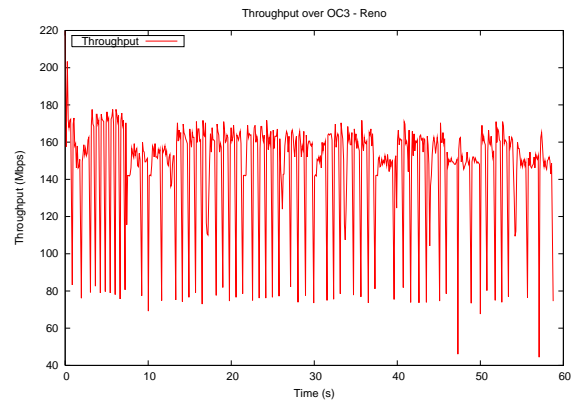
The periodic nature of *Pause* frames is explained by examining the lengths of the *Pause ON* periods, *Pause OFF* periods, and the number of *Pause* frames seen in each of these *Pause ON* periods (Fig. 4.3). During the *Pause ON* periods, the sender NIC sits idle, while the MSPP drains data from its buffer at the SONET circuit rate. For the OC3 rate (Fig. 4.3), the average length of the *Pause ON*

period is 0.027967s, during which time the MSPP drains 529 KB of data. This causes the MSPP buffer to fall below the *Pause OFF* threshold. As the sending host resumes transmission after the *Pause ON* period, it transmits data at a 1 Gbps rate during the following *Pause OFF* period. This causes data to be queued in the switch buffer at (1000 – 155) Mbps rate, resulting in 512 KB of data queued in 0.004965s. Hence, the switch crosses the *Pause ON* threshold again, and enters into another *Pause ON* period.

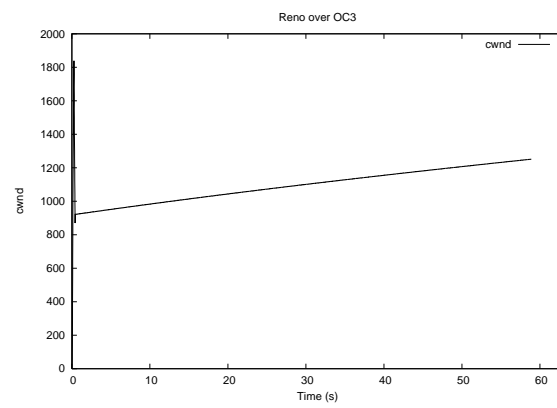
In Figures 4.4, 4.5, and 4.6, the throughput, *cwnd*, and RTT are shown for a 1 GB transfer over an OC3 circuit using Reno, BIC, and CTCP. Due to regular *Pause ON* and *Pause OFF* periods, the throughput graph for Reno (Fig. 4.4(a)) shows drops at regular intervals. The **average throughput** of the entire 1 GB transmission is 142 Mbps for **all three** transport protocols. As predicted by Eq. 4.1, the congestion window plot (Fig. 4.4(b)) for Reno after the initial slow-start period, and at regular intervals in BIC (Fig. 4.5(b)), drops after crossing 1750 packets as TCP enters the CWR state. In these experiments, the NIC transmit-queue length (*txqueuelen*) is set to 1000 packets, the switch buffer can hold 700 packets (1 MB *Pause ON* threshold) before entering into a *Pause ON* period, and there are approximately 50 packets on the wires; therefore, a local congestion event is received from the NIC buffer when the congestion window crosses 1750 packets. Hence, when TCP enters the CWR state, both Reno and BIC reduce their congestion windows by half shortly after reaching 1750 packets (the amount of time it takes to reduce the congestion window by half is very small, given the relatively short RTT, and appears in the figures to be instant, but is in fact not).

Following the CWR state and rate halving, Reno enters the congestion avoidance state, and increases the congestion window by only one packet in each RTT. Being more aggressive than Reno, BIC quickly recovers from the *cwnd* reduction. If the BDP of a path is large, this slow increase of the congestion window will hurt the overall throughput [8]. No drops in *cwnd* occur with CTCP, as this is held constant.

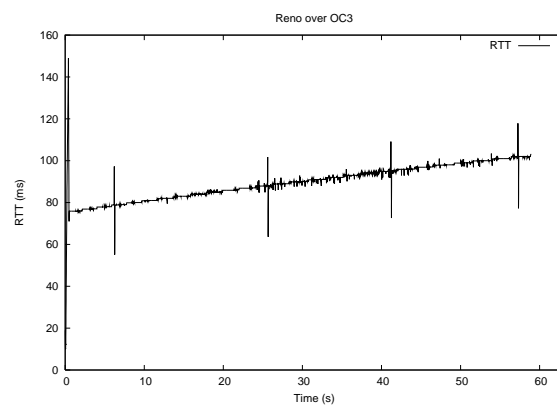
The difference in RTT between Reno/BIC and CTCP is significant. Queued packets in the NIC buffer and the MSPP buffer cause the RTT to build up. When these buffers are full, each new packet will be queued behind a maximum 1700 packets. For an OC3 circuit, the service time for



(a) Throughput

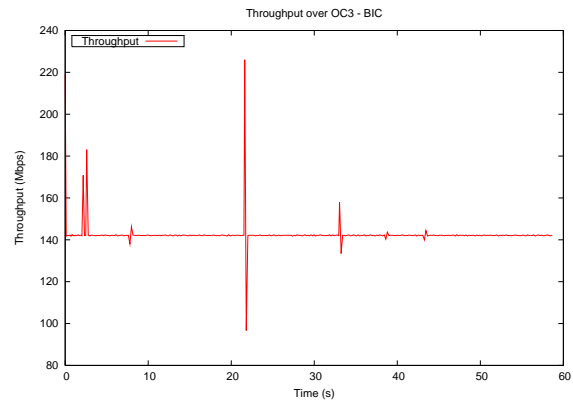


(b) Congestion window (cwnd)

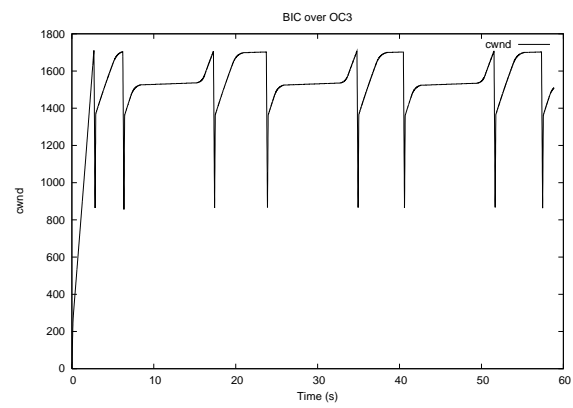
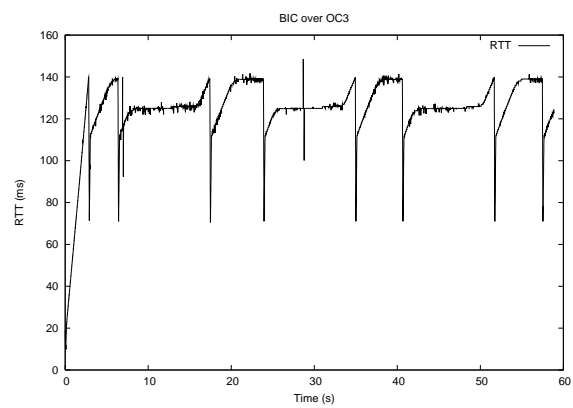


(c) Round-trip time (RTT)

Figure 4.4: Reno over OC3 - 1GB Transfer

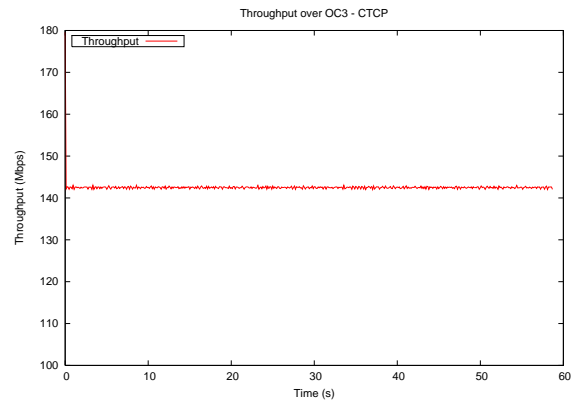


(a) Throughput

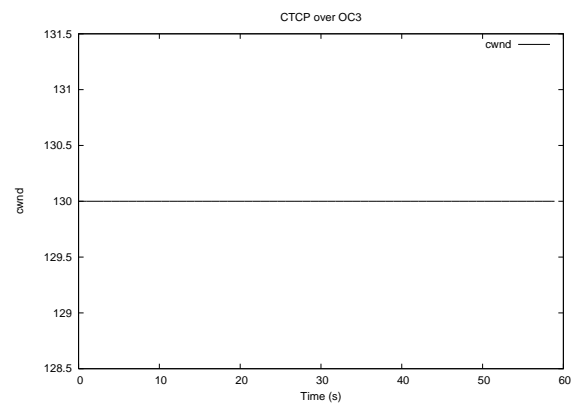
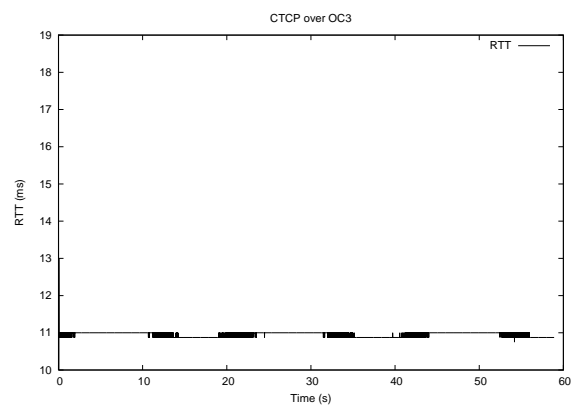
(b) Congestion window (*cwnd*)

(c) Round-trip time (RTT)

Figure 4.5: BIC over OC3 - 1GB Transfer



(a) Throughput

(b) Congestion window (*cwnd*)

(c) Round-trip time (RTT)

Figure 4.6: CTCP over OC3 - 1GB Transfer

a standard 1500B packet is $77\mu\text{s}$. Hence, a packet may experience a maximum queueing delay of $1700 \times 77\mu\text{s}$ or 131ms. Therefore, the maximum RTT observed when using Reno (Fig. 4.4(c)) and BIC (Fig. 4.5(c)) is as large as $(131 + 8.85) \approx 140\text{ms}$. In contrast, for CTCP, RTT holds steady at a value slightly higher than the base 8.85ms, since as seen in Fig. 4.6(b), *cwnd* is capped at 130 packets. This point is important when considering how a large flow could severely impact the latency of packets in a small flow if both are mapped to the same VCG.

4.5.4 Discussion

With the in-depth understanding from this work, several options can be suggested to improve performance. Given there is a control-plane procedure for circuit provisioning, this phase should get information about buffer sizes, select an appropriate TCP congestion control module, and configure TCP-layer receive and transmit buffers and Ethernet-layer *txqueuelen*, in order to achieve the best performance. Different options may be more or less easily implementable depending on one's situation.

One approach to limit the number of outstanding packets in Reno or BIC is to set the receive-side buffer to equal or be slightly larger than the initial BDP (with no buffered packets). In this example, this would limit the number of outstanding packets to a value of 100 packets even if *cwnd* keeps increasing to 1750. The RTT would then be held to a value close to 8.85ms. The receive-buffer size should be initialized on a per-socket basis within the application but this requires application code modification. If set using external OS commands, it becomes a system-wide parameter used for all TCP flows. If the value chosen is too small, it will hurt throughput of large BDP connections. If it is too large, it will impact RTT as shown with the above experiment.

With CTCP, the effect of bursting a whole BDP-sized window at the Gbps NIC rate when a transfer starts needs to be considered since slow start is disabled. If the ingress MSPP, where the GigE port is mapped to a lower-rate SONET circuit, does not have sufficient buffer space to hold this whole window of packets, *Pause* frames will be generated. Care should be taken to configure the *txqueuelen* variable within the sending host to a large value to prevent a CWR state.

The importance of this work is not limited to circuits. Implications for connectionless traffic

can be drawn from these results with respect to how “elephant” flows, if allowed to grow to a large congestion window, will cause “mice” flows to incur unnecessary packet delay and/or losses, and streamed flows to suffer jitter.

4.6 Conclusions

The key **contribution** of this work is a detailed understanding of the interactions between TCP congestion control algorithms, data-link layer flow control algorithms, and switch buffering. These interactions are observed using a new tool called BWdetail. The behavior of three variants of TCP: Reno, BIC, and CTCP, are observed across an Ethernet-SONET mismatched-rate circuit. These types of circuits will increasingly be used across the newly deployed dynamic circuit networks in research-and-education networks, such as Internet2 and ESnet4, and in commercial bandwidth-on-demand services. As there is a circuit provisioning phase, this phase can be used to obtain parameters such as end-to-end propagation delay. Using these values and the circuit rate, flow-control parameters such as *Pause ON* and *Pause OFF* threshold should be set within the circuit switches in the provisioning phase. Furthermore, in addition to tuning TCP send and receive buffer sizes, Ethernet-layer output queue length should be configured at the end hosts on a per-circuit basis. Such configurations are critical to exploiting the full transmission rate allocated to the circuit to achieve better performance (lower file transfer delays).

Chapter 5

On Virtualizing Ethernet Switches

5.1 Introduction

Virtualization is a central concept in the NSF initiative to create a Global Environment for Network Innovations (GENI) [96]. As the GENI testbed is expected to have both host systems and network switches/routers (among other entities), techniques are needed for virtualizing both hosts and switches/routers. The concept of virtualizing a computer host to create virtual machines has been widely implemented. There are several techniques for virtualizing hosts in both commercial and research/education communities such as VMware [97], Virtual Server [98], Xen [99], Planet-Lab [100], etc.

Two types of virtualizable switches/routers, which fall at opposite ends of the flexibility/performance spectrum, are: (i) software-based switches/routers created using off-the-shelf servers, as implemented in the Virtual Network Infrastructure (VINI) project [101], and (ii) a hardware-based high-speed router/switch with Advanced Telecommunications Computing Architecture (ATCA) components and network processors as proposed in [102].

We propose a switch/router virtualization solution that lies between these two extremes. It is based on using off-the-shelf switches/routers and poses only one requirement on these switches, i.e., that the switch/router has built-in capabilities to support isolated *slivers* (bandwidth partitions) on its data-plane interfaces. Ideally to create multiple logical switches from one physical switch, in addition to support for separable slivers on the data-plane, the switch should have built-in capabil-

ities to support multiple, separable (a) control-plane instances, e.g., routing protocol and signaling protocol processes, (b) management-plane instances, e.g., Simple Network Management Protocol (SNMP) agents with partitioned management information bases (MIBs), and (c) Command-Line Interface (CLI) processing instances. In our proposed solution for switch/router virtualization, software external to the switch is used to partition the control plane, management plane and CLI administrative-access, thus requiring built-in support for only data-plane partitioning.

In Section 5.2, we explain our motivation for carrying out this work, and contrast the goals of the VINI, ATCA and our approach to switch/router virtualization. In Section 5.3, we describe our overall virtualization approach, and then focus on one aspect, the sharing of the administrative CLI. Section 5.4 describes our implementation of a software module called the Slice Administration Controller (SAC), and reports on obtained measurements. We conclude the chapter in Section 5.5.

5.2 Motivation and Related Work

5.2.1 Initial drivers

Our initial interest in virtualization came about when we wanted to run some experiments on Internet2's Hybrid Optical Packet Infrastructure (HOPI) [103] testbed while another research group was simultaneously using the testbed. HOPI is an experimental testbed available for use by networking and other scientific researchers. When considered in the context of HOPI, GENI, and similar situations in which multiple researchers are using the same physical infrastructure, there is a *strong motivation* to virtualize a node in order to allow researchers to operate without interfering with each other.

The HOPI network consists of Force10 E600 nodes deployed at five cities, Washington, New York, Chicago, Seattle and Los Angeles. The Force10 E600 does meet our requirement of built-in support for sliver isolation on its data-plane interfaces. Therefore, we used this model in an implementation of our virtualization approach.

The E600 equipment can be configured to operate as an Ethernet switch ("layer-2 (L2)" switch), an IP router ("layer-3 (L3)" router), or a combination thereof. In our studies on the feasibility

of virtualization of this model, we configured it strictly as an Ethernet switch. This was partly because this was the intended mode of operation on the HOPI network, and also because the set of functions/features that need to be virtualized is much smaller in this mode, making it the better first-step choice.

5.2.2 Broader motivation and related work

In this section, we compare goals and usability of four different approaches to switch/router virtualization. In addition to the VINI, ATCA router, and our approach, we describe commercial interests in virtualization.

In *VINI*, the goal is to provide researchers access to virtualized high-end servers for a simultaneous testing of new protocols and services. Reference [101] mentions the possibility of eventually using “programmable hardware devices” instead of servers. A VINI implementation on PlanetLab runs XORP [104] routing software and *Click* packet forwarding software [105]. This testbed is suitable for researchers whose work can be evaluated in an overlay mode.

For high-speed data-plane experiments, the *ATCA programmable router* [102] is ideal. These routers allow data-plane researchers to download code for functions such as packet scheduling, forwarding table lookups, etc., to network processors/FPGAs on the line cards for testing and evaluation.

In *our virtualization approach*, which uses off-the-shelf switches and routers, this level of programmability, i.e., support for code downloads, is not available. Nevertheless these switches offer researchers the ability to *configure node operation* via the CLI. Given the vast number of features implemented in commercial switches, such configurability access will open up several lines of experimental research.

We list several examples of the type of networking research (in data, control, and management planes) possible with just such *configurability access* of switches: (1) data-plane research combining Weighted Random Early Discard (WRED) and transport protocols (WRED profiles can be set through the CLI), (2) data-plane research on QoS-support mechanisms, which, while limited by the capabilities of the switches, still offer much scope for experimental work to understand

how these capabilities should be configured for desired operations, (3) data-plane research on security and access-control lists by configuring various parameters related to these functions, (4) control-plane research in connection-oriented networks for advancing signaling protocols, routing protocols, advance-reservation schedulers, etc., as is being carried out on the CHEETAH [24] and DRAGON [4] testbeds, (5) control-plane research on routing protocols for IP networks, such as XORP [104], (6) management-plane research on algorithms and open-source tools [106], (7) measurement tools to estimate bandwidth, capacity, loss rates, and delay on end-to-end paths, such as those listed in [107]; visibility to the switches and routers could open up a new set for innovations, as current tools operate without such access.

Given this rich set of experimental lines of work enabled by offering researchers configurability access to switches through the CLI, we found a *broader motivation* than our initial HOPI-oriented driver for virtualizing off-the-shelf switches. Thus, even without the ability to download code to these switches, i.e., the “programmability” attribute defined in [96], there is value in virtualizing off-the-shelf switches.

There is growing interest in the *commercial sector* among equipment vendors to support built-in capabilities for switch/router virtualization of all four levels (the data, control, and management planes, and administrative access). Interestingly, new products are being created to virtualize both data-center switches [108] as well as high-end service-provider routers [109], [110]. For example, various departments of an organization can share the computers and switches in an enterprise data center. As these switches become more widely available, based on their supported capabilities, our software will need to be adapted to offer researchers the required flexibility. Just as PlanetLab OS virtualizes hosts in an environment suitable for researchers in spite of the availability of commercial tools, e.g., VMware, we expect that researchers will need open-source software support for virtualization of commercial switches/routers, a gap filled by this work.

5.3 Approach for Virtualization of an off-the-shelf Ethernet Switch

5.3.1 Virtualization architecture overview

In this section, we describe our approach for virtualizing an off-the-shelf Ethernet switch that has built-in support for sliver isolation on its data-plane interfaces. Fig. 5.1 shows our virtualization architecture. The Force10 E600 switch is used as an example of the type of commercial switch that can be virtualized with this approach.

First, we define a few terms. The term *sliver* is defined as a bandwidth partition on a port with a set of associated multiplexing/demultiplexing identifiers, e.g. VLAN identifiers. The term *slice* is used to represent a logical switch, which consists of a set of slivers on a specific set of ports of the physical switch. Our definition of the word “slice” is for a single switch, though this term has been used in the GENI community to represent a set of resources across a whole network (i.e., multiple switches). We decided to focus this paper on the virtualization problem for a single switch before tackling the broader problem of providing a researcher a virtual set of resources across a network of switches. Therefore, Fig. 5.1 shows just a single switch and associated software. Some of the software modules, such as the slice scheduler, could be generalized for a network of switches. The next term we define is *entity*. This represents both a human administrator of a logical switch (slice) as well as software modules implemented by the slice owner to interface with the slice scheduler and Slice Administration Controller (SAC) of Fig. 5.1, as these software modules offer both graphic and programmatic interfaces.

The main components of our virtualization software architecture are shown in Fig. 5.1. As stated in Section 5.1, multiple *control-plane instances* and *management-plane instances* are run outside the switches to provide corresponding functionality to the logical switches. Researchers access the slice scheduler (via a graphic or programmatic interface) to request reservations for switch slices for a fixed duration of time. The slice scheduler maintains reservations for discrete time periods (e.g., hourly basis). The slice scheduler translates a slice request into a set of required slivers, and checks whether the request can be accepted, based on resource availability in the requested time period. Sliver allocations for accepted reservations are written into the configuration database shown in

Fig. 5.1.

The last component of the virtualization software architecture shown in Fig. 5.1 is the Slice Administration Controller (SAC). The purpose of the SAC is to enable sharing of the administrative command-line interface. It is an important module to allow multiple researchers to *configure* operation of their logical switches, enabling the types of research listed in Section 5.2.2.

In this first phase of the work, we focused on testing the switch data-plane mechanisms for sliver isolation, and implemented and evaluated the SAC. Therefore, in the following two subsections, we provide details on these two aspects of the overall architecture.

5.3.2 Data-plane support for sliver isolation

The Force10 E600 switch implements the IEEE 802.1Q standard, which is a multiplexing scheme based on 12-bit Virtual LAN identifiers (IDs) and the 3-bit priority field. We use the term “class” when referring to the latter because in the virtualization application all slivers are treated equally.

Force10’s E600 switch implements a rich set of QoS features for traffic control on the data plane. These features include rate policing for ingress traffic, per-class queueing, rate limiting for egress traffic, and rate shaping. When a packet enters a port on a line card, it is first processed by a Flexible Packet Classification engine (FPC). The FPC examines information in the Ethernet, IEEE 802.1Q, and IP headers, and performs a lookup on a stored policy table to determine how this packet should be marked according to configured policies. After classification, the packet is processed using a two-rate three-color rate-policing mechanism [111], and placed into one of the eight queues according to the policies installed by a switch administrator through the CLI. The service order of these queues is controlled by a scheduler according to a Weighted Fair Queueing (WFQ) algorithm. Packets are then served out of the queues and passed through a set of switch fabric cards to egress line cards. In the egress direction, packets are again placed into eight separate queues and processed by a two-rate three-color marking mechanism. The queues are serviced using a WFQ algorithm.

On each physical port, the administrator can use a *rate-policing* command to control the operation of the two-rate three-color marking module in the ingress direction. A *rate-limiting* command

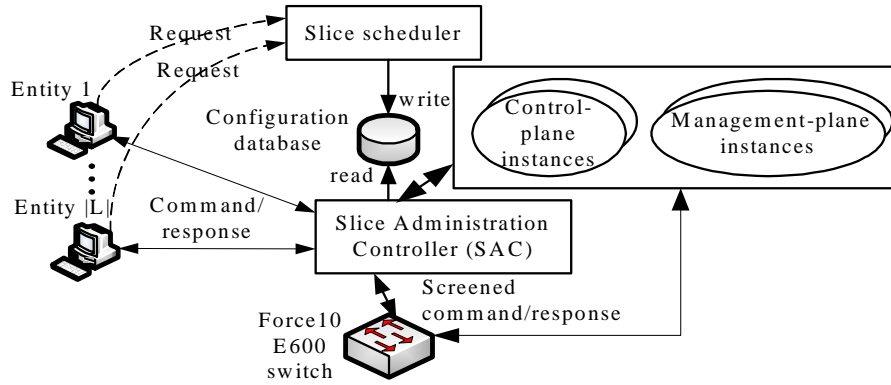


Figure 5.1: Virtualization Architecture

can be applied to the same port for the operation of the two-rate three-color marking module in the egress direction. If packets are classified by VLAN ID, we can specify at most six rate-policing commands and/or six rate-limiting commands on each physical port in the E600 switch. Data-plane isolation is ensured by a combination of actions performed in these data-plane circuits, along with actions performed in the slice scheduler and SAC. The slice scheduler tracks aggregate port bandwidth when granting slice reservations and the SAC monitors usage of assigned bandwidth when checking configuration commands from slice entities. Through experimentation, we have verified that the bandwidth partitions assigned to the various entities sharing the switch are enforced.

5.3.3 Slice Administration Controller (SAC)

The purpose of the SAC is to enforce that the instantiation of slices by entities is within the bounds the slice scheduler has assigned. In other words, the SAC makes sure an entity does not attempt to manipulate resources that have not been assigned to it by the slice scheduler. As shown in Fig. 5.1, the SAC reads the configuration database, updated by the slice scheduler, to discover the resources to which each entity has access in each time interval. The SAC then checks commands issued from each entity. It screens each command to verify that it only requires operations on resources allocated to the slice granted to the corresponding command-issuing entity, and then passes on approved commands to the Ethernet switch. Responses are relayed from the switch back to the entity.

A slice allocation for a given time interval is represented as follows: $(\mathbf{R}, \mathbf{B}, \mathbf{P}, \mathbf{V})$, where $(\mathbf{R} = r_l^e, 1 \leq l \leq L, e \in \mathbf{E})$ is the set of requested rates on all the links of the switch, \mathbf{E} is the set of entities, and L is the number of links (interfaces) on the switch. Similarly, the set of requested buffer sizes on all the links is represented by $(\mathbf{B} = b_l^e, 1 \leq l \leq L, e \in \mathbf{E})$. Sets \mathbf{P} and \mathbf{V} represent the set of class identifiers, and the set of VLAN IDs, respectively.

Slices allocated in a given time interval, T , will be assigned resources such that:

$$\left\{ \begin{array}{l} \mathbf{P}_i \cap \mathbf{P}_j = \emptyset, 1 \leq i, j \leq |\mathbf{E}_T|, \\ \mathbf{V}_i \cap \mathbf{V}_j = \emptyset, 1 \leq i, j \leq |\mathbf{E}_T|, \\ \sum_{e \in \mathbf{E}_T} r_l^e = C_l, 1 \leq l \leq L, \\ \sum_{e \in \mathbf{E}_T} b_l^e = B_l, 1 \leq l \leq L \end{array} \right. \quad (5.1)$$

where $\mathbf{E}_T \subseteq \mathbf{E}$ presents the set of entities that requested slices in a given time interval, T , $i \neq j$, and C_l and B_l are the capacity of and buffer size associated with link l , respectively.

We show an example of slice allocations for one time interval in Table 5.1. In this example, two entities are sharing this switch and the switch has four links. In Table 5.1, since QoS functions can be applied to either classes or VLANs, but not both, the slice scheduler pre-determined that *Link*₁ and *Link*₃ will operate in class mode, and *Link*₂ and *Link*₄ will operate in VLAN mode. This determination can be changed for every time interval based on the received requests. If, for example, $\mathbf{P}_1 = \{0, 1, 2, 3\}$, and $\mathbf{P}_2 = \emptyset$, the empty set, it means that entity 1 can issue rate limiting and rate policing commands for interfaces *Link*₁ and *Link*₃ for classes $\{0, 1, 2, 3\}$. By not specifying classes for entity 2 (since $\mathbf{P}_2 = \emptyset$), we know that its allocations $r_1^2 = 0$ and $r_3^2 = 0$. If an entity wants the full rate of a link, e.g., $r_1^1 = c_1$, where c_1 is the capacity of link 1, then it must indicate a priority class value for this link. The Force10 supports a command for setting a priority value for all frames arriving on a given interface. Thus, in our example, entity 1 can issue this command to have the switch mark all frames entering *Link*₁ with one of its four assigned priority classes before transmission. VLAN IDs are handled similarly. For example, $\mathbf{V}_1 = \{101 - 200\}$ and $\mathbf{V}_2 = \{201 - 300\}$. In this case, both entities would have non-zero rate/buffer tuples for *Link*₂ and *Link*₄.

Let us consider an example of how the SAC handles a command from an entity. Rate polic-

Table 5.1: Configuration database for one time interval

Entity	Class-mode		VLAN-mode		P	V
	<i>Link</i> ₁	<i>Link</i> ₃	<i>Link</i> ₂	<i>Link</i> ₄		
1	r_1^1, b_1^1	r_3^1, b_3^1	r_2^1, b_2^1	r_4^1, b_4^1	P ₁	V ₁
2	r_1^2, b_1^2	r_3^2, b_3^2	r_2^2, b_2^2	r_4^2, b_4^2	P ₂	V ₂

ing/limiting QoS commands specify a Committed Information Rate (CIR), and optionally Committed Burst Size (CBS), corresponding to the committed rate and committed burst size for a VLAN or a class. These commands are issued in “interface” mode, which means they apply to a specific interface. Let us represent such a command from entity e as $(e:l, v, cir, cbr)$, where l identifies the interface, v , the VLAN ID, cir , the CIR value, and cbs , the CBS value. The SAC maintains a table of aggregate rates already assigned by each entity to VLANs and/or classes for all the links on the switch. We represent the aggregate rate assigned to already-configured VLANs by entity e on a VLAN-mode link l as A_l^e . We use the symbol D_l^e to represent the already-assigned total buffer space on link l by entity e . Upon receiving the command, the SAC first checks that link l is set to be shared in VLAN mode at this time interval. If it passes this test, it then subjects the command parameters to the following tests:

$$v \in \mathbf{V}_e, cir \leq r_l^e - A_l^e, cbs \leq b_l^e - D_l^e \quad (5.2)$$

If these conditions are met, the SAC allows the command to be sent through to the switch. If not, the command is rejected.

5.4 Implementation and Measurements

We implemented the SAC software component of our virtualization architecture. In this section, we first describe design points, then the implementation of the SAC software component, and finally present our measurements of how much overhead the SAC introduces.

This case study proceeds from our initial motivation of virtualizing the common infrastructure

of Force10 Ethernet switches in the HOPI testbed to enable concurrent control-plane instances to coexist. Specifically, this scenario involves using VLANs and rate policing/limiting for data-plane virtualization, no changes to the control plane, and an external program to virtualize a small set of administrative functions.

5.4.1 SAC Design

We now discuss some design criteria for the SAC. There were five distinct design goals relating to the SAC: resource control, transparency, security, performance, and reliability.

5.4.1.1 Resource control

The primary function of the SAC is as described in Section 5.3.3, to control access to switch resources. The SAC implementation must, of course, support this function.

5.4.1.2 Access Transparency

An entity is not generally aware that it is issuing commands to the SAC instead of to the switch directly. The SAC provides access transparency as opposed to content transparency. Full content transparency is not provided. By limiting the transparency, we can give an entity the illusion of a fully functional switch over which it has full control, rather than a shared switch over which it can only control some resources.

5.4.1.3 Security

The SAC software module serves as a intermediary between entities and the switch as shown in Fig. 5.1, therefore communication into and out of the SAC must be secure. Access control to the SAC is limited only to authorized entities. The SAC holds the sole (with the exception of the switch owner) login to the switch. The resource control functions of the SAC provide a measure of security against unauthorized manipulation of resources at the switch. A compromised entity will be unable to effect other entities' experiments or any resources segmented off for production use.

5.4.1.4 Performance

With the motivation of accommodating multiple control-plane software solutions that may be issuing commands to the SAC at a relatively high rate (compared to a human user), we have a goal of not paying too high a performance cost in terms of added delay in command processing. The SAC performs as little processing on incoming commands as possible and delivers return information from the switch without alteration or any processing at all, in order to keep the SAC from creating too much overhead.

5.4.1.5 Reliability

The SAC and the switch could potentially crash or be administratively reset. This introduces a risk of the SAC becoming out of synchronization with the state of the switch. In order to provide reliable enforcing of resource sets, upon initialization the SAC discovers the current state of resources by querying the switch.

5.4.2 Implementation

5.4.2.1 Setup

The SAC is an application which is executed in lieu of a entity's shell in a Linux environment. A typical entry in `/etc/passwd` might end with `:/bin/bash`, so that upon login an instance of the bash shell is instantiated to receive commands from the entity. Instead, to use the SAC, the entry for an account ends with `:/home/account/vr`. Each entity that requires access to a switch is assigned its own separate account on a Linux machine. This scheme accomplishes several goals:

1. account management is performed on the Linux machine using the familiar account creation/deletion mechanisms
2. since each entity has its own account, each entity's resource set can be enumerated in a configuration file which resides in that account's home directory such as `:/home/account/vr_config`
3. secure authentication is handled by the Linux system

4. with the shell effectively replaced by the SAC, no access to the Linux system is granted to the entity

Each entity's `vr_config` file is defined by the administrator and stored in that entity's home directory. Upon instantiation, the SAC initializes the appropriate resource set for that entity by reading in the values from `vr_config`. These values would be obtained from the database shown in Fig. 5.1 if the complete virtualization architecture was implemented.

5.4.2.2 Operation

Once an entity has logged in to the Linux machine, and the SAC has initialized its values, according to Section 5.4.2.1, the SAC securely connects to the switch using SSH2. The prompt (e.g., *LOSA-Force10#*) returned upon login to the switch is immediately passed to the entity by the SAC, so the entity is transparently presented with the command prompt from the switch upon logging into the Linux machine. The entity then issues commands which are terminated at the SAC. All commands issued from the entity, which deal with resources, are parsed and checked against the resource set in order to verify that the resource is able to be manipulated by the entity. If the command is acceptable (i.e., the resource specified in the command is contained in the entity's resource set), then the command is passed to the switch unchanged. If, however, the resource specified in the command is not in the entity's resource set, then the SAC prints an error message (e.g., *% Error: Permission Denied for VLAN xx*) and sends a single new-line character to the switch to cause a command prompt to be returned, which is then passed to the user.

5.4.2.3 Security

Security between the SAC and both the entities and switch is provided by SSH2 in our implementation. Specifically, the `libssh2` [112] library is integrated into the SAC. In general, security between the entity and the Linux machine is dependent on which methods the administrator allows, though SSH2 is desired. Upon instantiation, the SAC attempts to connect using the SSH2 protocol to the switch with username/password authentication.

Table 5.2: Delay for various commands

	(ms)	(ms)	(%)
COMMAND	Direct	SAC	Overhead
show vlan id	111.7	140.8	26%
configure	71.7	73.8	3%
interface gi allowed	75.9	84.5	11%
interface gi denied	N/A	55.5	N/A
exit from if-gi to conf	57.3	64.6	13%
interface VLAN allowed	85.1	105.8	24%
interface VLAN denied	N/A	55.6	N/A
exit from VLAN to conf	57.1	66.8	17%
show config in conf-if	105.7	127.0	20%
rate limit allowed	135.0	142.5	6%
rate limit rate denied	N/A	51.5	N/A
rate limit VLAN denied	N/A	54.9	N/A
no rate limit	148.5	158.8	7%

5.4.3 Measurements

We gathered measurements, presented in Table 5.2, to discover the level of overhead from issuing commands through the SAC. Since the research entities we were supporting with the SAC could potentially be issuing commands at relatively high rates, the SAC should not introduce unacceptably high delay for processing commands. The column marked Direct contains the delay for processing commands issued directly to the switch, not through the SAC. The column SAC shows delay for commands which were issued to the SAC. Entries in the table marked “N/A” are not applicable since they correspond to functions where the SAC checks commands against a resource set, and this does not apply to a direct login session. Overall, the delay is acceptable relative to the research projects for which we designed the SAC.

5.5 Conclusions

We proposed a technique for virtualizing an off-the-shelf Ethernet switch, with built-in capabilities for isolating bandwidth partitions on its data-plane interfaces, using external software. Such virtual-

ization would enable the sharing of a physical switch by multiple, concurrent research experiments, with each experiment being assigned its own logical switch. Even without the ability to download code into the switches (i.e., programmability), many experiments can be conducted by offering researchers the ability to *configure* logical switch operation through a shared access to the switch's administrative command-line interface (CLI). We demonstrated our approach by implementing a *Slice Administration Controller* that enables shared, but policed, access of the CLI, for the Force10 E600 system, configured as a Layer-2 (Ethernet) switch.

Chapter 6

Conclusions and Future Work

In this chapter we summarize the dissertation, and discuss future work that could be done to extend this research.

6.1 Conclusions

We have studied connection-oriented service with fairness-considering advance-reservation scheduling on circuit-switched and packet-switched infrastructures. The work is divided into a theoretical component and an experimental component. Chapters 3 and 2 deal with the **theoretical component**, in which we studied advance-reservation scheduling, and compared reservation systems to queueing systems.

A fairness-considering advance-reservation system was presented in Chapter 3. The scheduling algorithm applies to any bin packing problem or advance-reservation scenario where a flexible class boundary is allowed. We found that with more channels in a bin and an increased number of bins, the larger system affords the optimization a greater opportunity to enforce fairness, but the cost is a higher mean response time. The system has difficulty ensuring fairness when the per-channel loads are lower, as there are fewer possible choices for the algorithm to make. It enforces fairness quite well at higher loads, and the fairness can be relaxed by a tunable variable to achieve lower mean response time by decreasing the fairness. Finally, the algorithm is compared to a non-optimizing first-come first-served algorithm to illustrate the performance tradeoff incurred in

achieving fairness.

In Chapter 2, a *general reservation system model (GRSM)* was shown as a way to uniformly express the key set of characteristics common to all reservation systems. Several examples of reservation systems and queueing systems were used to illustrate the variations on those characteristics, such as whether a customer wants the first available timeslot or wants to specify a timeslot. This work answers the important question of *when to use a reservation system instead of a queueing system*, as determined by a consideration of revenues (considering load), costs (including servers), and competition. Finally, we provided five distinct solutions for simplified models drawn from the GRSM under more certain sets of assumptions into which the example systems could be divided, showing the influence of such factors as reservation window size and server vacations.

Chapters 4 and 5 comprise the **experimental component**, in which we studied the data-plane considerations of providing connection-oriented service, and the issue of creating connection-oriented service on a packet-switched infrastructure.

Chapter 4 detailed the interactions between TCP congestion control algorithms, data-link layer flow control algorithms, and switch buffering. These interactions are observed using a new tool called *BWdetail*. The behavior of three variants of TCP: Reno, BIC, and CTCP, are observed across an Ethernet-SONET mismatched-rate circuit. These types of circuits will increasingly be used across the newly deployed dynamic circuit networks in research-and-education networks, such as Internet2 and ESnet4, and in commercial bandwidth-on-demand services. As there is a circuit provisioning phase, this phase can be used to obtain parameters such as end-to-end propagation delay. Using these values and the circuit rate, flow-control parameters such as the *Pause ON* and *Pause OFF* thresholds should be set within the circuit switches in the provisioning phase. Furthermore, in addition to tuning TCP send and receive buffer sizes, Ethernet-layer output queue length should be configured at the end hosts on a per-circuit basis. Such configurations are critical to exploiting the full transmission rate allocated to the circuit to achieve better performance (lower file transfer delays).

Finally, in Chapter 5 we proposed a technique for virtualizing an off-the-shelf Ethernet switch, with built-in capabilities for isolating bandwidth partitions on its data-plane interfaces, using ex-

ternal software. This includes a slice scheduler that acts as a reservation scheduler. In this way, a single physical switch can be divided into multiple logical switches. This allows a logical switch to be configured by control-plane software without exposing the entire packet-switched infrastructure.

6.2 Future Work

We discuss the following as possible extensions to this work:

1. The fairness-considering multi-class reservation system is designed for a single link. The system could be extended to consider multiple links. Domain administrators do not typically share full topology, so a multiple link treatment could include the concept of distributed scheduling.
2. The various reservation systems we investigated have a considerable component of human behavior that is difficult to quantify. Extensive field study and polling to characterize consumer behavior would allow more fine-tuned analyses of the different systems considered. Further, many assumptions were made for model tractability that could be removed and replaced with complex representations in terms of capturing the nuances of human decision making.
3. Just as we performed an in-depth study for circuits, a similar study would be necessary to understand virtual circuits in terms of the policing and shaping mechanisms used to provide connection-oriented service on packet-switched infrastructures. Likewise, extending on CTCP as a transport protocol for circuits could result in one specifically designed for virtual circuits.

Bibliography

- [1] C. Guok, D. Robertson, E. Chaniotakis, M. Thompson, W. Johnston, and B. Tierney, "A User Driven Dynamic Circuit Network Implementation," in *Proceedings of the Distributed Autonomous Network Management Systems Workshop*, New Orleans, LA, Nov 2008.
- [2] V. Jacobson, "Congestion Avoidance and Control," in *Proceedings of SIGCOMM*, Stanford, CA, Aug. 1988.
- [3] On-demand secure circuits and advance reservation system. [Online]. Available: <http://www.es.net/oscars/index.html>
- [4] Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON). [Online]. Available: <http://dragon.maxgigapop.net/>
- [5] Internet2 ION. [Online]. Available: <http://www.internet2.edu/ion/>
- [6] X. Zhu, "A study of bandwidth-sharing mechanisms in connection-oriented networks," Ph.D. dissertation, University of Virginia, Virginia, Feb. 2008.
- [7] S. Kapodistria, "The m/m/1 queue with synchronized abandonments," *Queueing Systems*, vol. 68, pp. 79–109, 2011.
- [8] S. Floyd, "HighSpeed TCP for large congestion windows," Feb. 2003. [Online]. Available: <http://www.icir.org/floyd/hstcp.html>
- [9] 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.

- [10] OpenFlow. [Online]. Available: <https://www.opennetworking.org/index.php>
- [11] M. McGinley, X. Zhu, and M. Veeraraghavan, "On Reservations Systems and Queueing Systems," *Submitted to Manufacturing and Service Operations Management*, 2011.
- [12] M. McGinley and M. Veeraraghavan, "Fairness in Multi-Class Book-Ahead Scheduling," *Submitted to IEEE Transactions Networking*, 2011.
- [13] M. McGinley, H. Bhuiyan, T. Li, and M. Veeraraghavan, "An in-depth cross-layer experimental study of transport protocols over circuits," in *Proc. of IEEE ICCCN 2010*, Zurich, Switzerland, Aug. 2010.
- [14] M. E. McGinley, T. Li, and M. Veeraraghavan, "On Virtualizing Ethernet Switches," in *Proc. of IEEE ICCCN 2008*, St. Thomas, USVI, Aug 2008.
- [15] M. Hovestadt, O. Kao, A. Keller¹, and A. Streit¹, "Scheduling in HPC resource management systems: Queuing vs. planning," in *Book Series: Lecture Notes in Computer Science, Job Scheduling Strategies for Parallel Processing*, vol. 2862. Springer Berlin, Heidelberg, 2003, pp. 1–20.
- [16] L. Abeni and G. Buttazzo, "Resource reservation in dynamic real-time systems," *Journal of Real-Time Systems, Springer, Netherlands*, vol. 27, no. 2.
- [17] D. G. Kendall, "Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain," *The Annals of Mathematical Statistics*, vol. 24, no. 3, pp. 338–354, 1953.
- [18] W. E. Johnston. Networking for the Future of DOE Science. [Online]. Available: <http://www.es.net/ESnet4/ESnet4-Networking-for-the-Future-of-Science-2008-02-09-FORTH-Crete.ppt>
- [19] L. Kleinrock, *Queueing Systems Volume 1: Theory*, 1st ed. Wiley, John and Sons, Incorporated, 1975.

- [20] M. Schwartz, *Telecommunication networks: protocols, modeling and analysis*. Boston, MA: Addison-Wesley, 1986.
- [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'07*, Washington, DC, Nov. 2007.
- [22] X. Zhu and M. Veeraraghavan, "Analysis and Design of Book-ahead Bandwidth-Sharing Mechanisms," *IEEE Transactions on Communications*, vol. 56, no. 12, pp. 2156–2165, Dec. 2008.
- [23] ESnet. [Online]. Available: <http://www.es.net/>
- [24] CHEETAH. [Online]. Available: <http://www.ece.virginia.edu/cheetah/>
- [25] X. Fang and M. Veeraraghavan, "On using circuit-switched networks for file transfers," in *Proc. of IEEE Globecom*, New Orleans, LA, Nov. 2008.
- [26] X. Fang, M. Veeraraghavan, M. E. McGinley, and R. W. Gisiger, "An overlay approach for enabling access to dynamically shared backbone GMPLS networks," in *IEEE ICCCN*, Honolulu, Hawaii, Aug 2007.
- [27] A. Banerjee, N. Singhal, J. Zhang, D. Ghosal, C.-N. Chuah, and B. Mukherjee, "A Time-Path Scheduling Problem (TPSP) for Aggregating Large Data Files from Distributed Databases using an Optical Burst-Switched Network," in *Proc. of IEEE ICC 2004*, Paris, France, Jun. 2004.
- [28] N. Boudriga, M. Obaidat, M. Cherif, and S. Guemara-ElFatmi, "An Advance Dynamic Resource Reservation Algorithm for OBS Networks: Design and Performance," in *Proc. of IEEE ICECS 2005*, 2005, pp. 1–4.
- [29] L.-O. Burchard, "On the Performance of Computer Networks with Advance Reservation Mechanisms," in *Proc. of the 11th International Conference on Networks*, Sydney, Australia, 2003.

- [30] C. Castillo, G. N. Rouskas, and K. Harfoush, "On The Design of Online Scheduling Algorithms for Advance Reservations and QoS in Grids," in *Proc. of IEEE IPDPS 2007*, Long Beach, California, Mar. 2007.
- [31] E. G. Coffman, P. Jelenkovic, and B. Poonen, "Reservation Probabilities," *Advances in Performance Analysis*, vol. 2, no. 2, pp. 129–158, 1999.
- [32] T. Erlebach, "Call Admission Control for Advance Reservation Requests with Alternatives," in *Proceedings of the 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks*, Rome, Italy, Sep 2002, pp. 51–64.
- [33] 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.
- [34] C. Curti, T. Ferrari, L. Gommans, B. van Oudenaarde, E. Ronchieri, F. Giacomini, and C. Vistoli, "On Advance Reservation of Heterogeneous Network Paths," *Future Generation Computer Systems Journal*, vol. 21, no. 4, pp. 525–538, 2005.
- [35] S. Figueira, N. Kaushik, S. Naiksatam, S. A. Chiappari, and N. Bhatnagar, "Advance Reservation of Lightpaths in Optical Network Based Grids," in *ICST/IEEE GridNets 2004*, San Jose, CA, 2004.
- [36] S. Figuerola, J. A. Garca, ngel Snchez, C. de Waal, and A. Willner, "The Network Service Plane: An Approach for Inter-Domain Network Reservations," in *Proc. of IEEE ICOTN 2008*, Athens, Greece, 2008.
- [37] A. G. Greenberg, R. Srikant, and W. Whitt, "Resource Sharing for Book-Ahead and Instantaneous-Request Calls," *IEEE/ACM Trans. Netw.*, vol. 7, no. 1, pp. 10–22, 1999.
- [38] R. Guérin and A. Orda, "Networks with Advance Reservations: The Routing Perspective," in *Proc. of IEEE INFOCOM*, Tel Aviv, Israel, 2000.

- [39] M. Hayashi, T. Miyamoto, and H. Tanaka, "Advance reservation-based network resource manager with adaptive path discovery scheme for SOA-based networking," in *Proc. of IEEE OFC/NFOEC 2007*, Anaheim, CA, 2008.
- [40] R. Hayashi, K. Shimizu, I. Inoue, and K. Shiimoto, "Novel Traffic Engineering for Reservation Services Network," in *7th Asia-Pacific Symposium on Information and Telecommunication Technologies, 2008*, Bandos Island, Maldives, 2008.
- [41] E. He, X. Wang, and J. Leigh, "A Flexible Advance Reservation Model for Multi-Domain WDM Optical Networks," in *Proc. of BROADNETS 2006*, San Jose, CA, 2006.
- [42] E. He, X. Wang, V. Vishwanath, and J. Leigh, "AR-PIN/PDC: Flexible Advance Reservation of Intradomain and Interdomain Lightpaths," in *Proc. of IEEE Globecom 2006*, San Francisco, CA, 2006.
- [43] D. Hetzer, I. Miloucheva, and K. Jonas, "Resource Reservation in Advance for Content On-demand Services," in *Proc. of IEEE Networks 2006*, New Delhi, India, 2006.
- [44] A. Jaekel, "Lightpath Scheduling and Allocation Under a Flexible Scheduled Traffic Model," in *Proc. of IEEE Globecom 2006*, San Francisco, CA, 2006.
- [45] A. Kaheel and H. Alnuweiri, "Batch Scheduling Algorithms: A Class of Wavelength Schedulers in Optical Burst Switching Networks," in *Proc. of IEEE ICC 2005*, Seoul, Korea, 2005.
- [46] J. S. Kaufman, "Blocking in a Shared Resource Environment," *IEEE Transactions on Networking*, vol. 29, no. 10, pp. 1474–1481, 1981.
- [47] N. R. Kaushik and S. M. Figueira, "A Dynamically Adaptive Hybrid Algorithm for Scheduling Lightpaths in Lambda-Grids," in *Proc. of IEEE/ACM CCGRID/GAN 2005*, Cardiff, UK, 2005.
- [48] N. R. Kaushik, S. M. Figueira, and S. Chiappari, "Flexible Time-Windows for Advance Reservation Scheduling," in *Proc. of IEEE MASCOTS 2006*, Monterey, California, 2006.

- [49] K. Kim and K. Nahrstedt, "A Resource Broker Model with Integrated Reservation Scheme," in *Proc. of IEEE ICME 2000*, New York, NY, 2000.
- [50] D. Kuo and M. Mckeown, "Advance Reservation and Co-Allocation Protocol for Grid Computing," in *Proc. of the First International Conference on e-Science and Grid Computing, 2005*, Melbourne, Australia, 2005.
- [51] B. Li, J. Chen, and D. Zhao, "Looking-ahead Algorithms for Single Machine Schedulers to Support Advance Reservation of Grid Jobs," in *Proc. of IEEE HPCC 2008*, Dalian, China, 2008.
- [52] H. R. Moaddeli, G. Dastghaibiyfard, and M. R. Moosavi, "Flexible Advance Reservation Impact on Backfilling Scheduling Strategies," in *Proc. of IEEE GCC 2008*, Shenzhen, China, 2008.
- [53] M. A. S. Netto and R. Buyya, "Rescheduling Co-Allocation Requests based on Flexible Advance Reservations and Processor Remapping," in *Proc. of IEEE/ACM Grid 2008*, Tsukuba, Japan, 2008.
- [54] S. Norden and J. Turner, "DRES: Network Resource Management using Deferred Reservations," in *Proc. of IEEE Globecom 2001*, San Antonio, Texas, 2001.
- [55] M. Degermark, T. Khler, S. Pink, and O. Schelen, "Advance Reservations for Predictive Service in the Internet," *ACM/Springer Journal of Multimedia Systems*, vol. 5, no. 3, pp. 177–186, 1997.
- [56] O. Schelen and S. Pink, "An Agent-based Architecture for Advance Reservations," in *Proc. of 22nd IEEE Conference on Local Computer Networks*, Minneapolis, Minnesota, 1997.
- [57] ———, "Resource sharing in advance reservation agents," *Journal of High Speed Networks, Special issue on Multimedia Networking*, vol. 7, no. 3, 1998.
- [58] K. Rajah, S. Ranka, and Y. Xia, "Advance Reservation and Scheduling for Bulk Transfers in Research Networks," *IEEE Transactions on Parallel and Distributed Systems*, to appear.

- [59] W. Reinhardt, "Advance Reservation of Network Resources for Multimedia Applications," in *Proceedings of the International Workshop on Advanced Teleservices and High-Speed Communication Architectures (IWACA)*, Heidelberg, Germany, 1994.
- [60] E. Schill, S. Khn, and F. Breiter, "Resource Reservation in Advance in Heterogeneous Networks with Partial ATM Infrastructures," in *Proc. of IEEE INFOCOM*, 1997, pp. 612–619.
- [61] S. Schmidt and J. Kunegis, "Scalable Bandwidth Optimization in Advance Reservation Networks," in *Proc. of IEEE ICON 2007*, Nov. 2007, pp. 95–100.
- [62] W. Smith, I. Foster, and V. Taylor, "Scheduling with Advanced Reservations," in *Proc. of IPDPS 2000*, Cancun, Mexico, 2000.
- [63] C.-N. Chuah, L. Subramanian, R. H. Katz, and A. D. Joseph, "QoS Provisioning using a Clearing House Architecture," in *Proc. of IWQOS 2000*, Pittsburgh, PA, 2000.
- [64] S. Tanwir, L. Battestilli, H. Perros, and G. Karmous-Edwards, "Dynamic Scheduling of Network Resources with Advance Reservations in Optical Grids," *Int. J. Netw. Manag.*, vol. 18, no. 2, pp. 79–105, 2008.
- [65] J. Ni, D. H. Tsang, S. Tatikonda, and B. Bensaou, "Optimal and Structured Call Admission Control Policies for Resource-Sharing Systems," *IEEE Transactions on Communications*, vol. 55, no. 1, pp. 158–170, 2007.
- [66] H. Lee, M. Veeraraghavan, H. Li, and E. K. P. Chong, "Lambda scheduling algorithm for file transfers on high-speed optical circuits," in *Proc. of IEEE CCGrid 2004*, Chicago, IL, 2004.
- [67] J. T. Virtamo, "A Model of Reservation Systems," *IEEE Transactions on Communications*, vol. 40, pp. 109–118, 1992.
- [68] T. D. Wallace and A. Shami, "Connection Management Algorithm for Advance Lightpath Reservation in WDM Networks," in *Proc. of BROADNETS 2007*, Raleigh, NC, 2007.

- [69] T. Wallace, A. Shami, and C. Assi, "Scheduling advance reservation requests for wavelength division multiplexed networks with static traffic demands," *IET Communications*, vol. 2, pp. 1023–1033, 2008.
- [70] P. Wieder, O. Waldrich, and W. Ziegler, "Advanced Techniques for Scheduling, Reservation, and Access Management for Remote Laboratories," in *Proc. of the Second International Conference on e-Science and Grid Computing, 2006*, Amsterdam, Netherlands, 2006.
- [71] L. C. Wolf and R. Steinmetz, "Concepts for Resource Reservation in Advance," *Multimedia Tools Appl.*, vol. 4, no. 3, pp. 255–278, 1997.
- [72] L. Wu, C. Wu, J. Cui, and J. Xing, "An Adaptive Advance Reservation Mechanism for Grid Computing," in *Proc. of PDCAT 2005*, Dalian, China, 2005.
- [73] J. Yin, Y. Wang, M. Hu, and C. Wu, "Predictive Admission Control Algorithm for Advance Reservation in Equipment Grid," in *Proc. of the IEEE International Conference on Services Computing (SCC)*, Honolulu, Hawaii, 2008.
- [74] J. Zheng and H. T. Mouftah, "A Framework for Supporting Advance Reservation Service in GMPLS-based WDM Networks," in *Proc. of PACRIM 2003*, Victoria, B.C., Canada, 2003.
- [75] —, "Routing and Wavelength Assignment for Advance Reservation in Wavelength-Routed WDM Optical Networks," in *Proc. of ICC 2002*, New York, NY, 2002.
- [76] —, "Supporting Advance Reservations in Wavelength-Routed WDM Networks," in *Proc. of ICCCN 2001*, Scottsdale, Arizona, 2001.
- [77] J. Zheng, B. Zhang, and H. T. Mouftah, "Toward Automated Provisioning of Advance Reservation Service in Next-Generation Optical Internet," *IEEE Communications Magazine*, vol. 44, no. 12, pp. 68–74, 2006.
- [78] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," DEC Research Report TR-301, Sep. 1984.

- [79] LHC - Large Hadron Collider. [Online]. Available: <http://lhc.web.cern.ch/lhc/>
- [80] Internet2 Network. [Online]. Available: <http://www.internet2.edu/network/>
- [81] ESnet. [Online]. Available: <http://www.es.net/>
- [82] FrameNet: Ethernet-Based Services; Sherpa VLAN Configuration Tool. [Online]. Available: <http://www.nlr.net/framenet.php>
- [83] GEANT2 AutoBAHN. [Online]. Available: <http://www.geant2.net/server/show/nav.756>
- [84] JGN2plus services. [Online]. Available: http://www.jgn.nict.go.jp/english/about_us/service.html
- [85] User-controlled lightpaths (UCLP). [Online]. Available: <http://www.canarie.ca/canet4/uclp/>
- [86] Internet2 DNC Software Suite. [Online]. Available: <https://wiki.internet2.edu/confluence/display/DCNSS/Home>
- [87] T. Armstrong and S. S. Gorshe, Eds., *IEEE Commun. Mag., Special issue on Generic Framing Procedure (GFP) and Data over SONET/SDH and OTN*, vol. 40, no. 5, May 2002.
- [88] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proceedings of the 2000 IEEE INFOCOM Conference*, Tel-Aviv, Israel, Mar 2000.
- [89] D. D. Clark, M. L. Lambert, and L. Zhang, "NETBLT: A Bulk Data Transfer Protocol," IETF RFC 998, Mar. 1987.
- [90] Y. Gu and R. L. Grossman, "UDT: An application level transport protocol for grid computing." *2nd International Workshop on Protocols for Long-Distance Networks PFLDNet*, Feb. 2005.
- [91] A. Banerjee, W. Feng, B. Mukherjee, and D. Ghosal, "RAPID: An end-system aware protocol for intelligent data transfer over lambda-grids." in *Proceedings of 20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, Rhode Island, Greece, 2006.

- [92] C. Jin, D. X. Wei, and S. H. Low, "Fast tcp: Motivation, architecture, algorithms, performance," in *Proceedings of IEEE INFOCOM*, Mar. 2004.
- [93] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," IETF RFC 2581, Apr. 1999.
- [94] X. Zheng and M. Veeraraghavan. CHEETAH end-host software design document. [Online]. Available: <http://cheetah.cs.virginia.edu/software/software-document.pdf>
- [95] Iperf. [Online]. Available: <http://sourceforge.net/projects/iperf>
- [96] Global Environment for Network Innovations (GENI). [Online]. Available: <http://www.geni.net/>
- [97] The VMware Website. [Online]. Available: <http://www.vmware.com>
- [98] Microsoft Virtual Server. [Online]. Available: <http://www.microsoft.com/windowsserversystem/virtualserver>
- [99] Xen. [Online]. Available: <http://xen.xensource.com>
- [100] PlanetLab. [Online]. Available: <http://www.planet-lab.org>
- [101] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI Veritas: Realistic and Controlled Network Experimentation," in *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2006, pp. 3–14.
- [102] J. Turner. A proposed architecture for the GENI backbone platform. [Online]. Available: <http://www.geni.net/GDD/GDD-06-09.pdf>
- [103] The Hybrid Optical and Packet Infrastructure Project (HOPI). [Online]. Available: <http://networks.internet2.edu/hopi/>
- [104] XORP Open Source IP Router. [Online]. Available: <http://xorp.org>

- [105] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, 2000. [Online]. Available: <http://citeseer.ist.psu.edu/article/kohler00click.html>
- [106] Internet2 Network NOC, Global NOC Software Distribution. [Online]. Available: <http://noc.net.internet2.edu/i2network/grnoc-tool-set.html>
- [107] Tools for Bandwidth Estimation. [Online]. Available: <http://www.icir.org/models/tools.html>
- [108] Cisco VFrame Data Center 1.1. [Online]. Available: <http://www.cisco.com/en/US/products/ps8463/index.html>
- [109] M. Kolon, "Intelligent logical router service," White Paper, Juniper. [Online]. Available: http://www.juniper.net/solutions/literature/white_papers/200097.pdf
- [110] "Cisco XR 12000 series service separation architecture tests," White Paper, EANTC, May 2005. [Online]. Available: http://www.eantc.de/fileadmin/eantc/downloads/test_reports/2003-2005/EANTC-Summary-Report-Cisco-12kXR.FINAL.pdf
- [111] J. Heinanen and R. Guerin, "A two rate three color marker," RFC 2698, Sep. 1999.
- [112] libssh2. [Online]. Available: <http://www.libssh2.org>