ABSTRACT

DESIGN AND ANALYSIS OF A SCALABLE TERABIT MULTICAST PACKET SWITCH: ARCHITECTURES AND SCHEDULING ALGORITHMS

by
Feihong Chen

Internet growth and success not only open a primary route of information exchange for millions of people around the world, but also create unprecedented demand for core network capacity. Existing switches/routers, due to the bottleneck from either switch architecture or arbitration complexity, can reach a capacity on the order of gigabits per second, but few of them are scalable to large capacity of terabits per second.

In this dissertation, we propose three novel switch architectures with cooperated scheduling algorithms to design a terabit backbone switch/router which is able to deliver large capacity, multicasting, and high performance along with Quality of Service (QoS). Our switch designs benefit from unique features of modular switch architecture and distributed resource allocation scheme.

Switch I is a unique and modular design characterized by input and output link sharing. Link sharing resolves output contention and eliminates speedup requirement for central switch fabric. Hence, the switch architecture is scalable to any large size. We propose a distributed round robin (RR) scheduling algorithm which provides fairness and has very low arbitration complexity. Switch I can achieve good performance under uniform traffic. However, Switch I does not perform well for non-uniform traffic.

Switch II, as a modified switch design, employs link sharing as well as a token ring to pursue a solution to overcome the drawback of Switch I. We propose a round robin prioritized link reservation (RR+POLR) algorithm which results in an improved performance especially under non-uniform traffic. However, RR+POLR

algorithm is not flexible enough to adapt to the input traffic. In Switch II, the link reservation rate has a great impact on switch performance.

Finally, Switch III is proposed as an enhanced switch design using link sharing and dual round robin rings. Packet forwarding is based on link reservation. We propose a queue occupancy based dynamic link reservation (QOBDLR) algorithm which can adapt to the input traffic to provide a fast and fair link resource allocation. QOBDLR algorithm is a distributed resource allocation scheme in the sense that dynamic link reservation is carried out according to local available information. Arbitration complexity is very low. Compared to the output queued (OQ) switch which is known to offer the best performance under any traffic pattern, Switch III not only achieves performance as good as the OQ switch, but also overcomes speedup problem which seriously limits the OQ switch to be a scalable switch design. Hence, Switch III would be a good choice for high performance, scalable, large-capacity core switches.

DESIGN AND ANALYSIS OF A SCALABLE TERABIT MULTICAST PACKET SWITCH: ARCHITECTURES AND SCHEDULING ALGORITHMS

by
Feihong Chen

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Electrical and Computer Engineering

May 2000

APPROVAL PAGE

DESIGN AND ANALYSIS OF A SCALABLE TERABIT MULTICAST PACKET SWITCH : ARCHITECTURES AND SCHEDULING ALGORITHMS

Feihong Chen

Dr. Ali N. Akansu, Dissertation Co-Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Necdet Uzun, Dissertation Co-Advisor — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Nirwan Ansari, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Symeon Papavassiliou, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. H. Jonathan Chao, Committee Member — Date
Professor of Electrical Engineering, Polytechnic University, Brooklyn, NY

BIOGRAPHICAL SKETCH

Author:	Feihong Chen
Degree:	Doctor of Philosophy in Electrical Engineering
Date:	May 2000

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering
  New Jersey Institute of Technology, Newark, NJ, 2000
- Master of Science in Electrical Engineering
  Beijing University of Posts and Telecommunications, Beijing, P.R.China, 1996
- Bachelor of Science in Electrical Engineering
  Xi'An University of Electronic Science and Technology (XIDIAN University), Xi'An, ShannXi, P.R.China, 1993

Major: Electrical Engineering

Presentations and Publications:

F. Chen, B. Yener, A.N. Akansu and S. Tekinay,
"A Novel Performance Analysis for the Copy Network in a Multicast ATM Switch,"
Proc. of IEEE ICCCN'98, Lafayette, LA, October 12-25, 1998, pp. 99-106.

F. Chen, N. Uzun and A. N. Akansu,
"A High Performance Output-Oriented Cell Scheduling Algorithm for Multicast ATM Switches,"
Proc. of CISS'99, Baltimore, MD, March 17-19, 1999, pp. 803-808.

F. Chen, N. Uzun and A. N. Akansu,
"A Large Scale Multicast ATM Switch With Input and Output Link Sharing,"
Proc. of IEEE GLOBECOM'99, Rio de Janeiro, Brazil, Dec. 1999, pp. 1251-1255.

F. Chen, N. Uzun and A. N. Akansu,
"A Scalable Multicast ATM Switch using Link Sharing and Prioritized Link Reservation,"
Proc. of IEEE ICCCN'99, Boston, MA, October 1999, pp.218-222.

F. Chen, N. Uzun and A. N. Akansu,
"Design of a Large Scale Multicast Packet Switch with a Distributed Resource Allocation Algorithm,"
Proc. of CISS'2000, Princeton, NJ, March 15-17, 2000, pp. FP 6.5 - FP6.10.

F. Chen, N. Uzun and A. N. Akansu,
"A Distributed Dynamic Scheduling Algorithm for a Terabit Multicast Packet Switch,"
to be presented at IFIP NETWORKING'2000, Paris, France, May 2000.

F. Chen, N. Uzun and A. N. Akansu,
" Design and Analysis of a Scalable Terabit Multicast Packet Switch with Dual Round Robin Dynamic Link Reservation,"
to be presented at IEEE ATM Workshop, Germany, June 2000.

F. Chen, N. Uzun and A. N. Akansu,
"A Scalable Terabit Multicast Packet Switch with Link Sharing and Dual Round Robin Dynamic Link Reservation,"
submitted to the IEEE/ACM Transaction on Networking, January 2000.

F. Chen, N. Uzun and A. N. Akansu,
"A Scalable Terabit Core Switch for Broadband Internet,"
submitted to IEEE MILCOM'2000, Los Angeles, LA, October 2000.

This work is dedicated to my husband and our parents.

ACKNOWLEDGMENT

First and foremost, I would like to express my sincere gratitude to my advisors and mentors, Dr. Ali N. Akansu and Dr. Necdet Uzun. I appreciate Dr. Akansu for his constant support, guidance, and encouragement in every step of my way. I would like to thank Dr. Uzun, an excellent advisor and a good friend, for his inspiration and valuable discussion; for the time he has put into helping me to make this dissertation the comprehensive work it is today; and for his belief in my abilities and encouraging me to aim high.

I would like to thank my dissertation committee, Dr. Nirwan Ansari, Dr. Symeon Papavassiliou, and Dr. H. Jonathan Chao, for their being interested in this work and stimulating many valuable discussions and insightful comments. I would also like to extend my appreciation to Dr. Bulent Yener and Dr. Sirin Tekinay for their help at the beginning of this research work.

Likewise, I would like to thank Ms. Anne McMahon, who is always there for us and helps a lot in many ways. I would also like to extend my appreciation to the staff of the ECE department for their constant assistance in just about everything.

Many thanks are also due to my colleagues at NJCMR and CCSPR, especially my good friends Mahalingam Ramkumar, Xiaodong Cai, Minyi Zhao, Pingping Zong, Bin He and Anil Bircan, for all the help they have offered towards completion of this dissertation, and for a lot of fun we shared together. I would also like to thank Xueming Lin, Ziqiang Xu, and Jianguo Chen for their continues help and advise from the first day I joined NJIT.

Finally, words can never express the gratitude I feel toward my husband and our parents, whose support provides the foundation upon which this work and other achievements were built.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF TABLES
(Continued)

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES (Continued)

CHAPTER 1

INTRODUCTION

1.1 Motivation

Internet today is under tremendous growth and enjoys its world wide success. The scalable and distributed nature of the Internet attracts more and more users and service providers. Meantime, many emerging applications demand increased bandwidth and generate a huge volume of traffic. It creates an unprecedented demand for core network capacity. Also, the exponential growth of traffic may cause several problems in the network such as congestion, unpredictable delay, insufficient reliability and low availability. Facing those challenges, the core switch/router is therefore required to be able to deliver higher performance in terms of large capacity, high speed, multicasting as well as the Quality of Service (QoS).

In a word, scalable multi-terabit multicast switches/routers are in demand. However, existing switches/routers, due to the bottleneck from either switch architecture or arbitration complexity, can reach a capacity on the order of gigabits per second but few of them is scalable to terabit capacity.

In this dissertation, we propose several switch architectures and scheduling algorithms to approach the desired scalable terabit multicast packet[1] switch.

1.2 Review

In the history of switch design [1] [2], various multicast ATM switches have been proposed in literature. As shown in Fig 1.1, switch fabric on which a switch architecture is built can be classified into three types : Banyan network, Crossbar network, and Clos network. Starlite switch [3], Turner's broadcast switch [4] and Lee's

[1]In this dissertation, a packet has fixed length of 53 bytes.

multicast switch [5] were the typical multicast switches based on Banyan network. Later on, a practical version of Lee's switch is proposed in [6]. And another advanced switch with fault-tolerant multistage interconnection network (MIN) switch is presented in [7]. Those switches have an advantage of a reduced hardware complexity. But, internal path conflict and head of line (HOL) blocking are the challenges for those switches to achieve high performance and scalability. One of the switches built on Crossbar network is Knockout Multicast switch [8], which utilizes a concentrator in every output port to resolve output contention. Following Knockout Multicast switch, Crossbar switch [9], Shared Concentration and Output Queueing Multicast (SCOQ) [10], Multicast Output Buffered ATM Switch (MOBAS) [11], Abacus [12], and a growable multicast switch [13] were proposed. Crossbar switches can achieve high performance because of output queueing and output contention resolution. The tradeoff is the cost of hardware complexity and speedup required. Growable packet switch [14] and ring sandwich network [15] were the multicast ATM switches based on Clos network. [20] presents a performance study for a buffered Clos switch. In fact, Clos network belongs to MIN but it only has 3 stages. Since Clos network can provide multiple paths from an input port to an output port, internal path conflicts are relaxed. Clos network has better performance than Banyan network but it has higher hardware complexity.

Existing packet switches including above multicast switches are able to achieve Gigabit/sec capacity. But, few of them provides further scalability to Terabit/sec. Besides the restraint from switch fabric, queueing strategy and cooperated scheduling scheme have a great impact on switch scalability as well. From switch buffering point of view, switches can be classified into output-queued (OQ) switches[2], input-queued (IQ) switches, and input-output-queued (IOQ) switches.

[2]OQ switches include centralized shared memory switches.

Switch Proposed Year	Based on Crossbar network/ Knockout switch	Based on Banyan Network		Based on Clos Network
1984		Starlit switch (Huang et al.)		
1988	Multicast Knockout switch (K.Y.Eng et al.)	Turner's broadcast switch	Lee's multicast switch	
1990	Gauss ASE (R.J.F. Vries)			
1991		Sunshine switch (J.N.Giacopelli et al.)	Recursive multistage structure (R. Cusani et al.)	Growable packet switch (D.J.Marchok et al.)
1994	SCOQ switch (D.X.Chen et al.)	LGMIN (W.D. Zhong et al.)	Multinet switch (H.S. Kim)	
1995	MOBAS (H.J. Chao et al.)			Ring sandwich network (Y.Yang et al.)
1997	Abacus switch (H.J. Chao et al.)		Guo & Chang's switch	

Figure 1.1 Review on multicast ATM switches

Fig 1.2 depicts a general model of an OQ switch. *OQ switches*, such as [8, 11, 12, 13, 16, 17, 18, 19, 21], proved to maximize throughput and optimize latency. Hence, OQ switches are able to provide Quality of Service (QoS) guarantees [22, 23, 24, 28]. But, switch fabric and output buffers have to operate N (N is the switch size in terms of the number of switch inputs/outputs) times as fast as the line rate, because cells arriving at switch inputs have to be delivered to and stored in output queues in a same cell slot. It may be practical to implement an output queued switch or router with an aggregated bandwidth of several 10Gb/s. But, it is not feasible to design a large OQ switch with fast line rate, because memory access speed achieved in commercial is not fast enough to support N times speedup.

On the other hand, *IQ switches* (see Fig 1.3) become more attractive because switch fabric and input memory only need to run as fast as the line rate. An IQ switch with FIFO queues is known to suffer head of line (HOL) blocking which limits the throughput to $(2 - \sqrt{2}) = 58.6\%$. To overcome HOL blocking, virtual

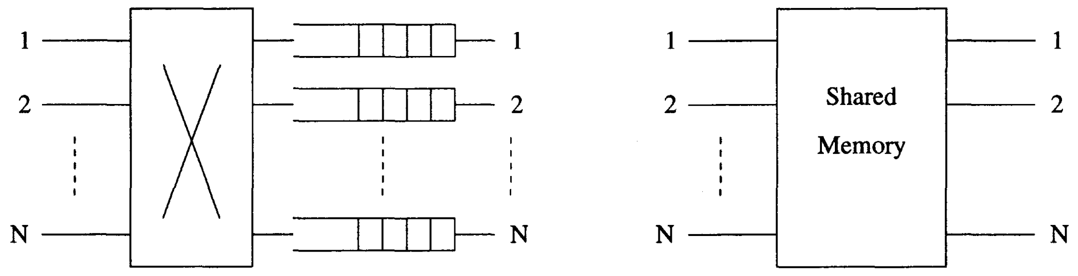


Figure 1.2 A general architecture of an output queued (OQ) switch

output queues (VOQs) are applied in every switch input together with scheduling algorithms like Longest Queue First (LQF)[30], Oldest Cell First (OCF)[31], Longest Port First (LPF) [32] to achieve 100% maximized throughput. To support multicast traffic, TATRA and WBA were proposed for IQ switches [33] [34]. A combined input output queued (CIOQ) switch has been proposed[35] and demonstrated that the CIOQ switch can precisely emulate the OQ switch when speedup $S \geq 2 - \frac{1}{N}$. In addition, [36] [37] [38] propose some priority queueing algorithms for integrated traffic.

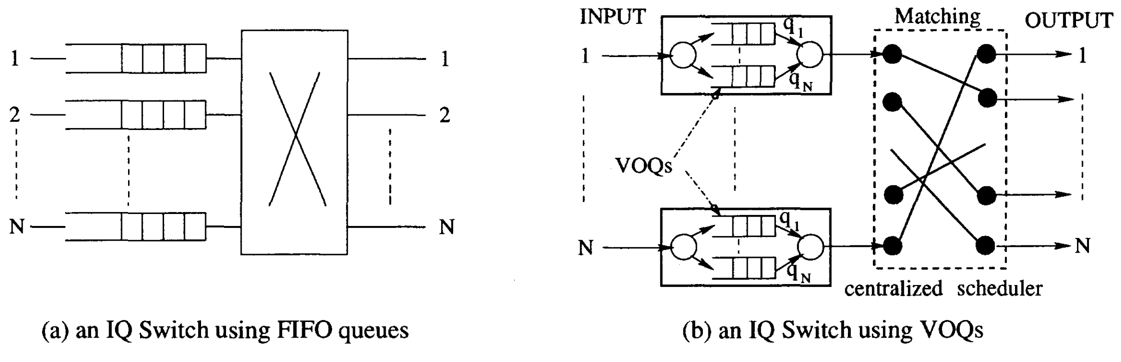
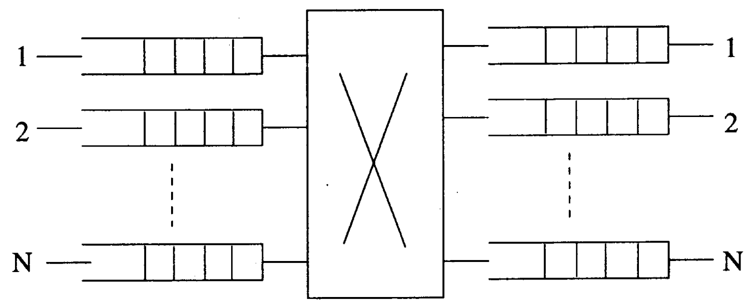

INPUT
Matching
OUTPUT
VOQs
centralized scheduler
(a) an IQ Switch using FIFO queues
(b) an IQ Switch using VOQs

Figure 1.3 A general architecture of input queued (IQ) switch : (a) an IQ switch using FIFO queues ; (b) an IQ switch using VOQs .

Though IQ switches are capable of supporting high speed line rate without any speedup in hardware, scheduling arbitration complexity of at least $O(N^{2.5})$ is a big obstacle if IQ switches grow to a large size. The reason is that, most scheduling algorithms proposed for IQ switches employ a centralized scheduler, which needs to collect traffic information from N switch inputs in every cell slot and consumes multiple iteration to determine the final input-output matching. Situation may become more complex under multicast traffic. As scheduling complexity increases with switch size N, an IQ switch using a centralized scheduler has difficulties in growing to a large switch size and terabit/sec capacity.

IOQ Switches are combinations of IQ switches and OQ switches (refer Fig 1.4). As comparison study in [39], OQ switches deserve the best throughput/delay performance for arbitrary traffic distributions. However, since the current memory access time is limited to a few nsec by state-of-the-art integrated circuit technology, output-buffered switch architecture is not scalable for large-capacity systems. On the other hand, IQ switches endures poor throughput/delay performance because of HOL blocking, but input-queued architecture is feasible to extend. The IOQ switch is a solution by trading off the high performance of the OQ switch and the low hardware complexity of the IQ switch.

One of few existing IOQ switch designs is CIOQ switch [35]. But, the reason for CIOQ switch in [35] to adopt both input queueing and output queueing is to provide QoS in IQ switches. As speedup is required in IQ switches for QoS purpose, output queueing is needed to avoid cell loss. CIOQ switch, in fact, can be classified as an IQ switch. The centralized scheduler sustains an arbitration complexity of $O(N^{2.5})$ so that CIOQ switch [35] is not scalable. In addition, the modular batcher-binary-banyan switch [40] proposed by T.T.Lee and Sunshine switch [41] proposed



Input-Output (IOQ) Queued Switch

Figure 1.4 A general architecture of input-output queued (IOQ) switch

by Bellcore are also IOQ switches. But, because of irregular interconnection pattern in hardware, those switches are limited to up to 20Gb/s.

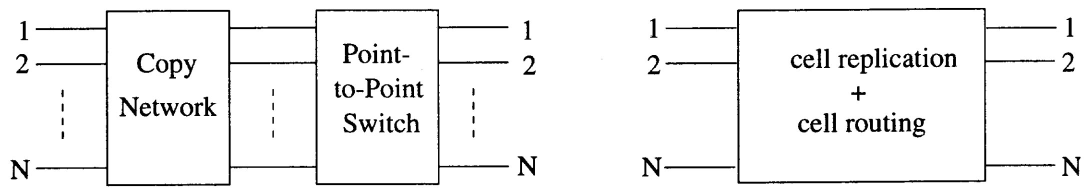
1.3 Design Issues

Several issues should be considered when we design a large-capacity switch. In this section, we mainly address following aspects which are targeted in our design of a scalable terabit multicast packet switch.

1.3.1 Multicasting

In today's B-ISDN and Internet, many services, such as teleconferencing, entertainment video, distributed data processing, are characterized by point(multipoint)-to-multipoint communication. Switches need to support not only point-to-point connections, but also multipoint connections. Multicast switch is a solution for sending information from one sender to a group of receivers.

Multicast functions in ATM switches can be implemented either with a separate nonblocking copy network followed by a point-to-point routing network, or with an



(a) a multicast switch consists of a copy network and a point-to-point switch

(b) a multicast switch with an integrated function of cell replication and cell routing.

Figure 1.5 Multicast switches

integrated switching fabric performing both replication and routing functions. Fig 1.5 illustrate the two alternatives.

The architecture of a nonblocking copy network followed by a traditional point-to-point ATM switching network is adopted by many commercially available switches [3, 4, 5, 6], because the traditional switch doesn't need to change completely but only adding a copy network ahead. Copy network replicates an input multicast cell [3] to the number of cell copies. Then the cell copy is routed to an output line through a point-to-point routing network. But, the copy network faces the problem of overflow which may cause performance degradation. In addition, there is an implementation redundancy by separating copy network and routing network. It increases hardware complexity.

Another architecture of multicast switch is shown in Fig 1.5(b). Cell duplication and cell routing are integrated together in implementations. For example, [7, 11, 12, 16, 17, 18, 19, 20, 43, 44] are the switches using either output buffer or shared-memory to handle the cell copy and to schedule cells at the same time. And the multicast IQ switches belong to this type of switch architecture. [28] is a typical shared-memory architecture combining cell duplication and cell delivery. Most

[3]In this paper, the multicast cell is defined as a cell with one destination or multiple destinations.

recent switch designs adopt this integrated switch architecture to reduce hardware complexity and achieve an efficient buffer management as well.

1.3.2 Scalability

Scalability can be evaluated from two aspects — capacity and expandability. Internet applications continue to grow and create an ever-increasing demand for bandwidth. Switches have to be scalable to avoid being frequently re-architectured in order to support massive increase of traffic. Thus, core switches face an emerging challenge to provide more than 100Gb/s even Terabit/s capacity. Existing switches using current state-of-the-art technology can obtain a capacity up to several 10Gb/s, but are not easy to pursue Terabits/sec due to some constrains such as memory access rate or arbitration complexity. For example, shared-memory switches are optimal in performance and also cost effective. But, switch size and capacity of shared-memory is ruled by the fact that :

$$\frac{R * N * 2}{\text{cell slot}} \leq \frac{1}{\text{RAM read/write cycle}} \tag{1.1}$$

where R is the input line rate, and N is the number of switch inputs (outputs). Bounded by the RAM read/write rate, it is observed that shared-memory switch is not able to gear to the high capacity expectation.

In addition to capacity, another necessary requirement for scalability is expandability. It considers whether switch architecture supports increased speeds or additional switch ports, and how flexible the switch can be to pursue an expanding configuration. The best solution would be a modular switch architecture.

1.3.3 Low Complexity

Both hardware complexity and scheduling arbitration complexity must be minimized. Hardware complexity is often measured in terms of logic gate counts, chip pinout, memory speed, implementation costs. From prototype design to real implementation, above concerns should be carefully evaluated. For example, the multicast switch using copy network usually has implementation redundancy and incurs high hardware costs. In addition, some switches such as the OQ switch are limited by the memory access speed because of the up to N times speedup required. The switch fabric with shuffle connection from N switch inputs to N switch outputs gains reliability but pays for high connection cost. In short, an efficient switch design should minimize the hardware complexity but without sacrificing reliability and performance.

Apart from hardware complexity, arbitration complexity should be low to gear up the hardware design. The IQ switch, for example, is better than the OQ switch in the aspect of hardware complexity. But, the IQ switch uses a centralized scheduler to resolve HOL blocking so that the IQ switch tolerant a high arbitration complexity of at least $O(N^{2.5})$. The arbitration complexity hinders the IQ switch to build a large scale switch.

In summary, we may need to trade off between the hardware complexity and arbitration complexity in order to pursue a good solution based on some specific design requirements.

1.3.4 High Performance

Switches should provide satisfactory performance. Bellcore has recommended performance requirements and objectives for a Broadband Switching Systems (BSS) [42]. Table 1.1 defines three classes of Quality of Service (QoS) and explains the associated performance objectives.

Table 1.1 Performance requirements and objectives for BSS [42]. * : includes non-queueing related delays but excludes propagation, and does not include delays due to processing above ATM layer. N/S : not specified.

Performance Parameters	CLP	QoS 1	QoS 3	QoS 4
Cell Loss Ratio	0	$< 10^{-10}$	$< 10^{-7}$	$< 10^{-7}$
Cell Loss Ratio	1	N/S	N/S	N/S
Cell Transfer Delay (99 percentile) *	1/0	150 μs	150 μs	150 μs
Cell Delay Variation (10^{-10} quantile)	1/0	250 μs	N/S	N/S
Cell Delay Variation (10^{-7} quantile)	1/0	N/S	250 μs	250 μs

QoS class 1 is dedicated to cell loss sensitive applications. It corresponds to AAL layer class A service which is defined by ITU-T XIII Group and ATM Forum. *QoS class 3* is applied for low latency, connection-oriented data transfer applications which is intended for AAL class C service. In addition, *QoS class 4* is related to low latency, connectionless data transfer applications which is for AAL class D service.

The performance parameters include *cell loss ratio*, *cell transfer delay*, and *cell delay variation*. The performance objectives associated to a QoS class are determined by the status of the cell loss priority (CLP) bit in the ATM cell header. End users can initialize the CLP bit but switches along the connection path can change it according to network conditions.

For all three QoS classes, the probability of cell transfer delay greater than 150μs is guaranteed to be less than 1 percent, i.e. :

$$\text{Pr. } [\text{ cell transfer delay} > 150\ \mu\text{s }] \ < \ 0.01$$

The probability of cell delay variation (CDV) greater than 250μs is required to be less than 10^{-10} for QoS class 1, and to be less than 10^{-7} for QoS class 3/4.

In addition to above performance objectives, switches need to be flexible to cooperate other technologies such as connection admission control, buffer management, traffic engineering in order to provide Quality of Service (QoS) guarantees.

1.4 Outline

Our goal is to design a scalable terabit multicast packet switch which is capable of multicasting, large capacity, low complexity, modular configuration, and high performance. In this dissertation, we propose three switch architectures with cooperated scheduling algorithms — namely Switch I, Switch II, and Switch III, to achieve the desired switch. Our designs benefit from unique features of modular switch architecture and distributed scheduling arbitration.

In chapter 2, we first present a theoretical work on the performance of copy network under three scenarios : (1) Non-Buffer-NonSplitting copy network (NBNS). (2) Shared-Input-Buffer-NonSplitting copy network (SIBNS). (3) Shared-Input-Buffer-Splitting copy network (SIBS). For NBNS, we derived the exact overflow and cell loss probabilities instead of the Chernoff Bound [5]. Furthermore, we propose a general Markov Model, a novel theoretical approach, for the performance analysis of the Shared-Input-Buffer copy networks. This analysis method can be applied for both SIBNS and SIBS. Theoretical and simulation results are compared for every scenario.

In chapter 3, we propose a novel switch design, namely Switch I, using input and output link sharing. Switch inputs and outputs are grouped into small modules called Input Shared Blocks (ISBs) and Output Shared blocks (OSBs). Link sharing resolves output contention and eliminates the speedup requirement for central switch fabric.

Two Round Robin (RR) scheduling algorithms are proposed. Both schemes provide a group mapping from an ISB to an OSB. Scheduling complexity is dramatically reduced. The switch can easily extend to high capacity and large scale. Performance evaluation demonstrates that the switch can achieve good performance under uniform multicast[4] traffic. However, isolated Input Shared Blocks (ISBs) prevent switch from achieving high performance under non-uniform traffic.

To overcome the weakness of Switch I, in chapter 4, we present Switch II, a modified switch design using link sharing and prioritized link reservation. ISBs are connected by a token ring. We propose a Round Robin Prioritized Output Link Reservation (RR+POLR) algorithm to allocate link resource and alleviate starvation of OSBs. Switch II obtains an improved performance under non-uniform traffic. But, RR+POLR algorithm is not flexible enough to adapt the dynamic traffic timely. Switch performance is highly determined by how fast link reservation rate the switch can pursue.

Switch III, as an enhanced switch design using link sharing and dual round robin dynamic link reservation, is finally proposed in chapter 5. Unlike the previous two switches, ISBs are connected by dual rings on which K link request tokens (REQs) and K link release tokens (RELs) circulate in a round robin manner. Cell delivery is based on link reservation in every ISB. We propose two Queue Occupancy Based Dynamic Link Reservation (QOBDLR) algorithms to achieve a fast and fair link resource allocation among ISBs. QOBDLR is a distributed link reservation scheme in a way that every ISB, according to its local information, can dynamically increase/decrease its link reservation by "borrowing" or "lending" links from/to each other. Arbitration complexity is $O(1)$. Switch III is competitive to OQ switches in

[4]In this work, multicast traffic includes unicast traffic, i.e., a multicast cell may have one or multiple destinations.

the sense that Switch III not only can achieve a comparable performance to OQ switches under any traffic pattern but also can eliminate N times speedup required in OQ switches.

At last, conclusion is drawn and future work is addressed in chapter 6.

CHAPTER 2

A NOVEL PERFORMANCE ANALYSIS FOR THE COPY NETWORK IN A MULTICAST ATM SWITCH

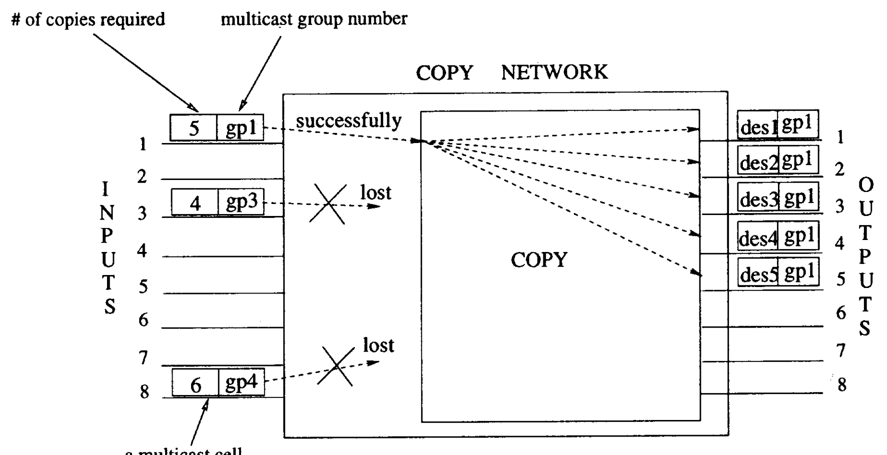
2.1 Introduction

To accommodate the growing demands for a wide class of services, such as voice, data, teleconferencing and entertainment video, a broadband packet network needs to support not only point-to-point connections, but also multipoint connections. Multicast switching is a solution for delivering information from a given source to a group of destination.

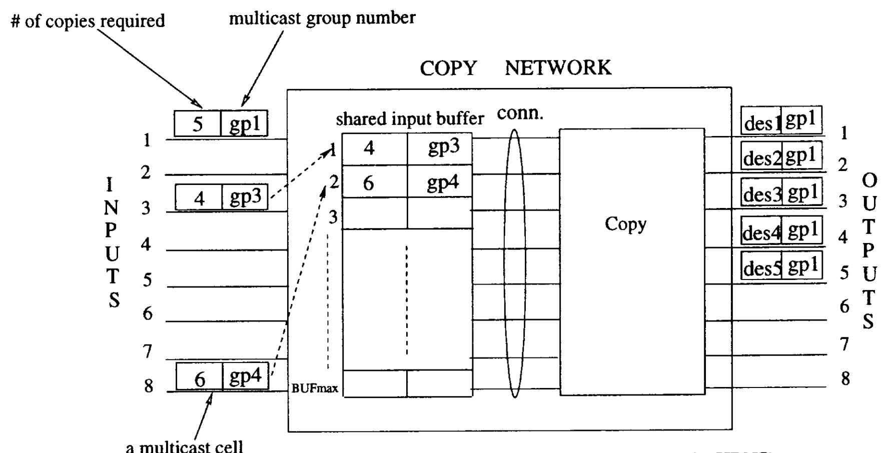
A conventional architecture of multicast ATM switches consists of a nonblocking copy network followed by a traditional point-to-point ATM switching network [4][5][26][47][49]. It provides point-to-multipoint connections by performing two operations : packet replication and packet switching. The function of copy network replicates an incoming cell to the number of required copies.

By applying a self-routing non-blocking fabric, the copy network does not have any internal conflict. But, the copy network faces the problem of overflow if the total copies required exceed the number of output lines of the network. Various scheduling algorithms[47][50][52] to maximize throughput of the copy network were proposed. They introduce additional buffers (input/output/central buffer) and/or scheduling algorithms (one-shot, splitting, etc.), in order to maximize the number of cell copies injected to the point-to-point switching network.

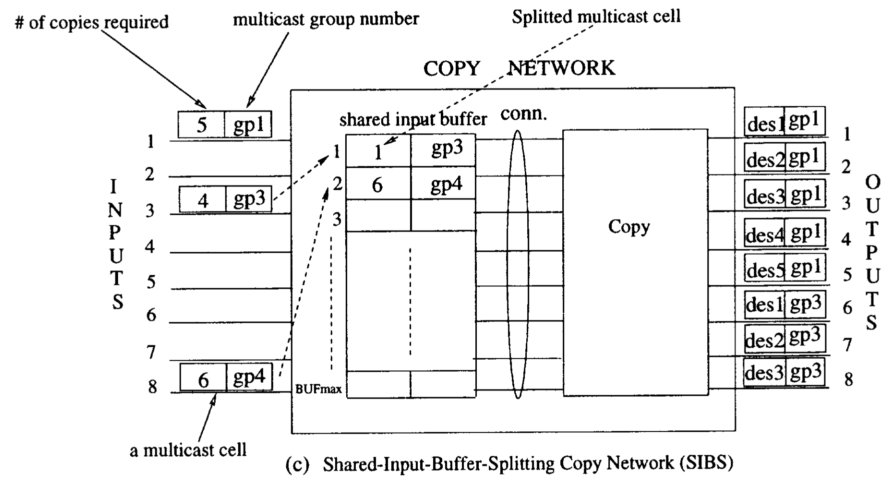
In this chapter, we present a theoretical work on the performance of the copy network in three typical scenarios (shown in Fig 2.1). In Non-Buffer Non-Splitting (NBNS) copy network (Fig 2.1(a)), all the copies required by a multicast cell are replicated in the same time slot. The copy network has no inside buffer to save blocked cells. NBNS causes high cell loss. To prevent the blocked cells from being



(a) Non-Buffer-NonSplitting Copy Network (NBNS)



(b) Shared-Input-Buffer-NonSplitting Copy Network (SIBNS)



(c) Shared-Input-Buffer-Splitting Copy Network (SIBS)

Figure 2.1 Three scenarios of copy network in a multicast ATM switch

lost, we introduce a shared input buffer in the copy network. Two scheduling algorithms are considered for the Shared-Input-Buffer copy network : *Non-Splitting* algorithm (SIBNS) (Fig 2.1(b)), all the copies required by a multicast cell are replicated in a same time slot; *Splitting* algorithm (SIBS) (Fig 2.1(c)), a multicast cell can be partially copied in a time slot, and the remains can be delayed to the next time slot.

For NBNS, we derived the exact overflow and cell loss probabilities instead of the Chernoff Bound [5]. Furthermore, we propose a novel theoretical approach based on a general Markov model, for the performance analysis of the Shared-Input-Buffer copy networks. This analysis method can be applied for both SIBNS and SIBS. Both theoretical analysis and simulation results are presented for every scenario. The comparison shows that shared-input-buffer (SIBNS and SIBS) can obtain an improved performance with lower cell loss and higher throughput. However, the tradeoff is long cell delay. With the splitting algorithm, SIBS can provide better performance than NBNS and SIBNS.

This chapter is organized as follows. In Section 2.2, we provide several notations and assumptions that we use throughout this chapter. Section 2.3 presents performance analysis for NBNS copy network. In Section 2.4, we propose a general Markov Model for the performance analysis of both SIBNS and SIBS copy networks. The analysis model is examined by the numerical and simulation results. Conclusions are finally drawn in Section 2.5.

2.2 Notation and Assumptions

We assume that : (1) input lines are independent and identically distributed; (2) cells' arrival is Poisson process. If an input line has cells arriving, this input line is an active line, otherwise, it's an idle line.

N : size of the copy network.(for 8inputs/8outputs copy network, N=8);

C_{max} **:** the maximum number of copies allowed for every multicast cell, $0 < C_{max} \leq N$;

C : random variable, represents the number of copies required. Assumed to be uniformly distributed;

C_k **:** Probability that the number of copies is k , i.e. pdf of random variable C;

$$C_k = P(C = k) = \frac{1}{C_{max}}, \quad k = 1, 2, ...C_{max} \tag{2.1}$$

P_{in} **:** Probability that an input line is active,

$$P_{in} = P(\text{the number of arriving cells} > 0) = 1 - e^{-\lambda t} \tag{2.2}$$

CP_i **:** random variable, i.e. the number of copies required by the i^{th} multicast cell in the processing queue ;

$CP_i(k)$ **:** Probability that the i^{th} multicast cell in the processing queue (or the multicast cell on the i^{th} input line) needs k copies, $k = 0, 1, 2, ...C_{max}$,

$$CP_i(k) = \begin{cases} 1 - P_{in} & \text{if } k = 0 \\ P_{in}C_k & \text{if } k = 1, 2......C_{max} \end{cases} \tag{2.3}$$

E[λ] : average input load from N input lines;

$$\begin{aligned} E[\lambda] &= \sum_{i=0}^{N} iP(IN = i) \\ &= \sum_{i=0}^{N} i * \binom{N}{i} P_{in}^{i}(1 - P_{in})^{N-i} \end{aligned} \tag{2.4}$$

$P(IN = i)$ is the probability that there are i multicast cells arriving from N input lines in a time slot.

2.3 Performance Analysis of NBNS Copy Network

Assume that, in every cell slot, the copy network serves incoming multicast cells from the 1st input line to the N^{th} input line (i.e., top-down order). If the total number of desired cell copies exceeds the size of the copy network, some multicast cell(s) arrived at the later input lines will be discarded.

$P_{ovfl}(i)$: Overflow probability of the i^{th} input line. When total copies required by the multicast cells of the first i active input lines exceeds the size of the copy network, overflow occurs in the i^{th} input line. The multicast cell on the i^{th} input line cannot be copied in this time slot.

$$\begin{aligned} P_{ovfl}(i) &= P(CP_1 + CP_2 + \cdots + CP_i > N) \\ &= \sum_{j=N+1}^{i*C_{max}} P(CP_1 + CP_2 + \cdots + CP_i = j) \end{aligned} \tag{2.5}$$

$P(CP_1 + CP_2 + \cdots + CP_i = j)$ can be obtained by the convolution of $CP_i(k)$.

P_{lost} : The average cell loss probability. Overflow results in cell loss. The first input line which may overflow starts is the $(\lfloor \frac{N}{C_{max}} \rfloor + 1)^{th}$ input.

$$\begin{aligned} P_{lost} &= \sum_{i=\lfloor \frac{N}{C_{max}} \rfloor +1}^{N} P(\text{the } i^{th} \text{ input line is active}) * P(i^{th} \text{ input line is overflow}) \\ &= \frac{1}{N} \sum_{i=\lfloor \frac{N}{C_{max}} \rfloor +1}^{N} P_{ovfl}(i) \end{aligned} \tag{2.6}$$

Throughput : The average number of multicast cells successfully passing through the copy network per time slot ;

$$Throughput = E[\lambda] * (1 - P_{lost}) = \left[\sum_{i=0}^{N} iP(IN = i) \right] (1 - P_{lost}) \tag{2.7}$$

The performance of NBNS copy network is examined in Fig 2.2, Fig 2.3 by simulation (+) and numerical (-.) results. Fig 2.2 shows the overflow probability of each input line. There is an unfairness : the later input line will have higher overflow probability. Our analysis provides an exact overflow probability, while the Chernoff Bound [5] is much looser. Fig 2.3 illustrates cell loss and throughput. Large copy load (C_{max}) and heavy input load (P_{in}) incur more cell loss and less throughput. NBNS does not introduce any cell delay in copy network.

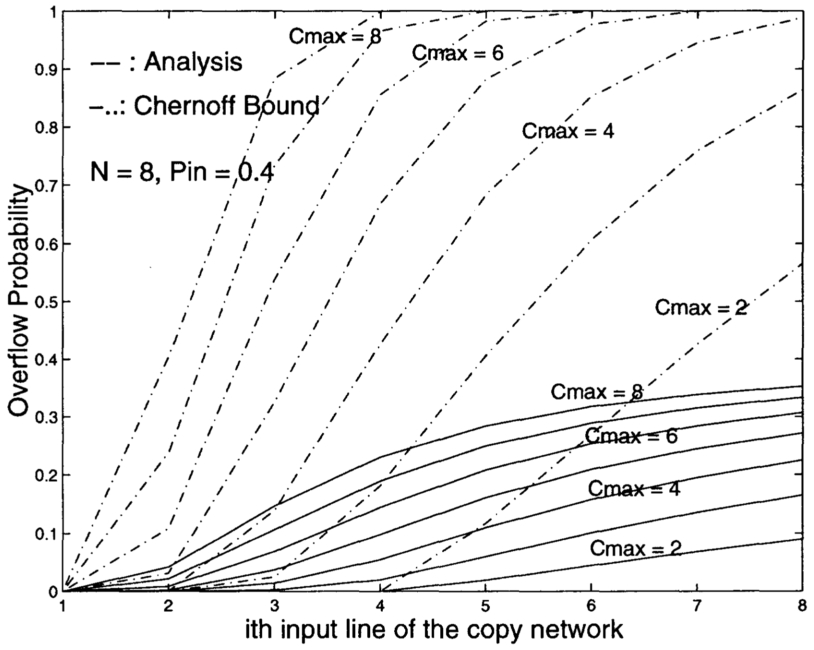
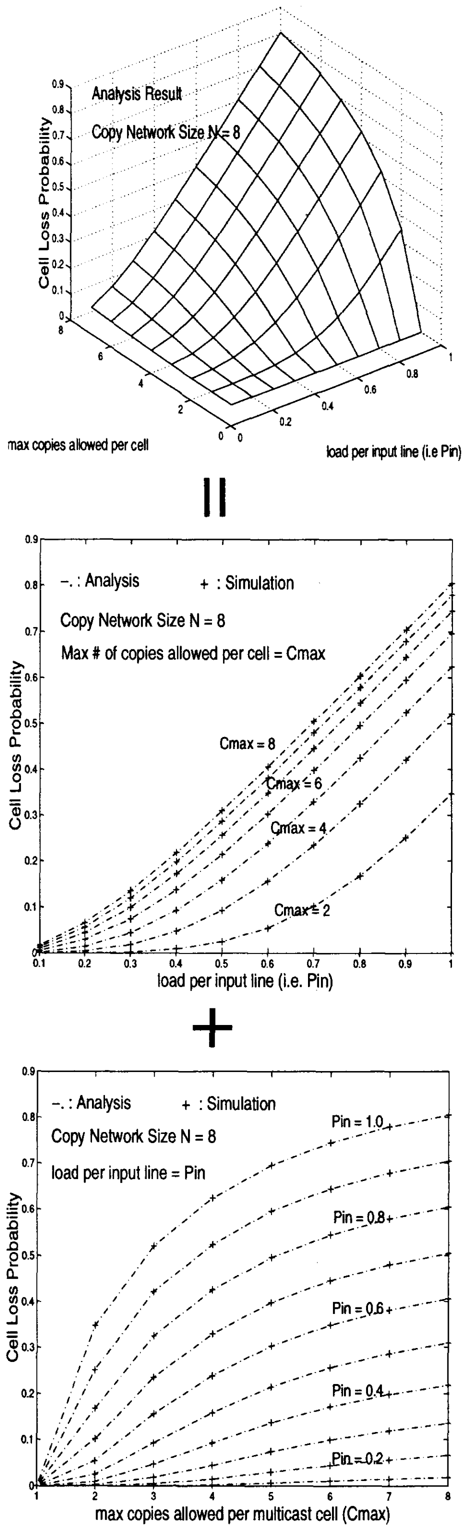


Figure 2.2 Overflow probability in NBNS

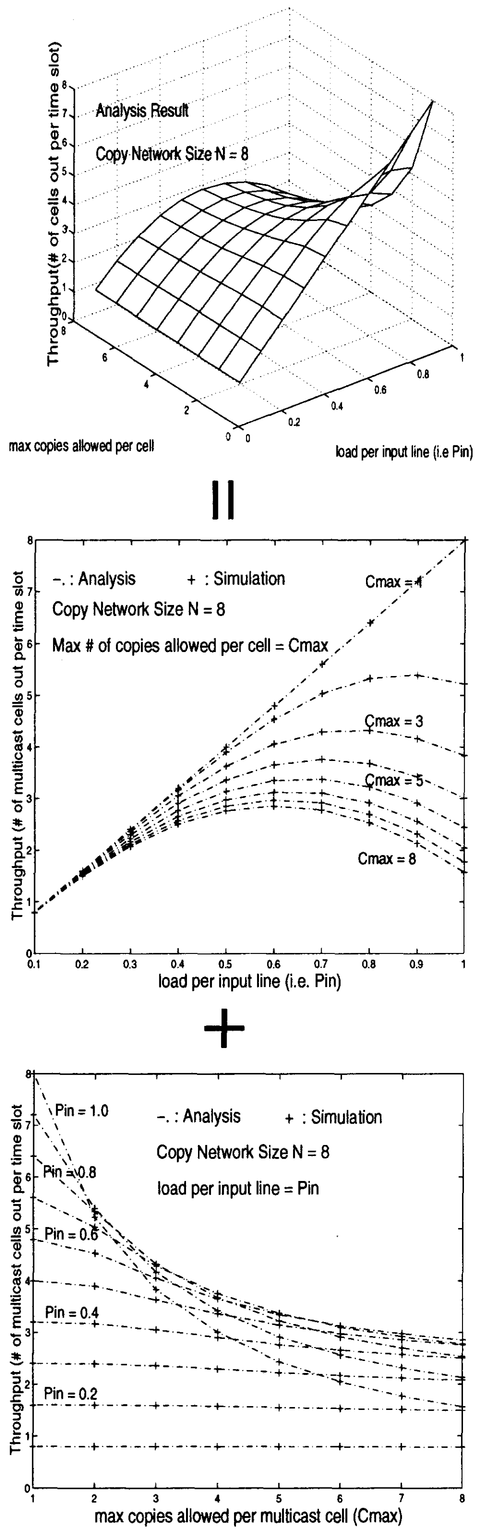
2.4 Performance Analysis for both SIBNS and SIBS Copy Networks

To improve the performance of copy network, a solution is to apply additional buffers [47][48][50][51]. In this paper, we focus on the shared input buffer with two scheduling methods (NonSplitting and Splitting algorithms).

SIBNS : In Shared-Input-Buffer Non-Splitting scenario, cell copies belonged to a same multicast cell should be delivered in a same cell slot. Otherwise, the multicast



(a) Cell Loss vs. Cmax and Pin



(b) Throughput vs. Cmax and Pin

Figure 2.3 Cell loss and throughput in NBNS

cell is blocked in the shared buffer with whole copy requirements. Buffered cells have higher priority to be served than a new arriving cell.

SIBS : In Shared-Input-Buffer Splitting scenario, the copy network can make partial copies for a multicast cell. The splitted cell is saved into shared buffer with remained copy requests.

2.4.1 Notation and Assumption

BUF_{max} : the maximum size of the shared input buffer.

BUF_m : the length of the shared input buffer at the end of the m^{th} time slot.

IN_m : the number of new arriving cells from N inputs in the m^{th} time slot. In every time slot, at most 1 cell comes into the copy network from each input line.

$$P(IN_m = n) = \begin{pmatrix} N \\ n \end{pmatrix} P_{in}^n (1 - P_{in})^{N-n} \tag{2.8}$$

OUT_m : the number of multicast cells successfully delivered out of the copy network in the m^{th} time slot. In a time slot, at most N multicast cells can go through the copy network (when each cell just needs 1 copy). The probability distribution of OUT_m is :

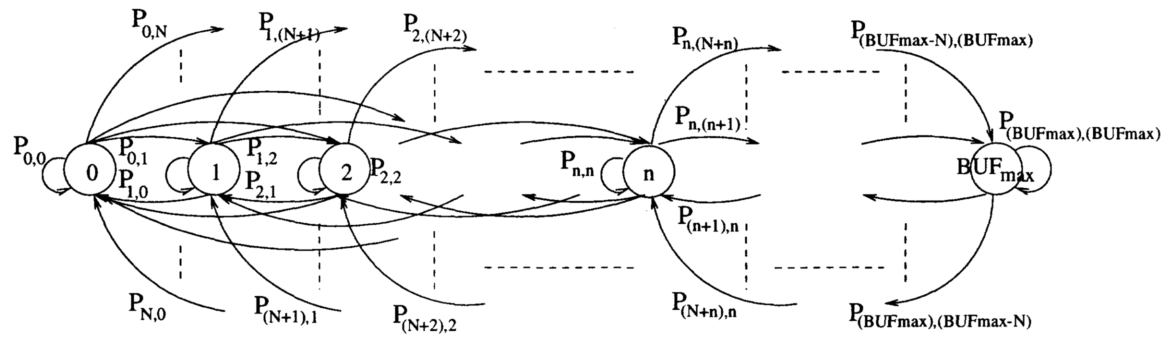
$$P(OUT_m = n) \overset{\text{def}}{=} P(CP_1 + CP_2 + ... + CP_n \leq N) \tag{2.9}$$

$LOST_m$: the number of cells lost in the m^{th} time slot.

$$LOST_m = BUF_{m-1} + IN_m - OUT_m - BUF_{max}; \text{ where } LOST_m \geq 0 \tag{2.10}$$

2.4.2 The Proposed Markov Model

In Fig 2.4, we propose a general Markov Model for SIBNS and SIBS. Each state indicates current queue length in the shared buffer, i.e., how many multicast cells are waiting in the shared memory.



NOTE : Pij is the state transition probability. Pij = 0 if |i-j|> N

Figure 2.4 The general Markov Model for both SIBNS and SIBS

The model we propose is unique in the sense that each multicast cell occupies only one unit in the buffer, no matter how many copies it requires. It can be applied to many different scheduling algorithms, buffer and copy network sizes.

2.4.2.1 State Transition Probabilities $P_{i,j}$: Let $P_{i,j}$ be the state transition probability from state i to state j. We have a State Transition Probability Matrix $\hat{P}_t$ shown in Fig 2.5. Note, the maximum number of states jumped in a cell slot is constrained by the size of the copy network, i.e. $P_{ij} = P(BUF_m = j|BUF_{m-1} = i)$, where $|i-j| \leq N$.

To figure out the state transition probability matrix $\hat{P}_t$, we separately look for three main parts as :

- $P_{i,0}$, where $0 \leq i \leq BUF_{max}$;

$$\begin{aligned} P_{i,0} &= P(BUF_m = 0|BUF_{m-1} = i) = \sum_{n=0}^{N-i} P(IN = n)P(OUT = i+n) \\ &\text{where } 0 \leq i \leq BUF_{max}, \text{ and } i+n \leq N \end{aligned} \tag{2.11}$$

$$P(OUT = i+n) \quad = \quad P(CP_1 + \cdots + CP_{i+n} \leq N)$$

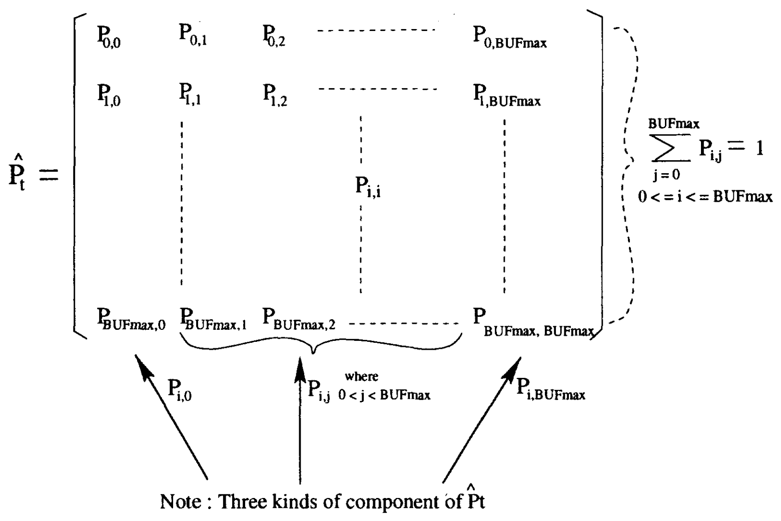

$\hat{P}_t =$
$P_{0,0}$ $P_{0,1}$ $P_{0,2}$ $P_{0,BUFmax}$
$P_{1,0}$ $P_{1,1}$ $P_{1,2}$ $P_{1,BUFmax}$
$P_{i,i}$
$P_{BUFmax,0}$ $P_{BUFmax,1}$ $P_{BUFmax,2}$ $P_{BUFmax, BUFmax}$
$\sum_{j=0}^{BUFmax} P_{i,j} = 1$
$0 <= i <= BUFmax$
$P_{i,0}$
$P_{i,j}$ where $0 < j < BUFmax$
$P_{i,BUFmax}$
Note : Three kinds of component of $\hat{P}t$

Figure 2.5 State transition probability matrix of Markov chain : $\hat{P}_t$

$$= \quad 1 - P(CP_1 + \cdots + CP_{i+n} > N) \qquad (2.12)$$

- $P_{i,j}$, where $0 < j < BUF_{max}$, $|i - j| \le N$;

$$\begin{aligned} P_{i,j} &= P(BUF_m = j | BUF_{m-1} = i) \\ &= \sum_{n=max\{0,(j-i)\}}^{min\{N,(N+j-i)\}} P(IN = n) P(OUT = i + n - j) \qquad (2.13) \end{aligned}$$

where $max\{0, (j - N)\} \le i \le min\{N + j, BUF_{max}\}$, and $i + n - j \le N$.

$$\begin{aligned} P(OUT = i + n - j) &= P(CP_1 + CP_2 + \cdots + CP_{i+n-j} \le N) \\ &\quad * P(CP_1 + \cdots + CP_{i+n-j} + CP_{i+n-j+1} > N) \\ &\quad \vdots \\ &\quad * P(CP_1 + \cdots + CP_{i+n-j} + \cdots + CP_{min\{N,(i+n)\}} > N) \\ &= P(CP_1 + \cdots + CP_{i+n-j} \le N) * \\ &\quad \prod_{k=i+n-j+1}^{minN,(i+n)} P(CP_1 + CP_2 + \cdots + CP_k > N) \qquad (2.14) \end{aligned}$$

The derivation of $P(OUT = i+n-j)$, here, has additional concerns. First of all, it should not be overflow for the first (i+n-j) multicast cells, i.e. $P(CP_1+CP_2+\cdots+CP_{i+n-j} \leq N)$. However, it must be overflow for any first k (k $\geq$ i+n-j) multicast cells in the buffer, i.e. $\prod_{k=i+n-j+1}^{min\{N,(i+n)\}} P(CP_1+CP_2+\cdots+CP_k > N)$.

- $P_{i,BUF_{max}}$, where $0 \leq i \leq BUF_{max}$, $LOST \geq 0$;

$$\begin{aligned} P_{i,BUF_{max}} &= P(BUF_m = BUF_{max} | BUF_{m-1} = i) \\ &= \sum_{n=(BUF_{max}-i)}^{N} P(IN = n) P_{out} \\ &= \sum_{n=(BUF_{max}-i)}^{N} P(IN = n) \left[\sum_{l=\lfloor \frac{N}{C_{max}} \rfloor}^{n+i-BUF_{max}} P(OUT = l) \right] \end{aligned} \quad (2.15)$$

$$P(OUT = l) = P(CP_1 + \cdots + CP_l \leq N) * \prod_{k=l+1}^{N} P(CP_1 + \cdots + CP_k > N);$$

$$Note,\ P(OUT < \lfloor \frac{N}{C_{max}} \rfloor) = 0.$$

From $P_{i,0}$, $P_{i,j}$ and $P_{i,BUF_{max}}$, we finally obtain the state transition probability matrix $\hat{P}_t$.

2.4.2.2 State Probabilities P_i : P_i is the steady state probability of the Markov Chain, i.e, the probability that the buffer length is i. Let Π be a vector of the stationary state probability, $\Pi = [P_0, P_1, \cdots, P_{BUF_{max}}]$. We have,

$$\Pi = \Pi * \hat{P}_t \ \ and \ \sum_{i=0}^{BUF_{max}} P_i = 1 \quad (2.16)$$

From Eq. 2.16, we get the stationary state probability P_i, where $0 \leq i \leq BUF_{max}$.

2.4.3 Performance Analysis

2.4.3.1 Cell Loss : Due to finite buffer size, cell loss will happen when the shared input buffer is overloaded. In our proposed Markov model, we assume that there are i multicast cells waiting in the buffer at the end of $(m-1)^{th}$ time slot. Cell loss happens when the Markov chain jumps to the state BUF_{max} at the m^{th} time slot.

$$\begin{aligned} P_{lost} &= \sum_{i=0}^{BUF_{max}} P_i P(\text{cell loss / from state i to state } BUF_{max}) \\ &= \sum_{i=x}^{BUF_{max}} P_i P(\text{cell loss / from state i to state } BUF_{max}) \\ &\quad \text{Where } x = BUF_{max} - N + u + 1,\ u = \lfloor \tfrac{N}{C_{max}} \rfloor \end{aligned} \tag{2.17}$$

we have the P_{lost} in detail as :

$$\begin{aligned} P_{lost} &= \sum_{i=BUF_{max}-N+u+1}^{BUF_{max}} P_i \\ &* \left\{ \sum_{n=BUF_{max}-i+u+1}^{N} P(IN=n) \left[\sum_{t=u}^{n+i-BUF_{max}-1} \frac{n+i-BUF_{max}-t}{n} P(OUT=t) \right] \right\} \end{aligned}$$

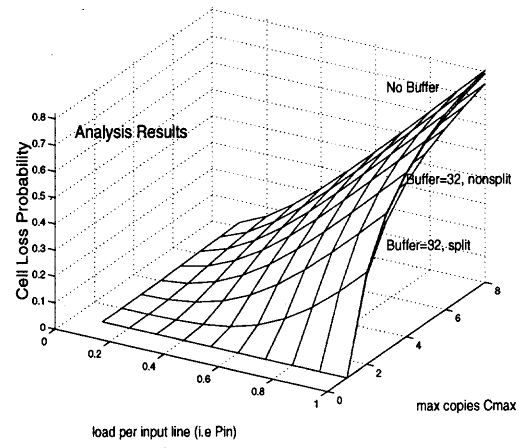
where $\quad P(OUT=t) = P(CP_1 + \cdots + CP_t \leq N) \prod_{k=t+1}^{N} P(CP_1 + \cdots + CP_k > N)$

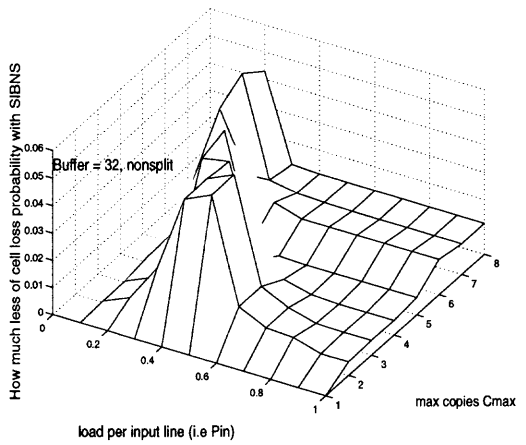
Cell loss is illustrated in Fig 2.6. SIBNS and SIBS copy networks causes less cell loss than NBNS copy network. In fact, SIBNS and SIBS significantly reduce the cell loss in some region where C_{max} and P_{in} jointly give an average load to the copy network. Compared with SIBNS, SIBS has lower cell loss.

2.4.3.2 Throughput : Throughput is evaluated as the number of multicast cells successfully passing the copy network every time slot.

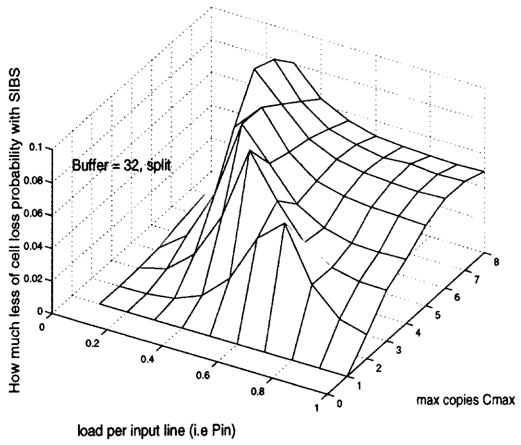
$$Throughput = [1 - P_{lost}]E[\lambda] = [1 - P_{lost}] \left[\sum_{i=0}^{N} iP(IN=i) \right] \tag{2.18}$$

(a) Comparison among NBNS, SIBNS, SIBS



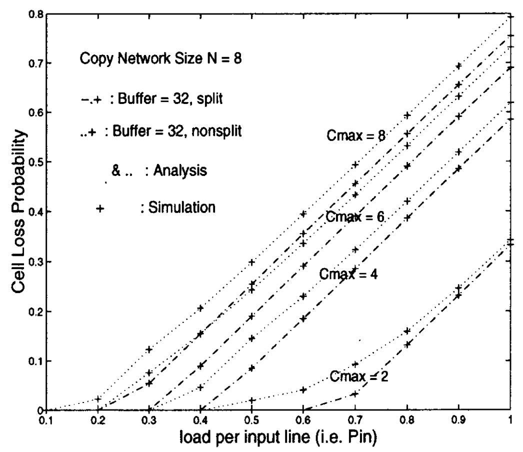


in SIBNS

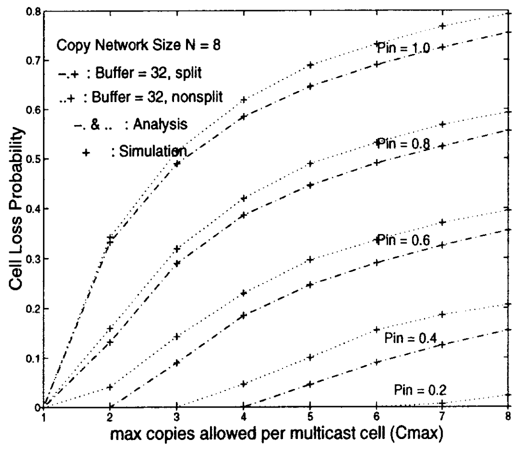


in SIBS

(b) How much lower cell loss resulted from SIBNS , SIBS than NBNS



cell loss probability vs. Pin

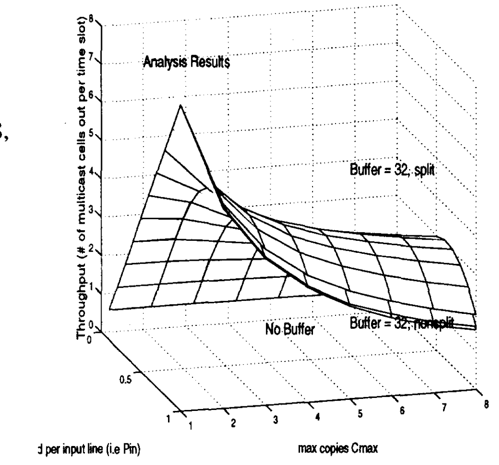


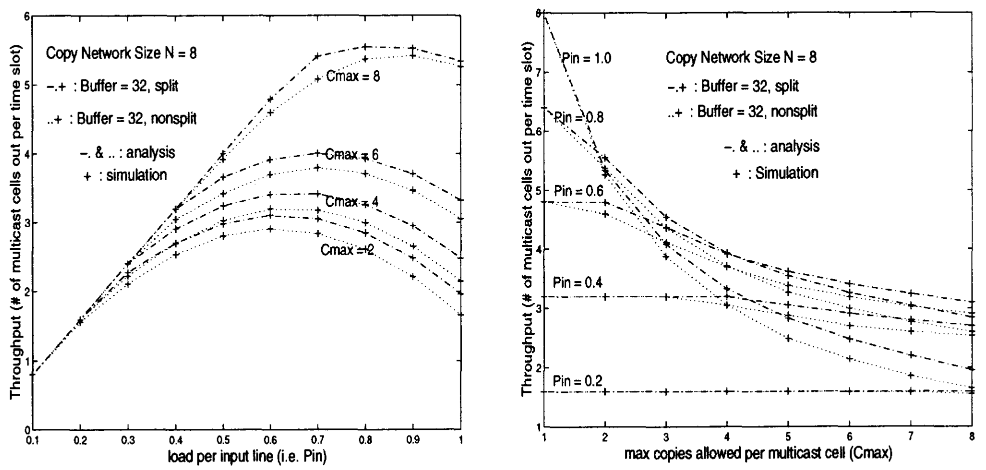
cell loss probability vs. Cmax

(c) Comparison of cell loss between SIBNS and SIBS (with BUFmax = 32)

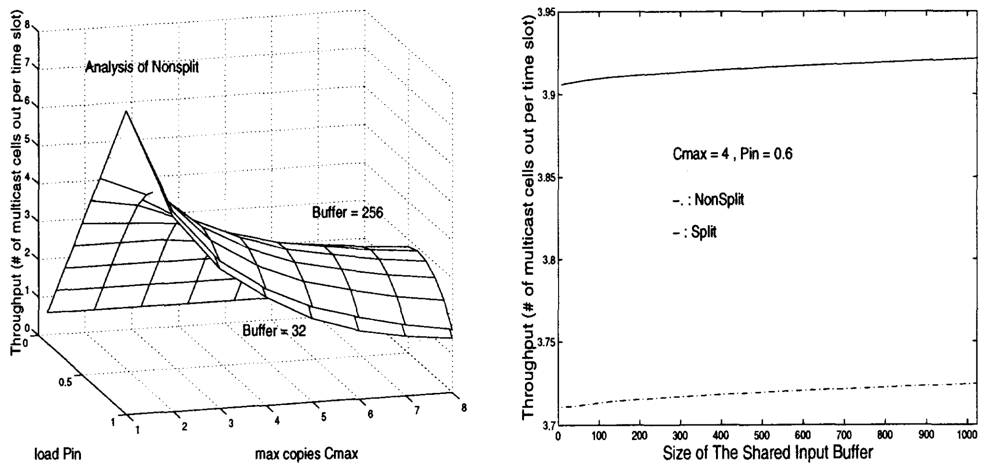
Figure 2.6 Cell loss in three scenarios : NBNS, SIBNS and SIBS

(a) Comparison of throughput among NBNS, SIBNS, and SIBS





(b) comparison of throughput between Nonsplit and Split (with Buffer = 32)



throughput in SIBNS with BUFmax = 32 or 256 comparison between SIBNS and SIBS vs. BUFmax

(c) comparison of throughput with different buffer size

Figure 2.7 throughput in three scenarios : NBNS, SIBNS and SIBS

Shown in Fig 2.7, SIBS achieves higher throughput than SIBNS and NBNS. Higher throughput results from lower cell loss. Throughput is increased with a large buffer size.

2.4.3.3 Cell Delay :

Assume that E(d) is the average delay for a multicast cell waiting in a copy network. E(n) is the average buffer length occupied by the blocked multicast cells. According to the Little's Formula, we have

$$E(n) = E(d) * \lambda'_{eff} \tag{2.19}$$

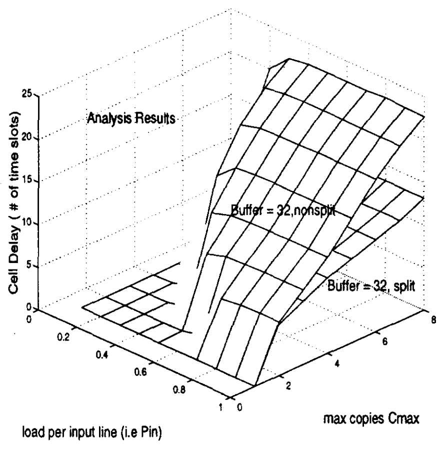
$E(n) = \sum_{i=0}^{BUF_{max}} iP_i$, where P_i is the state probability. λ'_{eff} is an effective average input load which is accepted by the copy network every time slot. Therefore, λ'_{eff} is actually the same as the throughput.

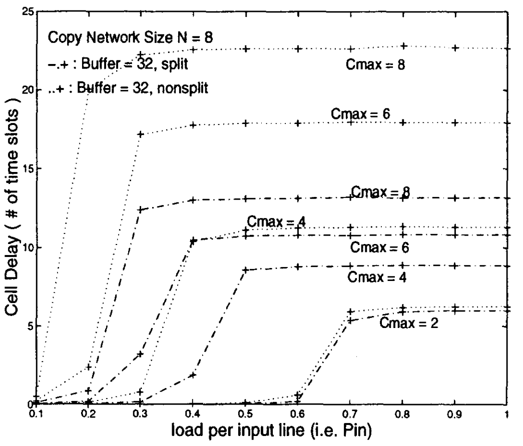
$$\lambda'_{eff} \quad = \quad E[\lambda][1 - P_{lost}] \tag{2.20}$$

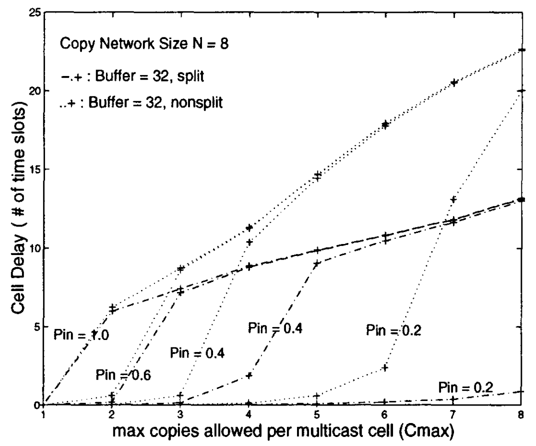
Fig 2.8 shows the performance of cell delay. When the copy load C_{max} or the input load P_{in} becomes heavy, more cells are blocked in the buffer. It causes increased cell delay. SIBS copy network has less cell delay than SIBNS copy network. Larger buffer results in longer cell delay. According to our assumption on C_k, the cell delay increases linearly with buffer size. But, the proposed theoretical approach can be applied to other distribution of C_k.

2.4.4 Validation of the Markov Model for SIBNS and SIBS

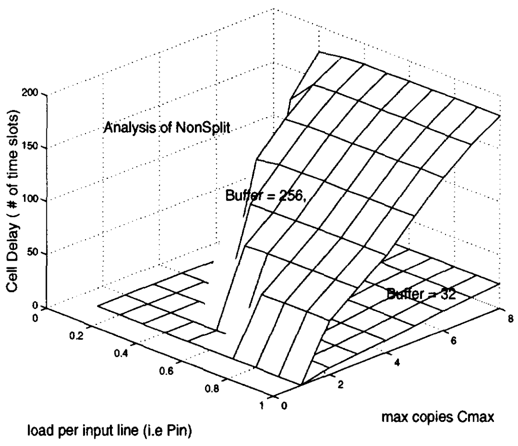
We propose a general Markov Model for the Shared-Input-Buffer copy network with and without splitting algorithm (SIBNS and SIBS). In fact, with different algorithms, the difference between the SIBNS and SIBS exists only in the place where we compute $P(OUT = m)$, which is the probability that m multicast cells successfully pass

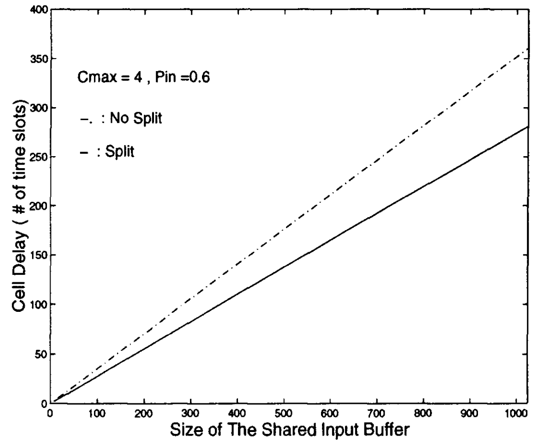






(a) comparison of cell delay between SIBNS and SIBS (with buffer size = 32)





cell delay in SIBNS with BUFmax = 32 or 256 companion of SIBNS and SIBS with BUFmax = 8 to 1024

(b) comparison of cell delay with different buffer size

Figure 2.8 Cell delay in shared-input-buffer copy networks : SIBNS and SIBS

through the copy network. The $P(OUT = m)$ is eventually derived in terms of the overflow probability as :

$$P(OUT = m) = P(CP_1 + CP_2 + \cdots + CP_m \leq N) * \prod_{k=m+1}^{N} P(CP_1 + CP_2 + \cdots + CP_k > N) \tag{2.21}$$

In Eq. 2.21, each one with the form like $P(CP_1 + CP_2 + \cdots + CP_i > N)$ could be obtained by the convolution of CP_k, like :

$$P(CP_1 + CP_2 + \cdots + CP_i > N) = \sum_{j=N+1}^{i*C_{max}} P(CP_1 + CP_2 + \cdots + CP_i = j) \tag{2.22}$$

With *NonSplitting* algorithm, in the *SIBNS* copy network, CP_1 is always the original copies required by the 1^{st} multicast cell. Therefore, we have

$$P(CP_1 + CP_2 + \cdots + CP_i = j) = \sum_{l=1}^{C_{max}} P(CP_1 = l)P(CP_2 + \cdots + CP_i = j - l) \tag{2.23}$$

However, in *SIBS* copy network, the 1^{st} multicast cell is probably splitted. Therefore, the copies required by the 1^{st} cell might be part of the original copy requirements.

$$\begin{aligned} &P(CP_1 + CP_2 + \cdots + CP_i = j) \\ &= \sum_{l=1}^{C_{max}} \left\{ \left[\sum_{m=l}^{C_{max}} P(CP_1^{'} = m)P(CP_1 = l/CP_1^{'} = m) \right] \right. \\ &\left. *P(CP_2 + CP_3 + \cdots + CP_i = j - l) \right\} \end{aligned} \tag{2.24}$$

where $CP_1^{'}$ is the original number of copies required by the 1^{th} multicast cell in the buffer.

In Eq. 2.24, $P(CP_1^{'} = m)P(CP_1 = l/CP_1^{'} = m)$ is the probability that the 1^{st} multicast cell in the buffer currently needs l copies instead of the m copies which is the original requirements. The remaining l copies might be any value which is positive but not larger than m. Therefore,

$$P(CP_1 = l/CP_1' = m) = \frac{1}{m} \quad \text{where } 1 \leq l \leq m \tag{2.25}$$

From the above discussion, the Markov Model we proposed here is generic for both SIBNS and SIBS scenarios. Our Markov Model and corresponding analysis is a novel approach for the performance analysis of the copy network in a multicast ATM switch.

2.5 Conclusion

In this chapter, we analyze the performance of the copy network in a multicast ATM switch under three scenarios : NBNS, SIBNS and SIBS. Theoretical analysis is done for the three cases and compared with simulation results. We proposed a general Markov Model for Shared-Input-Buffer copy network. Our analysis model is shown to be a novel approach for evaluating the performance of copy networks.

The multicast switch evaluated in this chapter is the switch design consisting of a copy network followed by a traditional point-to-point ATM switching network. Switch architecture endures a lot of redundancy due to the usage of copy network and routing network individually. Our switch designs proposed in later chapters will integrate the functions of cell replication and cell routing to reduce the hardware complexity.

CHAPTER 3

SWITCH I: A LARGE SCALE MULTICAST ATM SWITCH USING INPUT AND OUTPUT LINK SHARING

3.1 Introduction

In this chapter, we propose Switch I, a novel switch architecture using input and output link sharing. Switch inputs and switch outputs are grouped into small modules called Input Shared Blocks (ISBs) and Output Shared Blocks (OSBs). Input link sharing resolves output contention and avoids link starvation. Output link sharing eliminates the speedup requirement for the central switch fabric when more than one cell goes to a switch output. Two round robin scheduling algorithms — Individual Virtual Output Queue Round Robin (IVOQ Round Robin), and Grouped Virtual Output Queue Round Robin (GVOQ Round Robin), are presented. Both schemes support group mapping from an ISB to an OSB so that scheduling complexity is significantly reduced. Switch performance is evaluated through simulations. It shows that Switch I can achieve a comparable performance as the OQ switch under uniform traffic. Switch I is scalable due to its modular configuration.

This chapter is organized as follows. In Section 2, we describe the proposed switch architecture in detail. In Section 3, we introduce two cell scheduling algorithms : IVOQ Round Robin and GVOQ Round Robin. Switch performance is evaluated in Section 4. Conclusion is drawn in Section 5.

3.2 Switch Architecture

Fig 3.1 depicts the architecture of Switch I which consists of three major modules : *Input Shared Block (ISB), Output Shared Block (OSB)*, and *ATM Central Switch Fabric (ATMCSF)*. The N Switch inputs and the N switch outputs are respectively grouped into K ISBs and K OSBs, where $K = \frac{N}{m}$. At every ISB-ATMCSF interface, there are M *input links* shared by m related switch inputs. At every ATMCSF-OSB interface, there are M *output links* shared by m grouped switch outputs. In this dissertation, we only consider the case of $m = M$, and the study of $M > m$ which implies a virtual speedup in ATMCSF is the subject of our ongoing work. Applying input link sharing and output link sharing together makes the proposed switch a unique design.

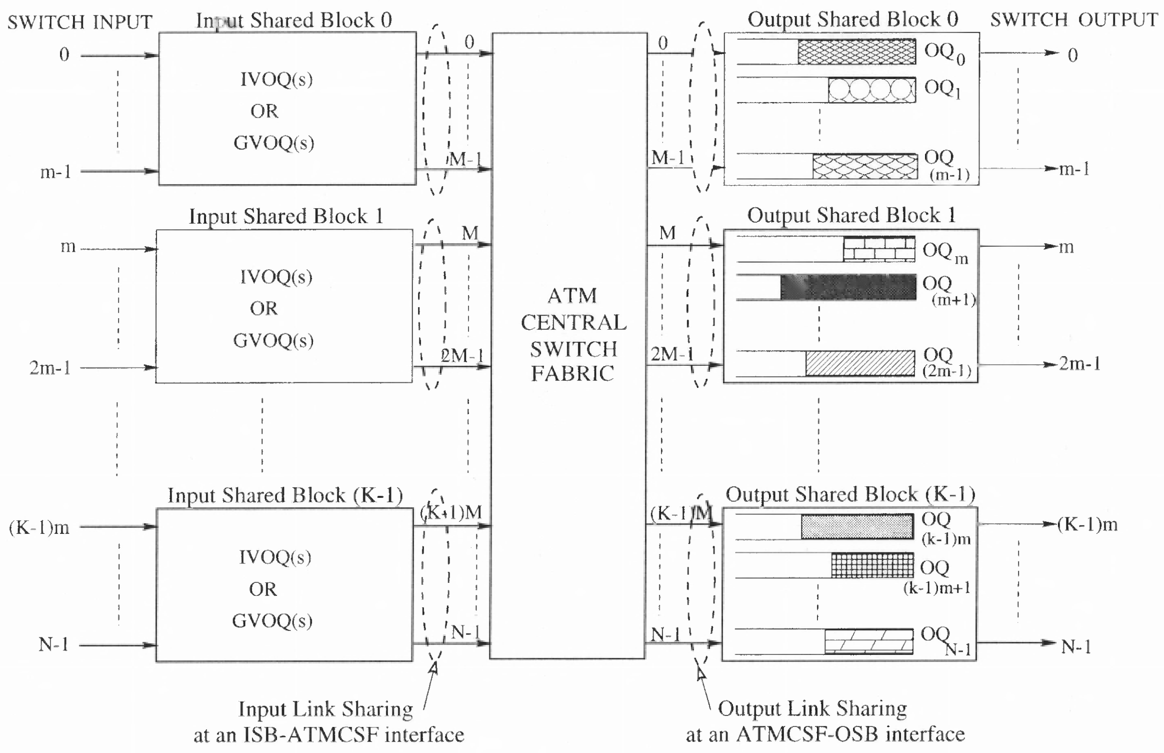
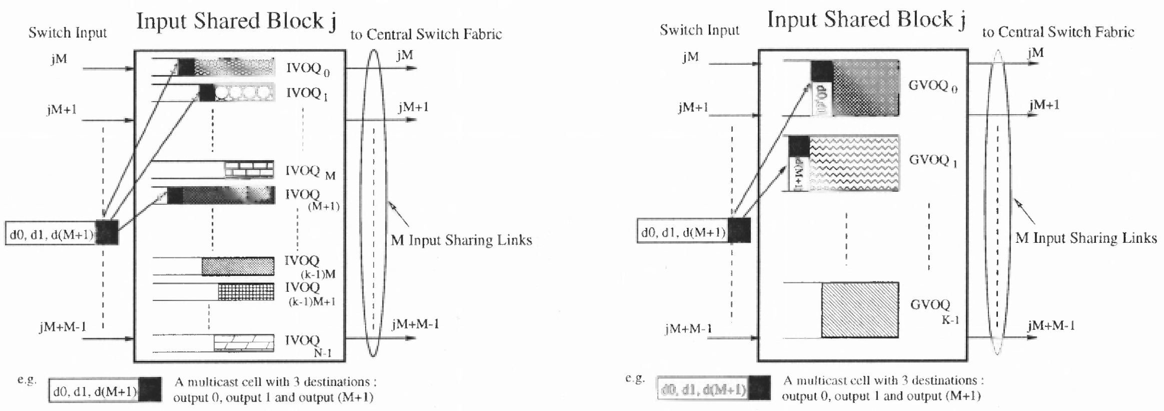


Figure 3.1 Switch I : an NxN switch consists of K ISBs, K OSBs and ATMCSF; $K = \frac{N}{m}$ and $m = M$ in this dissertation. Input link sharing is achieved at every ISB-ATMCSF interface, and output link sharing is achieved at every ATMCSF-OSB interface.

3.2.1 Input Shared Block

An ISB can be a shared memory receiving multicast cells from m $(= M)$ related switch inputs. A multicast cell is saved once in an ISB instead of keeping j identical cell copies (assume, j is the fanout of a multicast cell, $0 \leq j < N$). We investigate two schemes for shared memory management in an ISB (shown in Fig 3.2) — Individual Virtual Output Queue (IVOQ), and Grouped Virtual Output Queue (GVOQ) [54].



(a) Individual Virtual Output Queue (IVOQ)

(b) Grouped Virtual Output Queue (GVOQ)

Figure 3.2 Input Shared Block (ISB) : (a) the j^{th} ISB using IVOQs ; (b) the j^{th} ISB using GVOQs.

IVOQ scheme is shown in Fig 3.2(a). Every ISB keeps N virtual output queues. Each virtual queue is a linked list of the multicast cells going to the same switch output. The physical address to save a multicast cell will be stored into every related linked list. Cell delivery from a virtual output queue is based on FIFO principle. When a multicast cell has all cell copies delivered, the memory address for this cell will be released and available for a new cell.

GVOQ scheme is shown in Fig 3.2(b). Every ISB only maintains K $(= \frac{N}{m})$ grouped virtual output queues. A grouped virtual output queue is a linked list of the multicast cells targeting an OSB. If a multicast cell has more than one destination to an OSB, only a single connection carrying all desired destinations is attached to the related grouped virtual output queue. Hence, a cell delivered from an ISB to ATMCSF may carry multiple destinations, and will be stored into every related output queues when the cell is received by an OSB. Compared with the switch using IVOQs, the switch with GVOQs can forward more cell copies from ISBs to OSBs so that the switch can achieve better performance. GVOQ scheme follows FIFO principle to receive and deliver cells.

Since an ISB-ATMCSF interface has a capacity of M links, an ISB can deliver at most M cells to the central switch fabric in every cell slot. An ISB can send a cell through any of the M shared links. Input link sharing is able to avoid link starvation if some virtual output queue is empty, because other virtual output queues can utilize the idle link to deliver their cells. Input link sharing results in an improved performance.

3.2.2 Output Shared Block

An OSB is a shared memory containing M output queues as shown in Fig 3.3. In every cell slot, each output queue delivers one cell out of the related switch output. An ATMCSF-OSB interface only supports M links, hence, each OSB can accept at most M cells from the central switch fabric in every cell slot. ATMCSF can use any of the M shared links to pass a cell to an OSB. Without output link sharing, if more than one cell goes to the same switch output, either cells are blocked, or it is necessary for the switch fabric to speedup. However, output link sharing is able to avoid both problems.

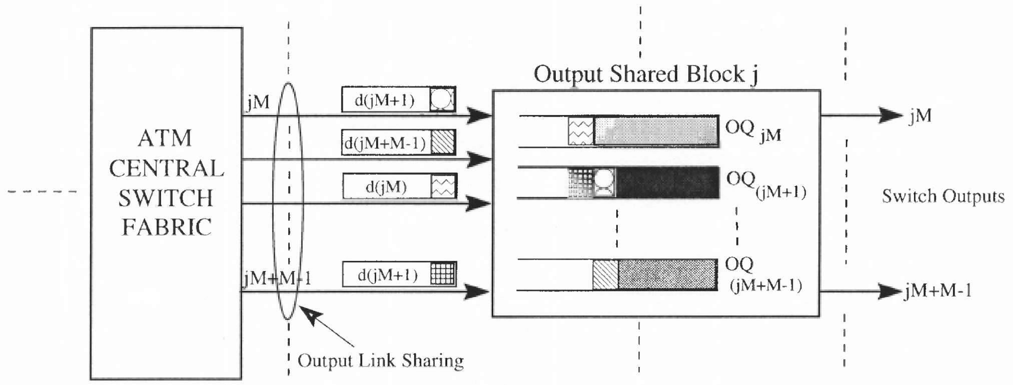


Figure 3.3 Output Shared Block (OSB) with output link sharing (here, $m = M$).

3.2.3 Central Switch Fabric

The central switch fabric (ATMCSF) should keep the same cell sequence for those cell copies which are delivered from an ISB to an OSB. Apart from this, no other restrictions are placed on ATMCSF. It can be any type of switch fabric (for example, Abacus switch [12]), and no speedup is necessary.

3.3 Cell Scheduling

Cell scheduling aims to deliver cells from K ISBs to K OSBs in a fast manner. As shown in Fig 3.4, Switch I utilizes the well-known Round Robin (RR) scheme as the cell scheduling algorithm.

In every cell slot, there is an one-to-one mapping from K ISBs to K OSBs, thus, each ISB is responsible for sending up to M cells to its matched OSB. Round Robin mapping ensures that an ISB has an opportunity to send cells to every OSB in every K cell slots. Fairness among ISBs is guaranteed. Since either IVOQs or GVOQs are employed in every ISB, we propose two scheduling algorithms — *IVOQ Round Robin* and *GVOQ Round Robin*.

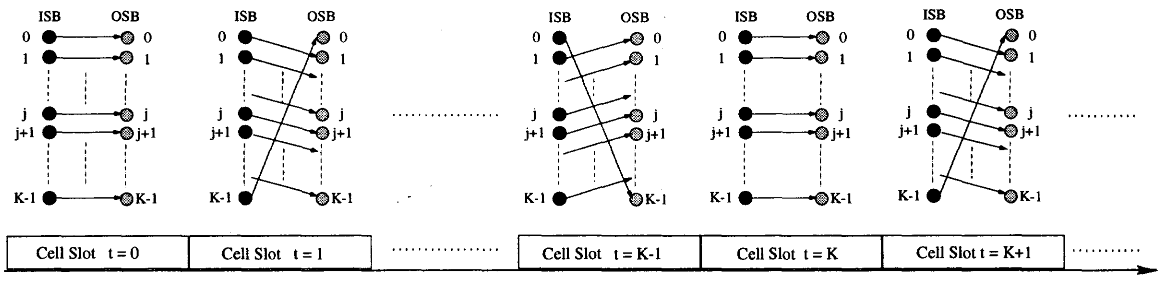


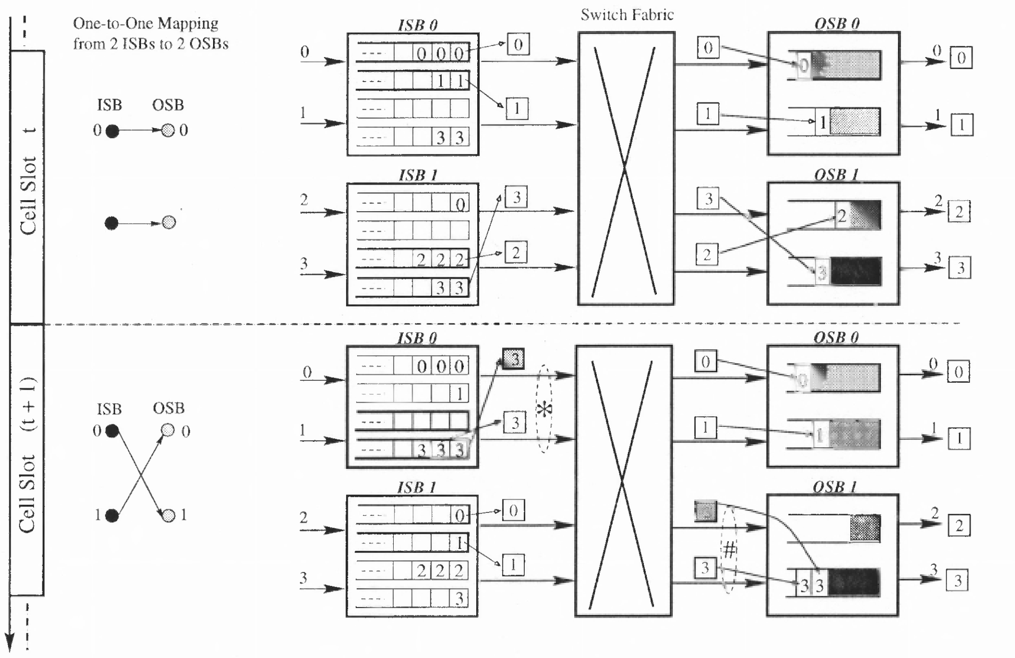
Figure 3.4 Switch I applies round robin (RR) cell scheduling which is based on an one-to-one group mapping from K ISBs to K OSBs.

3.3.1 IVOQ Round Robin

An example of IVOQ RR algorithm is illustrated in Fig 3.5. The basic rules of IVOQ RR algorithm are as follows.

Every ISB divides its N individual virtual output queues into K subgroups. Each subgroup has M virtual output queues which are engaged to a certain OSB. According to the one-to-one mapping in current cell slot, an ISB delivers cells from a subgroup of M virtual output queues to its matched OSB. The HOL cells from the selected M virtual queues are sent to central switch fabric.

If a polled virtual output queue is empty (refer to $*$ in Fig 3.5), other virtual queues in the same subgroup can deliver more than one cell. This is because of using input link sharing which can avoid link starvation. The scheduling complexity depends on how many iterations are needed to select up to M cells from M subgroup GVOQs. Thus, scheduling complexity is in the range of $[O(1),\ O(M)]$. In addition, it may happen that more than one cell goes to a same switch output (refer to $\sharp$ in Fig 3.5). Hence, output link sharing is needed to avoid internal cell loss and to



NOTE : (1) N = 4, K = 2, M = 2 ; (2) 0, 1, 2, 3 indicates a cell's destination to output 0, 1, 2, 3 ;
(3) ✻ : Input Link Sharing relaxes link starvation at an ISB-ATMCSF interface .
(4) # : Ouput Link Sharing eliminates speedup requirement in central switch fabric; ATMCSF needs to keep cell sequence .

Figure 3.5 IVOQ Round Robin in an 4x4 switch ($N = 4, m = M = 2, K = 2$)

eliminate speedup in the central switch fabric. The switch fabric is required to keep the same cell sequence for those cells delivered from an ISB to an OSB.

3.3.2 GVOQ Round Robin

An example of GVOQ RR algorithm is shown in Fig 3.6. GVOQ RR algorithm has several good features.

First, an ISB only maintains K grouped virtual output queues instead of keeping N individual virtual output queues. Each grouped virtual output queue is for an OSB (i.e. for grouped $m = M$ switch outputs). According to the one-to-one mapping from ISBs to OSBs in current cell slot, every ISB only needs to poll a grouped virtual output queue which is for its mapped OSB, then, delivers the first

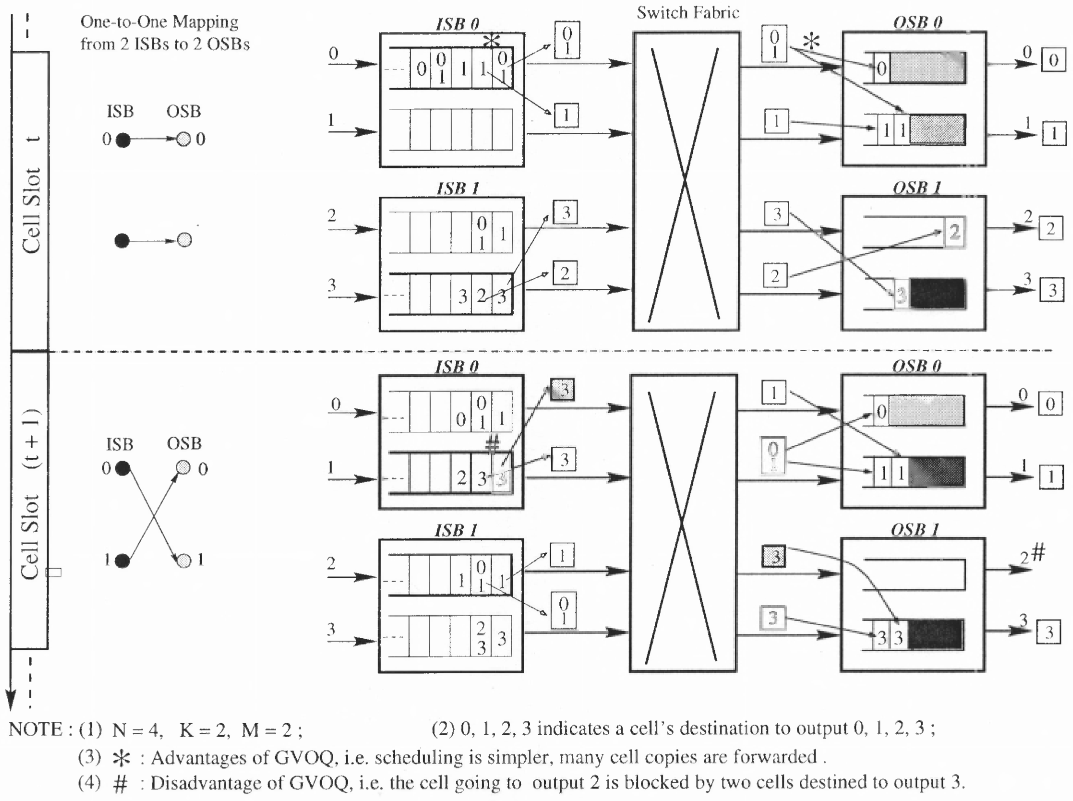

One-to-One Mapping from 2 ISBs to 2 OSBs
ISB
OSB
Cell Slot t
Cell Slot (t + 1)
ISB 0
ISB 1
OSB 0
OSB 1
Switch Fabric
NOTE : (1) N = 4, K = 2, M = 2 ;
(2) 0, 1, 2, 3 indicates a cell's destination to output 0, 1, 2, 3 ;
(3) ✻ : Advantages of GVOQ, i.e. scheduling is simpler, many cell copies are forwarded .
(4) # : Disadvantage of GVOQ, i.e. the cell going to output 2 is blocked by two cells destined to output 3.

Figure 3.6 GVOQ Round Robin in an 4x4 switch ($N = 4, m = M = 2, K = 2$)

M cells, if any, to ATMCSF (refer to $*$ in Fig 3.6). Scheduling complexity is $O(1)$ so that GVOQ RR is simpler than IVOQ RR.

Moreover, using GVOQs in ISBs is able to offer better performance especially under multicast traffic, because a cell delivered from a grouped virtual output queue can carry multiple destinations to the matched OSB (refer to $*$ in Fig 3.6). Compared with IVOQ RR, GVOQ RR algorithm results in a faster cell forwarding from ISBs to OSBs.

But, GVOQ RR algorithm has a flaw that GVOQ RR may block a cell going to an idle switch output while sending more than one cell to a switch output (refer to $\sharp$ in Fig 3.6).

3.4 Performance Evaluation

3.4.1 Traffic Model

We evaluate the switch performance under both uniform and non-uniform traffic. Multicast burst traffic is applied. As shown in Fig 3.7, we use an ON (active)/OFF (idle) model to describe the burst traffic. The back-to-back cells in an ON duration belong to the same VC, i.e. they have the same multiple destinations. Cell destinations are uniformly distributed among N switch outputs.

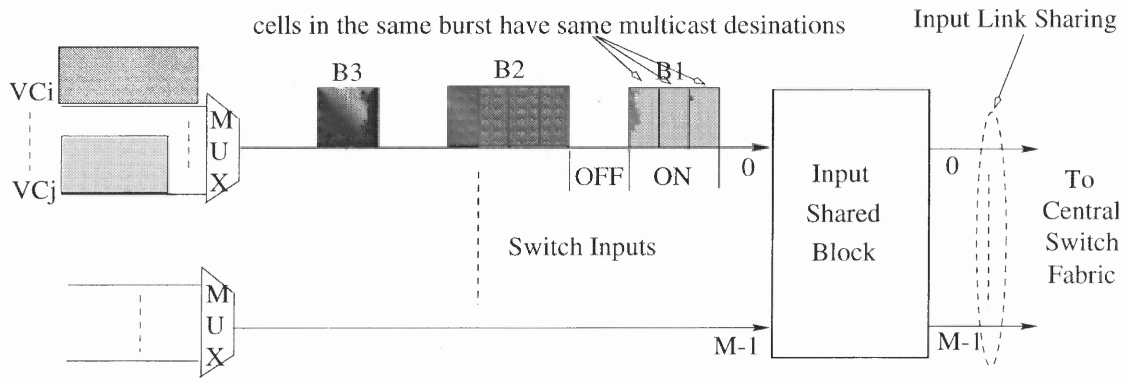


Figure 3.7 Traffic Model : Multicast Bursty Traffic

VBR source is used to generate the ON-OFF burst traffic. The related traffic parameters are : MBS (maximum burst size), LCR (line cell rate), ACR (average cell rate). LCR is about $\frac{155,520,000}{53*8} = 366,793$ (cells/sec) for an OC-3 link. We define F_{out} as the fanout of a multicast cell. F_{out} has a uniform distribution from 1 to C_{max} and average fanout $F = (C_{max} + 1)/2$, where C_{max} is the maximum cell copies allowed for a multicast cell. The input load ρ is defined as $\rho = (ACR * F)/LCR$, $\rho \leq 1$.

3.4.2 Switch Performance

Using OPNET, we simulate an 256x256 switch ($N = 256$) with either 32x32 ISBs/OSBs ($M = 32$, $K = 8$) or 08x08 ISBs/OSBs ($M = 8$, $K = 32$). As a comparison, we also simulate an 256x256 output queued (OQ) switch under same traffic condition. The OQ switch is assumed to have infinite output buffers. Cells arriving at switch inputs will be sent to the related output queues in the same cell slot.

There are two reasons for us to select an OQ switch as a comparison reference : (1) OQ switches proved to maximize throughput and optimize latency under any traffic pattern; (2) in the literature so far, few of existing switches are dedicated for a distributed large scale switch and have been evaluated under any traffic condition. Therefore, we believe that it is fair and effective to compare our designs with an OQ switch under same traffic patterns.

The performance of Switch I under **uniform** traffic is illustrated in Table 3.1 and Table 3.2. For uniform traffic, the input load to an ISB uniformly targets N switch outputs. We apply both unicast traffic and multicast traffic. In **unicast traffic**, every arriving cell only carries a single destination. But, in **multicast traffic**, a coming cell may have multiple destinations. Cells' destinations are uniformly distributed among N switch outputs.

Through simulation, we would like to : (1) investigate the impact of ISB/OSB size on switch performance; (2) evaluate IVOQ RR and GVOQ RR algorithms; (3) compare Switch I with the OQ switch.

Table 3.1 shows the switch performance under **uniform unicast traffic**. Fig 3.8 $\sim$ Fig 3.11 depict the performance on throughput, end-to-end cell delay (i.e. D_{E-to-E}), cell delay in ISB (i.e. D_{ISB}), occupancy of OSB (i.e. S_{OSB}) respectively.

Table 3.1 Switch I : switch performance under uniform unicast traffic with different input load ρ. The observed performance statistics are : (1) throughput; (2) average end-to-end cell delay and delay jitter (D_{E-to-E}, (Min, Max)); (3) average cell delay in ISB and delay jitter (D_{ISB}, (Min, Max)); (4) average occupancy of OSB (S_{OSB} and (Min, Max)).

		Uniform, Unicast, Burst Traffic (F = 1 , Fout = 1, MBS = 32)				
		OQ Switch ***(256x256 Switch)***	***Switch I***			
			256x256 Switch, 08x08 ISBs/OSBs (N = 256, m = M = 8, K = 32)		***256x256 Switch, 32x32 ISBs/OSBs*** (N = 256, m = M = 32, K = 8)	
			IVOQ RR	GVOQ RR	IVOQ RR	GVOQ RR
ρ = **0.99**	Throughput	0.99	0.96149	0.95868	0.98578	0.98463
	D_{E-to-E} (Min, Max)	48 (1.0 , 197)	102 (6.8 , 309)	116 (7.0 , 322)	61 (5.6 , 228)	66 (6,8 , 242)
	D_{ISB} (Min, Max)	N/A	78.0 (1.0 , 148)	86.0 (1.0 , 152)	22.8 (1.0 , 123)	23.0 (1.0 , 91)
	S_{OSB} (Min, Max)	N/A	133 (66 , 230)	196 (109 , 295)	1256 (1082 , 1418)	1370 (1152 , 1560)
ρ = **0.90**	Throughput	0.90	0.88956	0.88886	0.89873	0.89866
	D_{E-to-E} (Min, Max)	11.4 (1.0 , 105)	57.1 (6.0 , 166)	61.4 (6.5 , 187)	21.0 (6.0 , 105)	23.2 (6.4 , 116)
	D_{ISB} (Min, Max)	N/A	44.2 (1.0 , 112)	44 (1.0 , 114)	5.8 (1.0 , 38.6)	5.8 (1.0 , 24)
	S_{OSB} (Min, Max)	N/A	50.6 (22.5, 104)	71 (34.6 , 134)	317 (238, 408)	331 (248. 423)
ρ = **0.70**	Throughput	0.70	0.69631	0.69623	0.69928	0.69927
	D_{E-to-E} (Min, Max)	4.0 (1.0 , 46.5)	29.3 (6.0 , 178)	31.4 (6.0 , 144)	9.2 (4.0 , 60)	9.2 (6.0 , 62)
	D_{ISB} (Min, Max)	N/A	22.5 (1.0 , 162)	22.2 (1.0 , 126)	4.6 (1.0 , 18.4)	4.6 (1.0 , 14.0)
	S_{OSB} (Min, Max)	N/A	22.6 (7.2 , 40.0)	24.0 (8, 45)	99 (72 . 131)	100 (71 , 133)
ρ = **0.50**	Throughput	0.50	0.49830	0.49829	0.49957	0.49957
	D_{E-to-E} (Min, Max)	2.4 (1.0 , 26)	14.2 (5.0 , 66)	14.0 (5.2 , 64)	4.0 (1.0 , 52)	4.0 (2.4 , 48.2)
	D_{ISB} (Min, Max)	N/A	11.4 (1.0 , 48)	10.0 (1.0 , 44)	2.5 (1.0, 14.8)	2.5 (1.7 , 13)
	S_{OSB} (Min, Max)	N/A	11.0 (2.2 , 23)	12.0 (2.4 , 25)	45 (30 , 64)	45 (30 , 64)

It is observed that, larger size of ISBs/OSBs results in better performance. For example, if input load ρ is 99%, Switch I with 32x32 ISB(s) obtains approximately 98.5% throughput, while the switch with 8x8 ISB(s) endures 3% less throughput. The reason is that, an 32x32 ISB receives cells from 32 switch inputs so that the preserved cells in an ISB are more varied in terms of the destination requirements. Hence, larger ISB is more likely to provide a saturated input load to the central switch fabric (i.e. keep every input line of ATMCSF busy).

Comparison also shows that, IVOQ RR algorithm exceeds GVOQ RR algorithm on switch performance. The reason is that, IVOQ RR can send M cells to different switch outputs of its mapped OSB, if none of the M related virtual queues is empty. But, since GVOQ RR simply schedules the first M cells from a grouped virtual output queue to ATMCSF, more than one cell may go to the same switch output. Hence, IVOQ RR achieves higher throughput than GVOQ RR. Moreover, under unicast traffic, GVOQ RR losses its unique merit to deliver multicast cells to ATMCSF. But, it is worth to notice that IVOQ RR algorithm and GVOQ RR algorithm yield a very similar performance when input load ρ is reduced. When traffic load is light, an ISB usually can send arriving cells to the mapped OSB very quickly so that few cells are blocked in ISBs.

Compared to the OQ switch, Switch I exhibits a promising performance under the uniform unicast traffic. Switch I causes less than 4.0% throughput degradation under heavy traffic load, but achieves a very similar throughput when traffic load decreases. The OQ switch defeats Switch I due to the reason that the OQ switch is work-conserving [1] at every time slot, but switch I is not. Switch I endures longer cell delay (i.e. D_{E-to-E}) than the OQ switch especially under heavy input load. Longer

[1]A switch is work-conserving, if in every cell slot, a switch output is not idle as long as there are cells going to that output port.

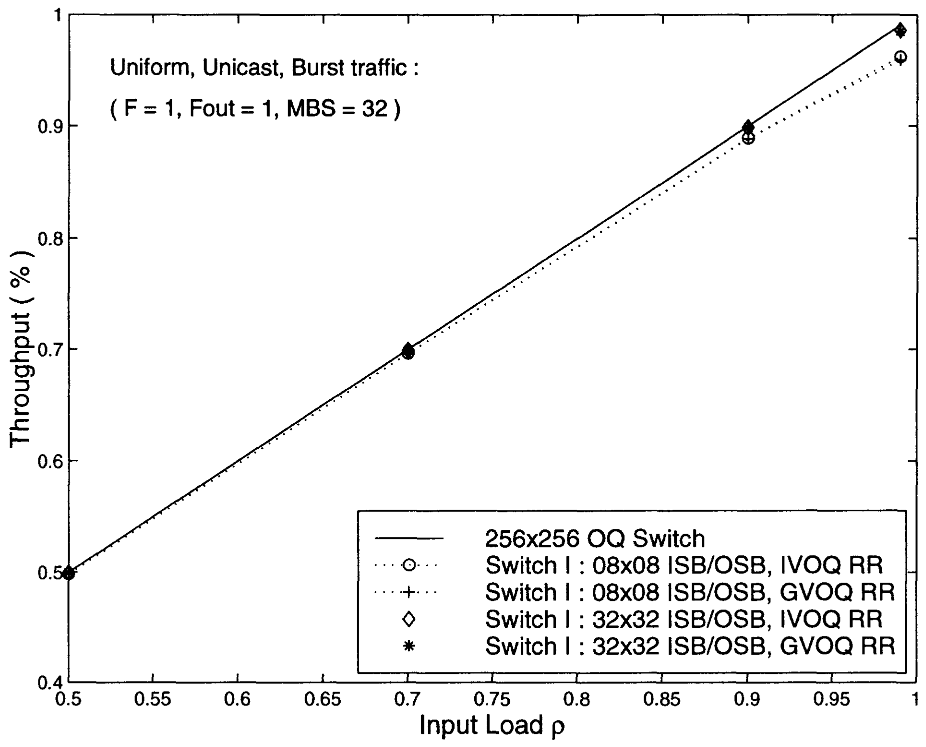


Figure 3.8 Switch I : throughput under uniform unicast traffic.

cell delay is due to lower throughput. We also measured the queueing latency in ISBs (i.e. D_{ISB}) and the occupancy of OSBs (i.e. S_{OSB}) for Switch I. When input load increases, both D_{ISB} and S_{OSB} become longer. Since the central switch fabric is assumed to deliver cells from input shared links to output shared links in the same cell slot, hence, end-to-end cell delay is resulted from two parts : *cell delay in ISBs* and *cell delay in OSBs*; i.e. we have

$$D_{E\text{-}to\text{-}E} = D_{ISB} + D_{OSB} \approx D_{ISB} + \frac{S_{OSB}}{M}; \tag{3.1}$$

where $\frac{S_{OSB}}{M}$ is the average length of the output queue in OSBs, assuming that $m = M$. Thus, this ratio approximates D_{OSB}, i.e. the average *cell delay in OSBs*.

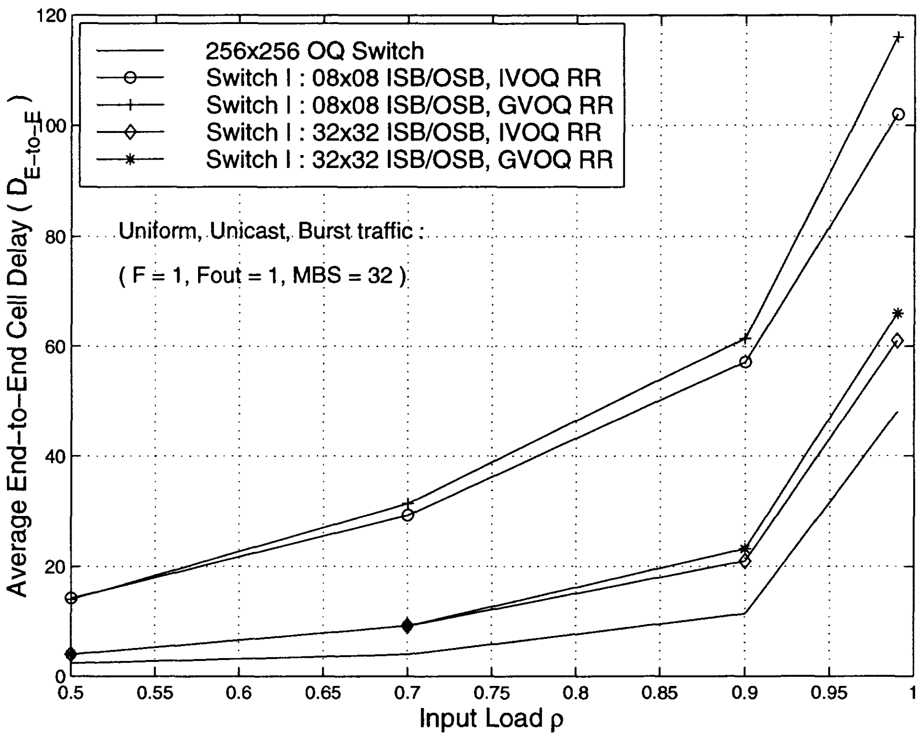


Figure 3.9 Switch I : average end-to-end cell delay (D_{E-to-E}) under uniform unicast traffic.

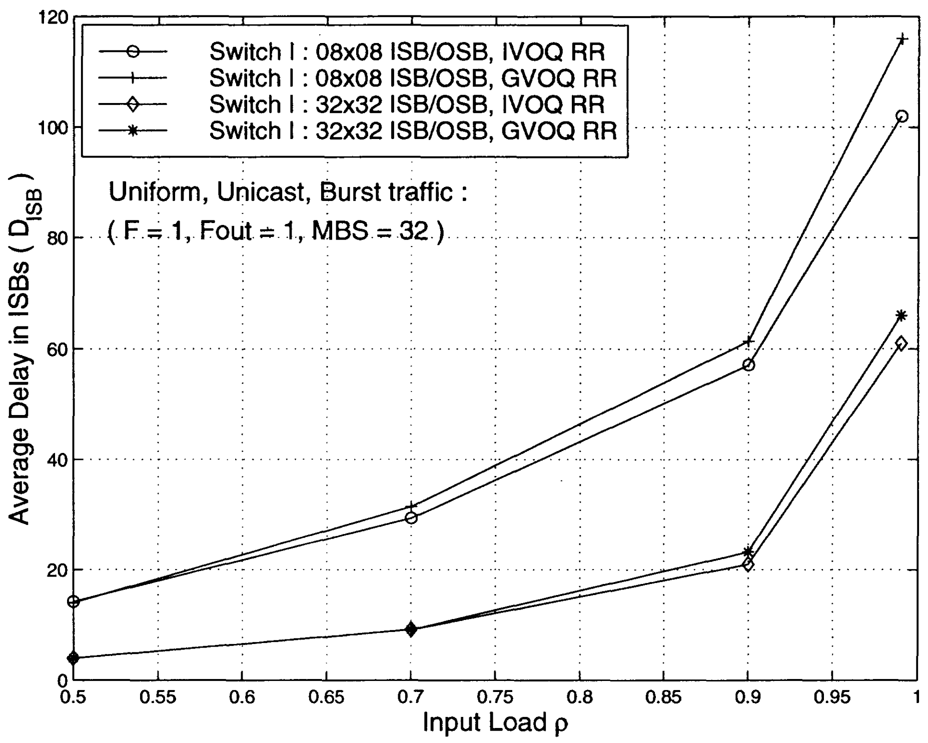


Figure 3.10 Switch I : average cell delay in ISB (D_{ISB}) under uniform unicast traffic.

Switch performance under **uniform multicast traffic** is illustrated in Table 3.2. As we discussed previously, Switch I with 32x32 ISB/OSB achieves better performance than that using 8x8 ISB/OSB. But, GVOQ RR outperforms IVOQ RR for multicast traffic. It is due to the fact that GVOQ RR can forward splitted *multicast* cells to ATMCSF, but IVOQ RR only delivers *unicast* cells to ATMCSF. Hence, GVOQ RR provides a faster cell forwarding. Compared with IVOQ RR algorithm, GVOQ RR can be claimed as a cost-effective algorithm in the sense that GVOQ RR can obtain a similar or better performance than IVOQ RR while gaining a lot by reducing the complexity on memory management and cell scheduling.

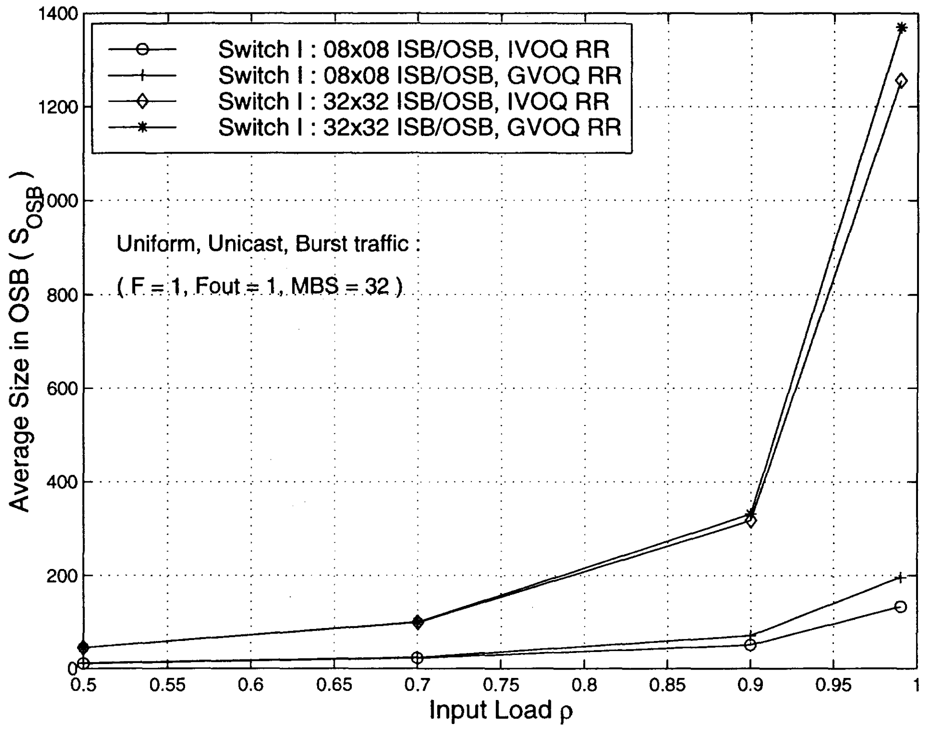


Figure 3.11 Switch I : average size of OSB (S_{OSB}) under uniform unicast traffic.

The switch performance observed from Fig 3.12 ~ Fig 3.15 show that Switch I is able to pursue a comparable performance to the OQ switch under uniform multicast traffic. In addition, larger size of ISBs/OSBs results in better performance. But, to mimic the OQ switch, small latency in ISBs is expected. If $D_{ISB} = 0$, it implies that the proposed switch can pass cells to OSBs as fast as the OQ switch, however, our switch does not need any speedup in central switch fabric. This is our essential goal.

Up to now, Switch I is demonstrated to support uniform traffic. However, Switch I has a weakness to support **non-uniform** traffic in which cells accumulated in an ISB may prefer to go to some switch outputs but do not go to other outputs. A typical example of the non-uniform traffic is so called "1 ISB → 1 OSB hotspot

Table 3.2 Switch I : switch performance under uniform multicast traffic with different input load ρ. The observed performance statistics are : (1) throughput; (2) average end-to-end cell delay and delay jitter (D_{E-to-E}, (Min, Max)); (3) average cell delay in ISB and delay jitter (D_{ISB}, (Min, Max)); (4) average occupancy of OSB (S_{OSB} and (Min, Max)).

		Uniform, Multicast, Burst Traffic (F = 4 , Cmax = 8, MBS = 32)				
			Switch I			
		OQ Switch *(256x256 Switch)*	***256x256 Switch, 08x08 ISBs/OSBs*** (N = 256, m = M = 32, K = 8)		***256x256 Switch, 32x32 ISBs/OSBs*** (N = 256, m = M = 8, K = 32)	
			IVOQ RR	GVOQ RR	IVOQ RR	GVOQ RR
ρ = **0.99**	Throughput	0.99	0.96689	0.97540	0.98662	0.98898
	$\mathbf{D}_{\text{E-to-E}}$ (Min, Max)	24.6 (1.0 , 153)	100 (5.8 , 652)	81 (5.8 , 488)	39.3 (5.8 , 231)	34.3 (5.7 , 203)
	$\mathbf{D}_{\text{ISB}}$ (Min, Max)	N/A	77.5 (1.0 , 520)	55 (1.0 , 252)	12.8 (1.0 , 122)	7.6 (1.0 , 18.0)
	$\mathbf{S}_{\text{OSB}}$ (Min, Max)	N/A	98 (30.6 , 147)	154 (73, 271)	622 (471 , 796)	764 (586 , 965)
ρ = **0.90**	Throughput	0.90	0.88989	0.89204	0.89865	0.89917
	$\mathbf{D}_{\text{E-to-E}}$ (Min, Max)	9.5 (1.0 , 95)	64 (5.0 , 448)	57 (5.2 , 307)	21.5 (5.6 , 169)	17.2 (5.6 , 146)
	$\mathbf{D}_{\text{ISB}}$ (Min, Max)	N/A	58 (1.0 , 368)	44 (1.0 , 243)	8.5 (1.0 , 76)	5.9 (1.0 , 17)
	$\mathbf{S}_{\text{OSB}}$ (Min, Max)	N/A	37 (17 , 75)	59 (24 , 115)	249 (180 , 312)	364 (201 , 483)
ρ = **0.70**	Throughput	0.70	0.69654	0.69689	0.69932	0.69941
	$\mathbf{D}_{\text{E-to-E}}$ (Min, Max)	3.6 (1.0 , 38)	36 (5.5 , 227)	28.8 (5.0 , 136)	10.8 (2.2 , 94)	8.1 (2.0 , 76)
	$\mathbf{D}_{\text{ISB}}$ (Min, Max)	N/A	25 (1.0 , 198)	22 (1.0 , 113)	5.6 (1.0 , 72.5)	4.5 (1.0 , 12.8)
	$\mathbf{S}_{\text{OSB}}$ (Min, Max)	N/A	16 (5, 31)	21 (8 , 39)	82 (58.0 , 110.0)	86.5 (60 , 117)
ρ = **0.50**	Throughput	0.50	0.49868	0.49874	0.49967	0.49967
	$\mathbf{D}_{\text{E-to-E}}$ (Min, Max)	2.0 (1.0 , 22)	32 (5.0 , 157)	24 (5.0 , 86)	5.8 (1.0 , 66)	3.6 (1.0 , 47)
	$\mathbf{D}_{\text{ISB}}$ (Min, Max)	N/A	16.5 (1.0 , 147)	15.5 (1.0 , 76)	4.5 (1.0 , 25)	2.1 (1.0 , 11)
	$\mathbf{S}_{\text{OSB}}$ (Min, Max)	N/A	8.4 (1.0 , 19)	9.4 (1.0 , 22)	38 (22 , 55)	38.3 (22.3, 57)

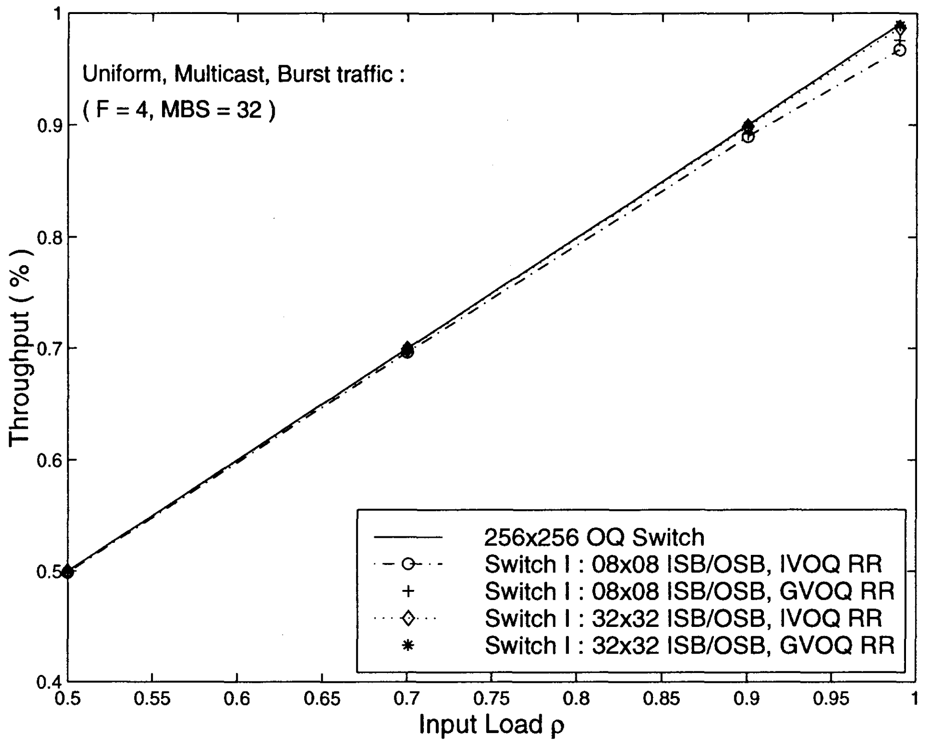


Figure 3.12 Switch I : throughput under uniform multicast traffic.

traffic", i.e. arriving cells to the i^{th} ISB only target the switch outputs belonged to the i^{th} OSB. Fig 3.16 shows the throughput performance of Switch I under " 1 ISB $\rightarrow$ 1 OSB hotspot traffic". It is observed that Switch I suffers a significant performance degradation when compared with the OQ switch. When input load is 99%, Switch I only yields 50% throughput under multicast traffic, and approximately 12.9% throughput under unicast traffic. The reason for that is, Round Robin cell scheduling only allows an ISB to deliver cells to its mapped OSB in a cell slot. If an ISB has no cells to its related OSB, other ISBs do not have authority to deliver cells to the starved OSB. How to solve this problem motivates new solutions which will be presented in the later chapters.

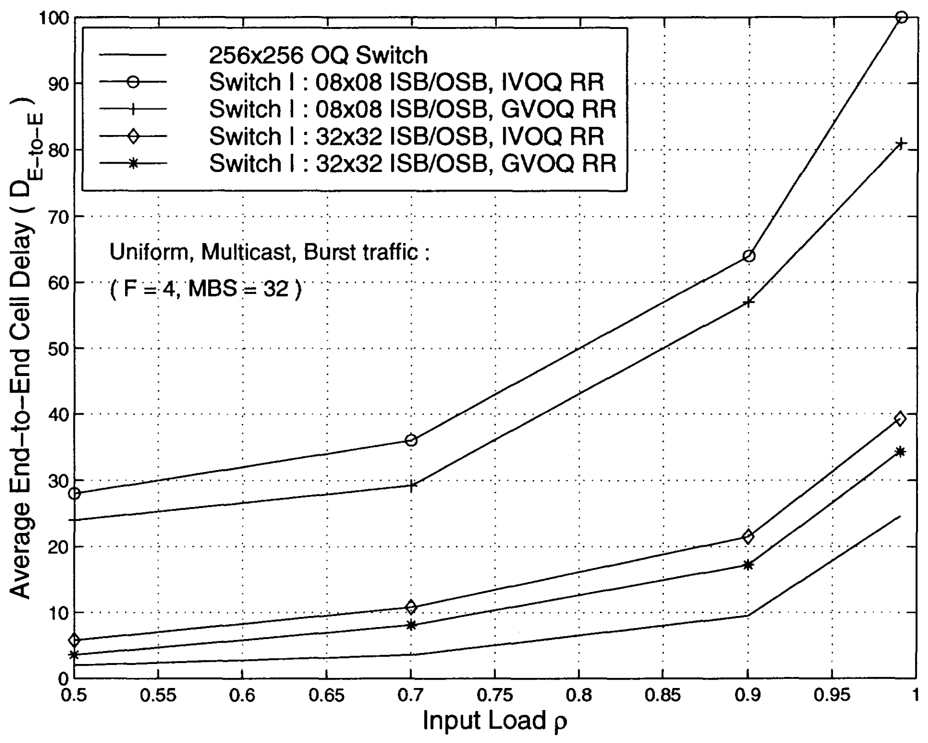


Figure 3.13 Switch I : average end-to-end cell delay (D_{E-to-E}) under uniform multicast traffic.

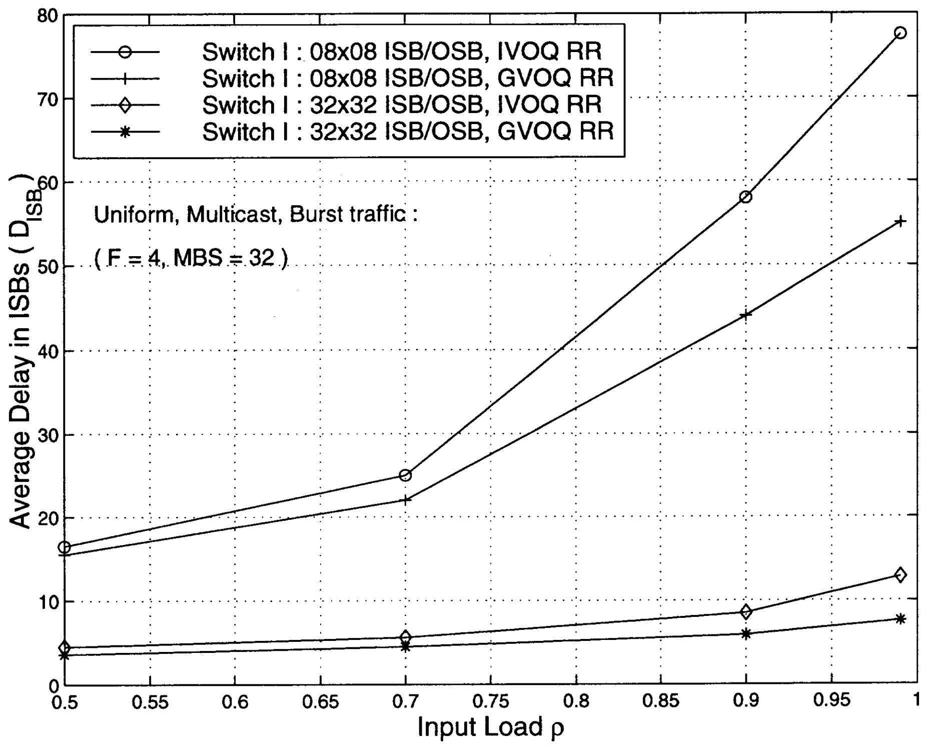


Figure 3.14 Switch I : average cell delay in ISB (D_{ISB}) under uniform multicast traffic.

3.5 Conclusion

In this chapter, we proposed Switch I, a novel switch architecture using input and output link sharing. The merits of Switch I are the modular switch architecture and the distributed cell scheduling. Compared to the OQ switch, Switch I eliminates speedup requirement for the central switch fabric. Moreover, RR scheduling algorithms resolve output contention in a distributed manner and guarantee fairness for switch inputs. Scheduling complexity of IVOQ RR algorithm is at most $O(M)$, while it is $O(1)$ for GVOQ RR algorithm. Compared with the centralized schedulers with a complexity of at least $O(N^{2.5})$ proposed for IQ switches in [30, 31, 32, 35],

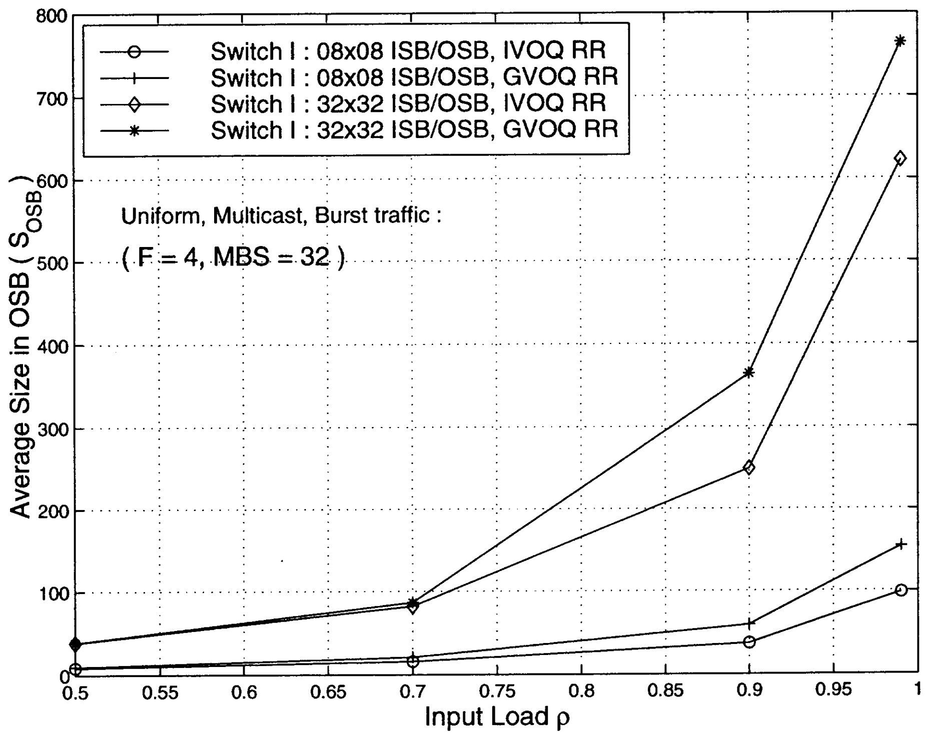


Figure 3.15 Switch I : average size of OSB (S_{OSB}) under uniform multicast traffic.

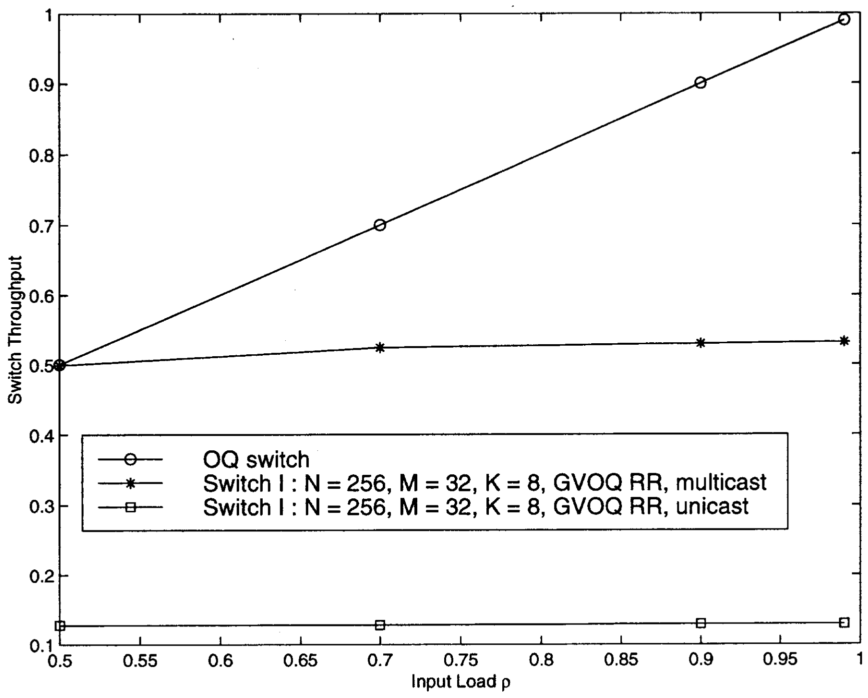


Figure 3.16 Performance of Switch I under '1 ISB $\rightarrow$ 1 OSB hotspot non-uniform traffic'.

cell scheduling in our design is much simpler. Hence, Switch I shows good features to be a scalable design.

But, Switch I has a drawback to support non-uniform traffic in which cells injected in an ISB are not uniformly target N switch outputs. Since RR scheduling algorithm only allows an ISB to deliver cells to its matched OSB in every cell slot, if an ISB does not have cells to go to the polled OSB, other ISBs do not have authority to deliver cells to the idle OSB. Starvation of OSB(s) will cause performance degradation. To resolve this problem, we will present a modified switch design in chapter 4.

CHAPTER 4

SWITCH II: A MODIFIED SWITCH DESIGN USING LINK SHARING AND PRIORITIZED LINK RESERVATION

4.1 Introduction

In previous chapter, we proposed Switch I as a basic switch design using input and output link sharing. Switch I is demonstrated to be able to support uniform traffic, but it suffers a disability to provide high performance under non-uniform traffic [1]. To overcome the drawback of Switch I, in this chapter, we present Switch II which is a modified switch design using link sharing and prioritized link reservation.

In Switch II, ISBs are connected through a token ring. Cell delivery in a cell slot is based on link reservation in every ISB. We propose a *round robin prioritized output link reservation (RR+POLR)* algorithm to resolve contentions on input shared links and output shared links. Basically, Switch II still applies RR scheme to obtain an one-to-one mapping from K ISBs to K OSBs in every cell slot. An ISB has the highest priority to reserve as many links as possible to its mapped OSB. If an ISB can not fully occupy the M links to its mapped OSB, the ISB will issue a token to inform other ISBs that there are idle links remained at the specific ATMCSF-OSB interface. Therefore, other ISBs can reserve and utilize the available links to transfer their cells to the OSB. Starvation of OSBs is alleviated. Switch II can pursue an improved performance especially under non-uniform traffic.

4.2 Switch Architecture

Fig 4.1 exhibits the architecture of Switch II consisting of K ISBs, ATMCSF, K OSBs, and a token ring. Functions of ISB, OSB and ATMCSF are the same as what

[1] Non-Uniform traffic is usually characterized by "hot spot" phenomenon (i.e. cells accumulated in an ISB may prefer some switch outputs but seldom go to other outputs).

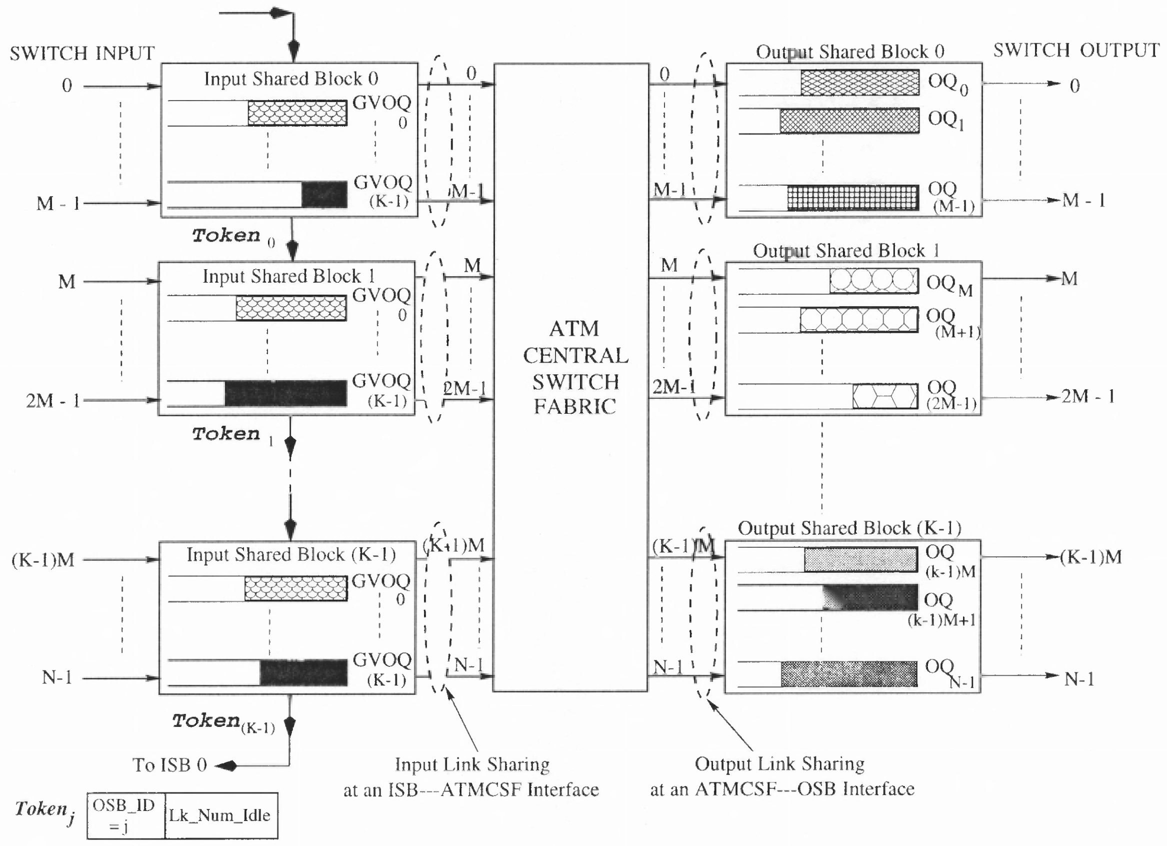


Figure 4.1 Switch II : an NxN switch consists of K ISBs, K OSBs, ATMCSF, and a token ring; $K = \frac{N}{M}$. Cell delivery in a cell slot is based on link reservation. We propose a round robin prioritized output link reservation (RR+POLR) algorithm.

we presented in Switch I. We concluded in chapter 3 that GVOQ is a cost-efficient scheme compared to IVOQ. Hence, both Switch II and Switch III will apply GVOQs in each ISB.

ISBs are connected by a token ring on which K tokens circulate in a round robin manner. Each token is related to a specific OSB, for example, $Token_j$ is engaged to the j^{th} OSB ($0 \leq j < K$). As shown in Fig 4.1, a token has two fields : (1) "OSB_ID" is the identification of an OSB; (2) "Num_Lk_Idle" records the number of available links at the identified ATMCSF-OSB interface, Num_Lk_Idle ≥ 0.

4.3 Cell Scheduling

4.3.1 Cell Delivery

Cell delivery is based on link reservation. Every ISB should make link reservation in advance in order to obtain the desired links at the targeted ATMCSF-OSB interfaces.

Every ISB has a *link reservation vector* and a *queue occupancy vector*. We use LK_RSV^i and Q^i to represent the two vectors in the i^{th} ISB $(0 \leq i,\ j\ < K)$:

- LK_RSV^i : **link reservation vector in the i^{th} ISB.**

  LK_RSV^i $= [r_0^i\ ,\ r_1^i\ ,\ \cdots,\ r_{(K-1)}^i]$, where r_j^i indicates how many links at the ATMCSF-OSB j interface are reserved by the i^{th} ISB, $0 \leq r_j^i \leq M$.

- Q^i : **queue occupancy vector in the i^{th} ISB.**

  Q^i $= [q_0^i\ ,\ q_1^i\ ,\ \cdots,\ q_{(K-1)}^i]$, where q_j^i is the queue length (# of cells) of the j^{th} GVOQ in the i^{th} ISB, $q_j^i \geq 0$.

Link reservation vector is renewed in every cell slot. Each ISB resets its link reservation vector at the beginning of a cell slot, then starts reserving output shared links according to a *Round Robin Prioritized Output Link Reservation (RR+POLR)* algorithm. When a cell slot ends, every ISB delivers cells to the central switch fabric according to its current link reservation. For example, if LK_RSV^i is [2, 0, $\cdots$, 4] in current cell slot, the i^{th} ISB will send two cells to OSB 0 and four cells to OSB (K-1), but no cells are scheduled to other OSBs.

4.3.2 Link Reservation : RR+POLR Algorithm

Definition 1 : *Link Reservation Rule.*

(1) $\sum_{j=0}^{K-1} r_j^i \leq M$, i.e. the total links reserved by the i^{th} ISB can not exceed M which is the maximum number of links at an ISB-ATMCSF interface;

(2) $\sum_{i=0}^{K-1} r_j^i \leq M$, i.e. the total links reserved by all ISBs to the j^{th} OSB can not exceed M which is the maximum number of links at the ATMCSF-OSB interface.

Definition 2 : *Link Reservation Slot, i.e. Rsv_Slot.*

Rsv_Slot is defined as a small time interval during which an ISB receives a token and makes link reservation to the identified OSB. Rsv_Slot is independent from Cell_Slot, usually, Rsv_Slot $\ll$ Cell_Slot. When a cell slot is due, every ISB delivers cells to ATMCSF according to its current link reservation vector.

Table 4.1 illustrates RR+POLR algorithm which is performed in every cell slot. Each ISB resets its link reservation vector at the beginning of a cell slot. Switch II adopts *Round Robin (RR)* scheme proposed in Switch I (Fig 3.4) to obtain an one-to-one mapping from K ISBs to K OSBs in every cell slot. The OSB mapped to an ISB is called the ISB's *Master-OSB*. In the 1^{st} Rsv_Slot of a cell slot, an ISB has the highest priority to reserve as many links as possible to its Master-OSB. If an ISB does not fully occupy the M links to its Master-OSB, a token carrying available links for this OSB will be issued by the ISB. Tokens pass through ISBs one by one in a round robin manner. When an ISB receives a token carrying available links, the ISB can reserve as many links as possible to the identified OSB.

Fig 4.2 depicts the operations of RR+POLR algorithm in a cell slot. Here, we give an example that RR+POLR algorithm is performed in the 1^{st} Cell_Slot.

In the 1^{st} Rsv_Slot of every cell slot, according to the one-to-one mapping, an ISB has the highest priority to reserve as many links as possible to its Master-OSB. Link reservation in an ISB is determined by queue occupancy of the related GVOQ :

Table 4.1 RR+POLR Algorithm

```
RR+POLR Algorithm :
/* the $i^{th}$ ISB ($0 \leq i < K$) makes link reservation in a cell slot */

Given : $Q^i$ , LK_RSV$^i$ ;

Step 1 : At the start of a new cell slot ,
    1.1  reset LK_RSV$^i$ = [ $r_0^i$ , $r_1^i$ , $\cdots$, $r_{K-1}^i$] = [0 , 0 , $\cdots$, 0 ];
    1.2  find the $i^{th}$ ISB's Master_OSB resulted from the RR one-to-one
         mapping in current cell slot;

Step 2 : In the 1st Rsv_Slot of a cell slot ,
    2.1  the $i^{th}$ ISB has the highest priority to reserve as many links
         as possible to its Master_OSB :
             $r^i_{Master\_OSB} = min\{ M , q^i_{Master\_OSB} \}$ ;
    2.2  the $i^{th}$ ISB issues a token which is related to its Master_OSB,
         i.e. TOKEN$_{Master\_OSB}$ in which :
             Num_Lk_Idle = M - $r^i_{Master\_OSB}$ ;

Step 3 : In the $n^{th}$ Rsv_Slot of a cell slot ( $n \geq 2$ ) ,
    3.1  the $i^{th}$ ISB receives a token, saying TOKEN$_l$ , $0 \leq l < K$ ; then,
         the $i^{th}$ ISB will reserves as many links as possible to the $l^{th}$ OSB :
             $r_l^i = min\{$ Num_Lk_Idle$_l$ , $q_l^i$ , $(M - \sum_{j=0}^{(K-1)} r_j^i)$ $\}$ ;
    3.2  the $i^{th}$ ISB modifies TOKEN$_l$ as follows :
             Num_Lk_Idle = Num_Lk_Idle - $r_l^i$ ;
         then, the $i^{th}$ ISB passes TOKEN$_l$ to next ISB;
    3.3  if ( n*Rsv_Slot < Cell_Slot)
             Go to Step 3 ( now, n = n+1);
         else   Go to Step 4 ;

Step 4 : When a cell slot ends ,
    4.1  the $i^{th}$ ISB delivers cells to ATMCSF according to LK_RSV$^i$;
    4.2  the $i^{th}$ ISB discards its received token;
    4.3  Go to Step 1 ;
```

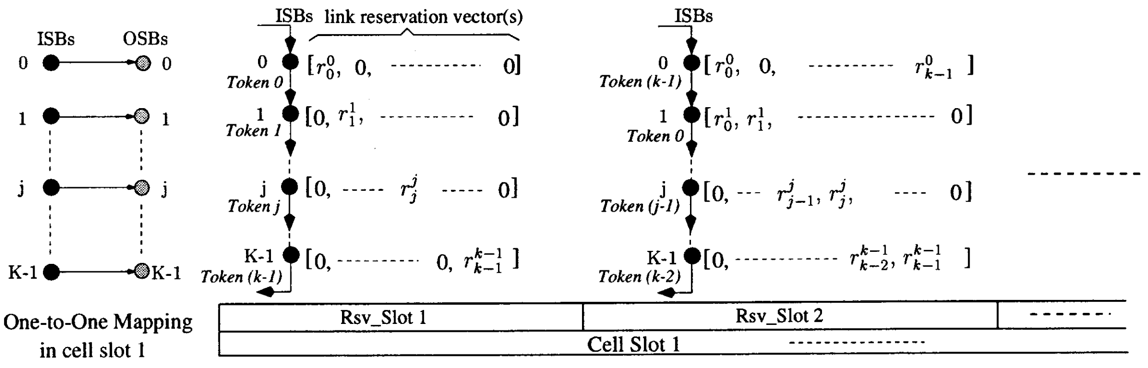


Figure 4.2 Round Robin Prioritized Output Link Reservation (RR+POLR) algorithm performed in the 1st Cell_Slot.

$$for\ \forall i \in [0, K-1), \quad r^i_{Master-OSB} = \begin{cases} M & \text{if } q^i_{Master-OSB} \geq M; \\ q^i_{Master-OSB} & \text{if } q^i_{Master-OSB} < M; \end{cases} \tag{4.1}$$

After reserving links to its Master-OSB, every ISB initiates a token about its own Master-OSB, and fills in "Num_Lk_Idle" field to record how many links to its Master-OSB are still available. Then, every ISB passes its new-born token to the down-link neighboring ISB. So far, K tokens are generated and start circulating on the token ring. A Token passes an ISB in every Rsv_Slot.

In the n^{th} $(n > 1)$ Rsv_Slot of the same cell slot, every ISB will receive a token from its up-link neighbor. If the received token carries available links, an ISB checks the queue occupancy of the related GVOQ and reserves as many links as possible to the identified OSB. The total links reserved in an ISB should not exceed M (i.e. $\sum_{j=0}^{(K-1)} r^i_j \leq M$ for $\forall i \in [0, K)$). The value of the "Num_Lk_Idle" field in the token will be reduced by the number of links occupied by the ISB. At the end of the n^{th} Rsv_Slot, every ISB will hand over its received token to next ISB.

When a cell slot is due, every ISB delivers cells to the central switch fabric based on its current link reservation vector. Meantime, all existing tokens in current cell slot will be destroyed by ISBs. **Note that**, link reservation rate (i.e. Rsv_Slot) is independent of a cell scheduling cycle (i.e. Cell_Slot), usually Rsv_Slot $\ll$ Cell_Slot. If K*Rsv_Slot $\leq$ Cell_Slot, a token can finish a complete ring in a cell slot. Otherwise, a token only goes through some ISBs in a cell slot.

When a new cell slot starts, each ISB resets its link reservation vector. According to the one-to-one mapping in the new cell slot, an ISB resumes link reservation with the highest priority from its new Master-OSB. Round Robin mapping ensures that an ISB treats every OSB as its Master-OSB in every K cell slots. Fairness in RR+POLR algorithm is guaranteed.

4.3.3 Remarks

4.3.3.1 Switch II vs. Switch I : In Switch II, each ISB needs to reset its link reservation vector in every cell slot. Link reservation rate is identified by Rsv_Slot which is independent from cell delivery rate represented by Cell_Slot.

We use an integer R to represent the ratio of Cell_Slot and Rsv_Slot, i.e. $R = \frac{Cell\_Slot}{Rsv\_Slot} \geq 1$. If $R = 1$, a token is only able to pass an ISB in a cell slot. Switch II will be the same as Switch I in this case, because an ISB just makes link reservation for its mapped OSB but does not have opportunity to reserve links to other OSBs. If $R > 1$, Switch II is superior over Switch I because a token can go through R ISBs so that an ISB can reserve links to several OSBs a cell slot. If $R \geq K$, a token can finish a complete ring during a cell slot, hence, an ISB is able to make link reservation for every OSB. Obviously, Switch II deserves better performance than Switch I if $R > 1$.

4.3.3.2 Complexity of RR+POLR Algorithm : RR+POLR algorithm is a distributed link reservation algorithm in the way that an ISB reserves links to an OSB according to its queue occupancy of the related GVOQ. Arbitration complexity if $O(1)$, though an ISB may repeat the same arbitration R times for different OSBs in a Cell_Slot.

4.3.3.3 Fairness of RR+POLR Algorithm :

Because of employing the one-to-one RR mapping, an ISB fairly selects each OSB as its Master_OSB in every K cell slots. In another words, every ISB has the same opportunity to make link reservation for K OSBs. Fairness is guaranteed in RR+POLR algorithm.

4.4 Switch Performance

In this section, we investigate Switch II through a performance comparison between Switch I, Switch II and an OQ switch. We simulate an 256x256 (N = 256) switch consisting of 32x32 ISBs/OSBs (M = 32, K = 8) for Switch I and Switch II. As a comparison, we also simulate an 256x256 OQ switch under the same traffic condition. Output queued switch is assumed to have infinite output buffers. A cell arriving at any switch input will be forwarded to the related output queues in the same cell slot. The OQ switch is work-conserving so that it results in the best performance.

VBR sources are applied to generate the input traffic, as shown in Fig 3.7. We use the ON (active)/OFF (idle) model to describe the burst-idle process of input traffic stream. The back-to-back cells in an ON duration belong to a same VC, i.e. they have same multiple destinations. No cells arrive in an idle period. Traffic parameters are : MBS which is the maximum burst size; LCR which is the line cell rate, it approximates $\frac{155,520,000}{53*8} = 366,793(cells/sec)$ for an OC-3 link; ACR which is the average cell rate. The effective input load is defined as $\rho = (ACR * F)/LCR$, $\rho \leq 1$, F is the average fanout.

As we mentioned, if $R = \frac{Cell\_Slot}{Rsv\_Slot} = 1$, Switch II will be the same as Switch I. If $R > 1$, Switch II will exceed Switch I because in a cell slot, an ISB can reserve links to multiple OSBs so that link starvation of OSBs may be relaxed. Since we have shown in chapter 3 that Switch I can obtain a good performance under uniform traffic, Switch II will be able to achieve a good performance under uniform traffic. Hence, we will not evaluate Switch II under uniform traffic, but, we mainly examine Switch II under **non-uniform traffic** to show the differences and likenesses from Switch I.

Three circumstances are likely to build a non-uniform traffic. (1) If maximum bursty size MBS is very large, then cells in an ON period will keep targeting the same multiple destinations for a relatively long time. Cell destinations of an ISB are not uniformly distributed among N switch outputs in a time period. (2) If bursts are correlated with each other, cells in successive ON bursts have the same destination outputs. Even though MBS is small, cells accumulated in several bursts will make the traffic non-uniformly distributed among N output ports. (3) In an extreme case, arriving cells to an ISB only go to a specific OSB. This is so called '1 ISB $\rightarrow$ 1 OSB HotSpot Traffic'. In our simulation, we use this traffic to perform a comparison study because this traffic model is able to clearly expose the strength or weakness of different switches.

Table 4.4 illustrates switch performance under **unicast** "1 ISB $\rightarrow$ 1 OSB HotSpot Traffic". The average hot spot burst length is $5 * MBS = 100$ successive cells. It is observed that Switch I suffers a dramatic performance degradation, while Switch II obtains better performance when R ($= \frac{Cell\_Slot}{Rsv\_Slot}$) increases.

Throughput performance is compared in Fig 4.3. Switch I with GVOQ RR obtains a very low throughput about 13% when input load is 99%. The reason for

Table 4.2 Switch II : performance comparison under non-uniform unicast traffic with different input load ρ. The observed performance statistics are : (1) throughput; (2) average end-to-end cell delay and delay jitter (D_{E-to-E}, (Min, Max)); (3) average cell delay in ISB and delay jitter (D_{ISB}, (Min, Max)); (4) average occupancy of OSB (S_{OSB} and (Min, Max)).

		1 ISB - 1 OSB HotSpot, Unicast, Burst Traffic (F = Fout = 1 , MBS = 20, HotSpotLen = 20*5)			
		OQ Switch *(256x256 Switch)*	***Switch I*** ***with GVOQ RR*** ***N = 256 ; K = 8 ; m = M = 3 ;***	***Switch II with RR+POLR*** ***N = 256 ; K = 8 ; m = M = 32 ;*** $R = \frac{Cell\_Slot}{Rsv\_Slot}$;	
				R = 4	***R = 8***
ρ = **0.99**	Throughput	0.99	0.12949	0.51807	0.98933
	$D_{E\text{-to-}E}$ (Min, Max)	88 (1.0 , 335)	2260 (32.5 , 2454)	1260 (6.4 , 1388)	94 (6.4 , 350)
	D_{ISB} (Min, Max)	N/A	2197 (8.3 , 2386)	1198 (0.1 , 1309)	1.0 (0.1 , 1.0)
	S_{OSB} (Min, Max)	N/A	14.2 (5.5 , 26.8)	112 (59.0 , 156)	2825 (2565 , 3334)
ρ = **0.90**	Throughput	0.90	0.12926	0.51805	0.89983
	$D_{E\text{-to-}E}$ (Min, Max)	33.8 (1.0 , 248)	1860 (30 , 2012)	875 (5.8 , 912)	40.2 (5.8 , 264)
	D_{ISB} (Min, Max)	N/A	1833 (7.0 , 1985)	445 (0.1 , 863)	0.53 (0.1 , 1.0)
	S_{OSB} (Min, Max)	N/A	11.5 (5.5 , 17.4)	105 (62.5 , 145)	958 (797 , 1134)
ρ = **0.70**	Throughput	0.70	0.12883	0.50567	0.69993
	$D_{E\text{-to-}E}$ (Min, Max)	11.2 (1.0 , 112)	1730 (25.0 , 1887)	24.3 (5.4 , 156)	16.2 (5.2 , 121)
	D_{ISB} (Min, Max)	N/A	1705 (5.8 , 1791)	5.6 (0.1 , 18.6)	0.15 (0.1 , 1.0)
	S_{OSB} (Min, Max)	N/A	8.0 (6.0 , 12.4)	97 (70 , 129)	247 (183 , 322)
ρ = **0.50**	Throughput	0.50	0.12806	0.49925	0.49997
	$D_{E\text{-to-}E}$ (Min, Max)	5.0 (1.0 , 65)	1610 (21.5 , 1812)	13.5 (5.0 , 75)	10.2 (4.8 , 75)
	D_{ISB} (Min, Max)	N/A	1594 (4.0 , 1670)	2.4 (0.1 , 16.2)	0.08 (0.01, 1.0)
	S_{OSB} (Min, Max)	N/A	7.4 (6.0 , 9.6)	97.0 (70.5 , 130)	87 (54.2 , 127)

that is, an ISB accommodates cells going to a specific OSB so that an ISB only have cells to be delivered to OSBs in 1 out of every 8 successive cell slots. Link resources are wasted by using the one-to-one RR mapping scheduling algorithm. However, Switch II using RR+POLR provides a mechanism for ISBs to compensate each other to reserve links and forward cells to the non-fully loaded OSBs. Starvation of OSBs may be relaxed according to the ratio R. For example, if $R = 4$ which implies that an ISB is allowed to reserve links to 4 OSBs in a cell slot, switch throughput leads to 52%; if $R = 8$, i.e. an ISB has the opportunity to reserve links to every OSB, Switch II can achieve a similar performance of throughput as the OQ switch.

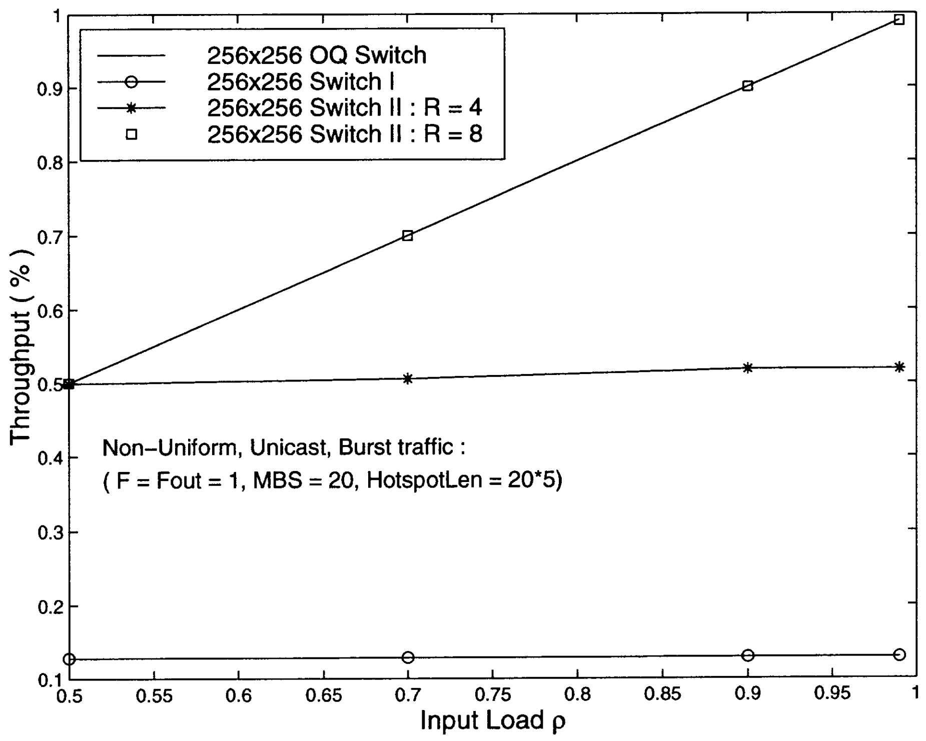


Figure 4.3 Switch II : throughput under "1 ISB $\rightarrow$ 1 OSB hotspot unicast traffic".

Fig 4.4 evaluates the average end-to-end cell delay (D_{E-to-E}), which is defined as the latency for a cell to pass through the switch. D_{E-to-E} is measured in terms of the number of cell slots. When $R = 8$, Switch II yields a very similar performance of D_{E-to-E} as the OQ switch. Compared to the lower bound of D_{E-to-E} obtained in the OQ switch, Switch II incurs no more than 8 cell slots longer delay. But, Switch I as well as Switch II with $R = 4$ sustain much longer cell delay. Longer cell delay is due to the lower throughput. Since more and more cells are backlogged in ISBs, cell delay keeps increasing in both of the two switches.

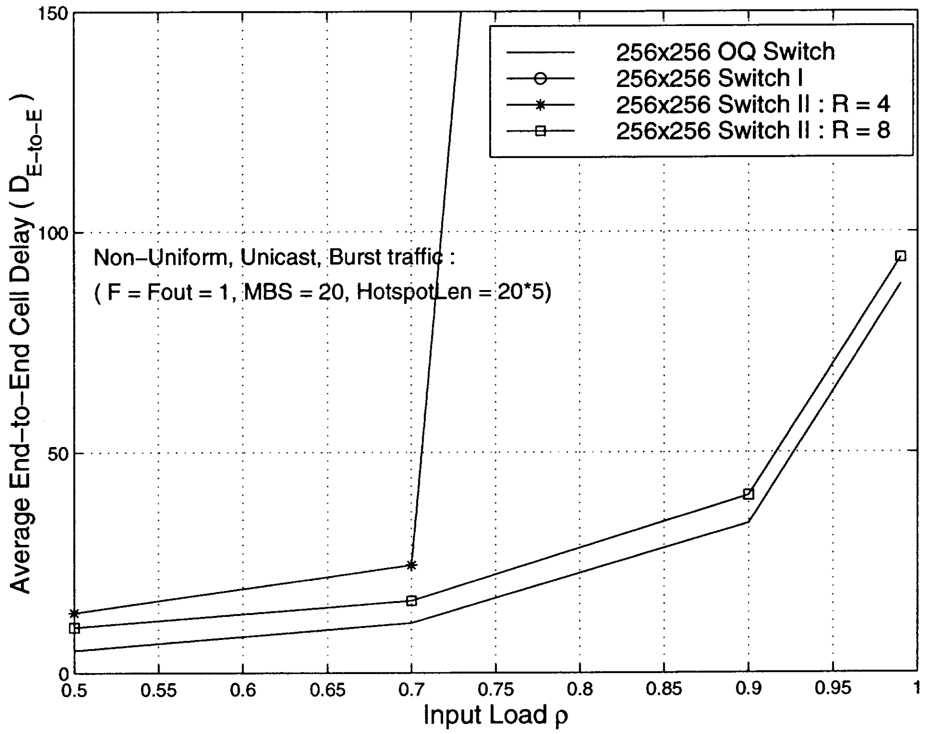


Figure 4.4 Switch II : average end-to-end cell delay (D_{E-to-E}) under "1 ISB $\rightarrow$ 1 OSB hotspot unicast traffic".

The end-to-end cell delay is resulted from two parts : queueing delay in ISBs (D_{ISBs}), and queueing delay in OSBs (D_{OSBs}). Fig 4.5 shows the average queueing

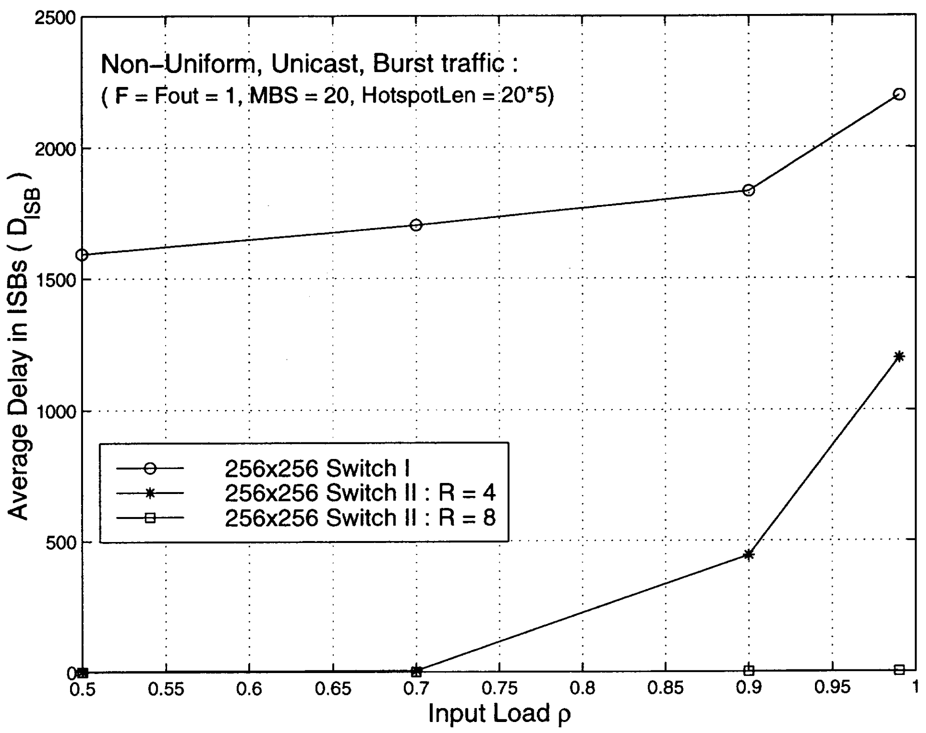


Figure 4.5 Switch II : average cell delay in ISB (D_{ISB}) under "1 ISB $\rightarrow$ 1 OSB hotspot unicast traffic".

delay in ISBs (i.e. D_{ISB}). When $R = 8$, Switch II causes at most 1 cell slot delay of D_{ISB}. It indicates a potential capability of Switch II to forward cells as fast as the OQ switch. In this circumstance, since most arriving cells are transmitted to OSBs immediately, OSBs in Switch II can employ any existing scheduling strategy to provide QoS guarantee as the OQ switch does. But, it is worth to notice that Switch II is sensitive to the ratio R. As shown in Fig 4.5, if $R = 4$, Switch II tolerates much longer delay in ISBs especially under heavy input load. Switch I performs even worse because $D_{E\text{-}to\text{-}E}$ is mainly incurred by the latency in ISBs (i.e. $D_{E\text{-}to\text{-}E} \approx D_{ISB}$). In this case, Switch I is more like an IQ switch.

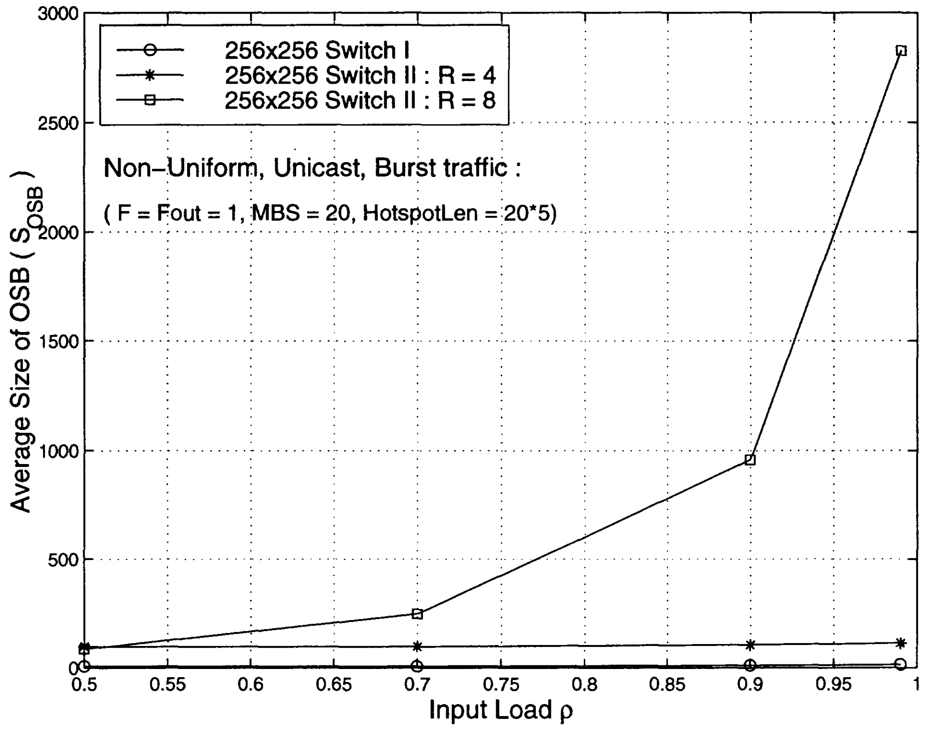


Figure 4.6 Switch II : average size of OSB (S_{OSB}) under "1 ISB → 1 OSB hotspot unicast traffic".

In addition, we measure the average size of OSBs (i.e. S_{OSB}) in Fig 4.6. S_{OSB} reflects the occupancy of OSBs in terms of the number of accommodated cells. Switch II with faster link reservation rate (i.e. large ratio R) is able to forward cells to OSBs quickly so that it will have more cells saved in OSBs.

Moreover, Table 4.4 investigates switch performance under **multicast** "1 ISB → 1 OSB HotSpot Traffic". Fig 4.7 ∼ Fig 4.10 respectively depicts performance of different aspects. Generally, we have similar observations as what we had discussed for unicast traffic. In addition, under multicast traffic, switches benefit from GVOQs so that multicast cells can be forwarded from ISBs to OSBs. Therefore, under the

Table 4.3 Switch II : performance comparison under non-uniform multicast traffic with different input load ρ. The observed performance statistics are : (1) throughput; (2) average end-to-end cell delay and delay jitter (D_{E-to-E}, (Min, Max)); (3) average cell delay in ISB and delay jitter (D_{ISB}, (Min, Max)); (4) average occupancy of OSB (S_{OSB} and (Min, Max)).

		1 ISB - 1 OSB HotSpot, Multicast, Burst Traffic (F = 4 , MBS = 20, HotSpotLen = 20*5)			
		OQ Switch *(256x256 Switch)*	*Switch I* *with GVOQ RR* *N = 256 ; K = 8 ; m = M = 3 ;*	*Switch II with RR+POLR* *N = 256 ; K = 8 ; m = M = 32 ;* $R = \frac{Cell\_Slot}{Rsv\_Slot}$;	
				R = 2	*R = 4*
ρ = 0.99	Throughput	0.99	0.53693	0.98822	0.98958
	$D_{E\text{-to-}E}$ (Min, Max)	78.5 (1.0 , 338)	1750 (14.2 , 2016)	110.5 (3.7 , 512)	102 (3.5 , 506)
	D_{ISB} (Min, Max)	N/A	210 (1.0 , 1740)	2.0 (1.0 , 27.8)	0.43 (0.1 , 4.0)
	S_{OSB} (Min, Max)	N/A	90 (67 , 136)	2392 (1663 , 3280)	2482 (1652 , 3445)
ρ = 0.90	Throughput	0.90	0.53381	0.89927	0.89963
	$D_{E\text{-to-}E}$ (Min, Max)	29.3 (1.0 , 253)	1560 (13.6 , 1845)	58.9 (3.6 , 381)	50.5 (3.6 , 374)
	D_{ISB} (Min, Max)	N/A	170 (1.0 , 1488)	1.1 (0.1 , 23.2)	0.40 (0.1 , 4.0)
	S_{OSB} (Min, Max)	N/A	90 (64 , 136)	839 (408 , 1446)	843 (395 , 1466)
ρ = 0.70	Throughput	0.70	0.51085	0.69957	0.69973
	$D_{E\text{-to-}E}$ (Min, Max)	9.1 (1.0 , 119)	1030 (12.5 , 1485)	32.5 (3.2 , 176)	28.5 (3.2 , 176)
	D_{ISB} (Min, Max)	N/A	113 (1.0 , 920)	0.6 (0.1 , 11.0)	0.38 (0.1 , 4.0)
	S_{OSB} (Min, Max)	N/A	89.2 (56.0 , 131)	230 (95 , 417)	221.5 (87 , 413)
ρ = 0.50	Throughput	0.50	0.49925	0.49979	0.49990
	$D_{E\text{-to-}E}$ (Min, Max)	4.3 (1.0 , 66.0)	66 (11.2 , 830)	18.8 (3.0 , 82.2)	18.8 (3.0 , 82.0)
	D_{ISB} (Min, Max)	N/A	1.6 (1.0 , 49.2)	0.52 (0.1 , 7.8)	0.37 (1.0 , 4.0)
	S_{OSB} (Min, Max)	N/A	80 (35.5 , 129)	89.5 (26.0 , 186)	82.5 (22.5 , 178)

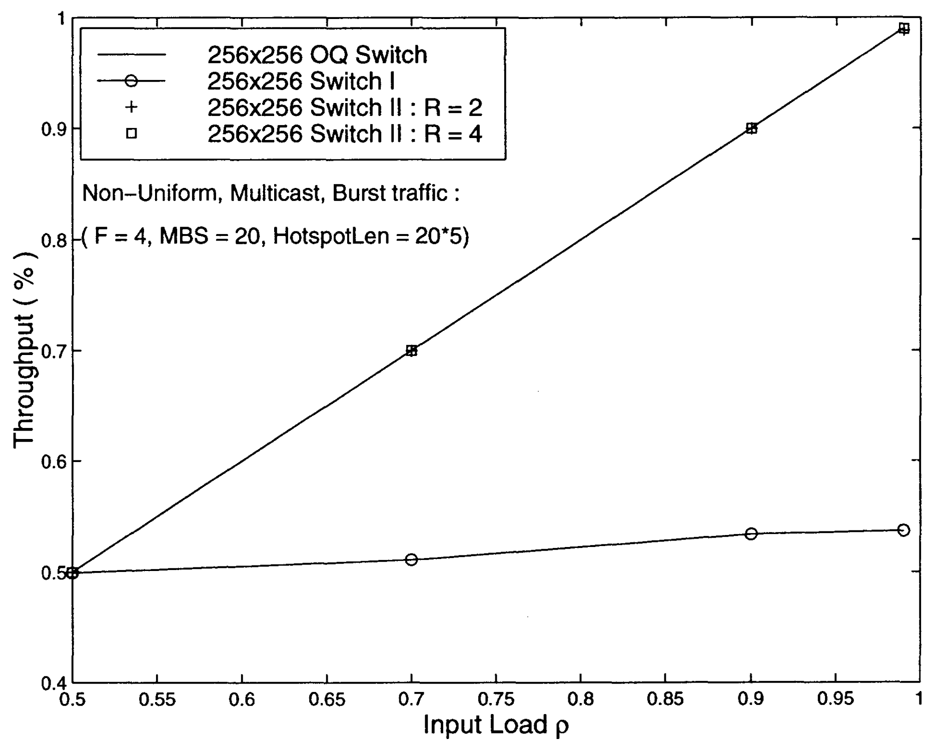


Figure 4.7 Switch II : throughput under "1 ISB → 1 OSB hopspot multicast traffic".

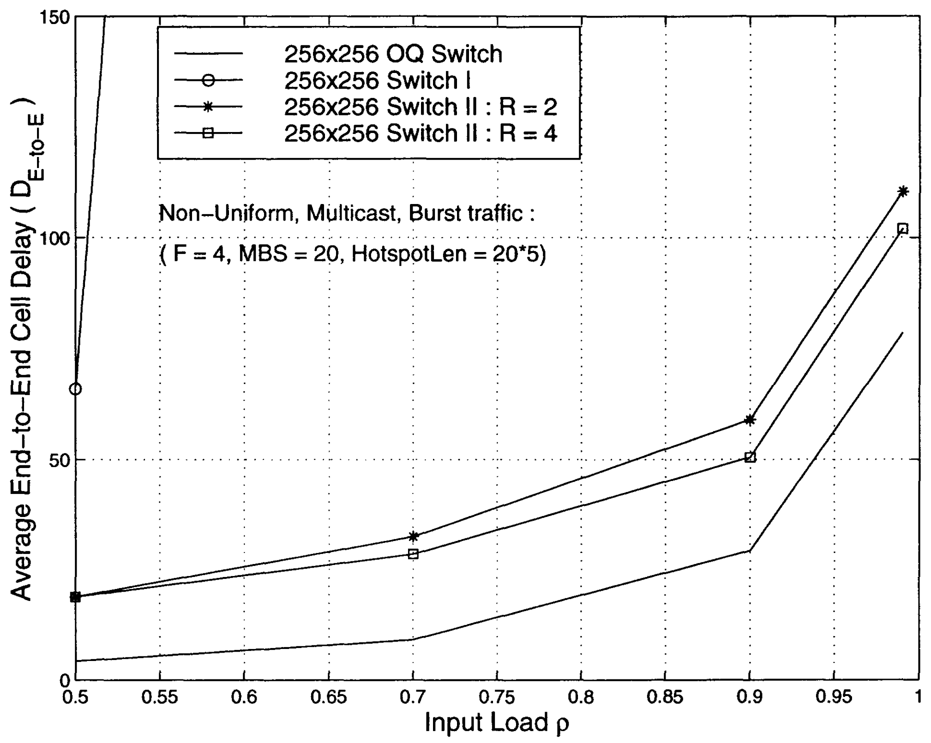


Figure 4.8 Switch II : average end-to-end cell delay ($D_{E\text{-}to\text{-}E}$) under "1 ISB $\rightarrow$ 1 OSB hopspot multicast traffic".

same input load ρ, switches achieve better performance than in unicast traffic. For example, Switch II with $R = 4$ is able to obtain a comparable performance to the OQ switch under multicast traffic. But, it is not the case in unicast traffic.

4.5 Conclusion

4.5.1 Advantages of Switch II

Switch II inherits the modular switch architecture of Switch I. It benefits from input and output link sharing, hence, no speedup is necessary in central switch fabric.

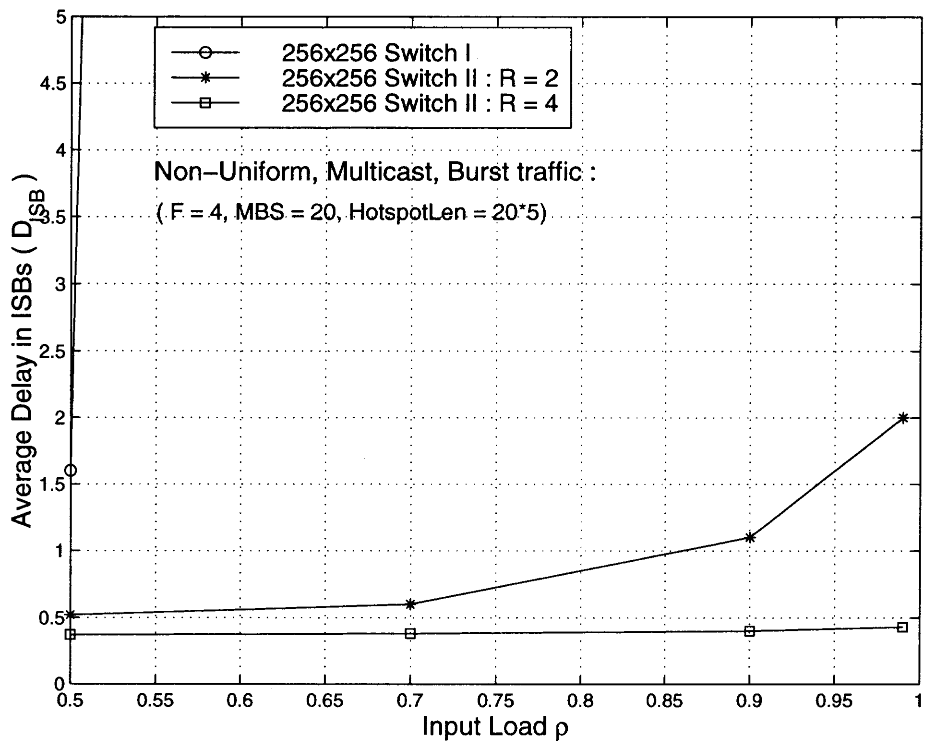


Figure 4.9 Switch II : average cell delay in ISB (D_{ISB}) under "1 ISB → 1 OSB hopspot multicast traffic".

To resolve input and output contention, we propose a Round Robin Prioritized Output Link Reservation (RR+POLR) algorithm. Cell delivery is determined by link reservation in every ISB. RR+POLR algorithm is a distributed resource allocation algorithm, in the sense that an ISB makes link reservation for an OSB in a Rsv_Slot according to queue occupancy of the related GVOQ. Arbitration complexity is O(1).

Switch II uses RR+POLR to avoid starvation of OSBs so that it achieves an improved performance especially under non-uniform traffic. If $R \geq K$, i.e. a token can circulate a complete ring in a cell slot, RR+POLR may be able to gain a comparable performance to the OQ switch.

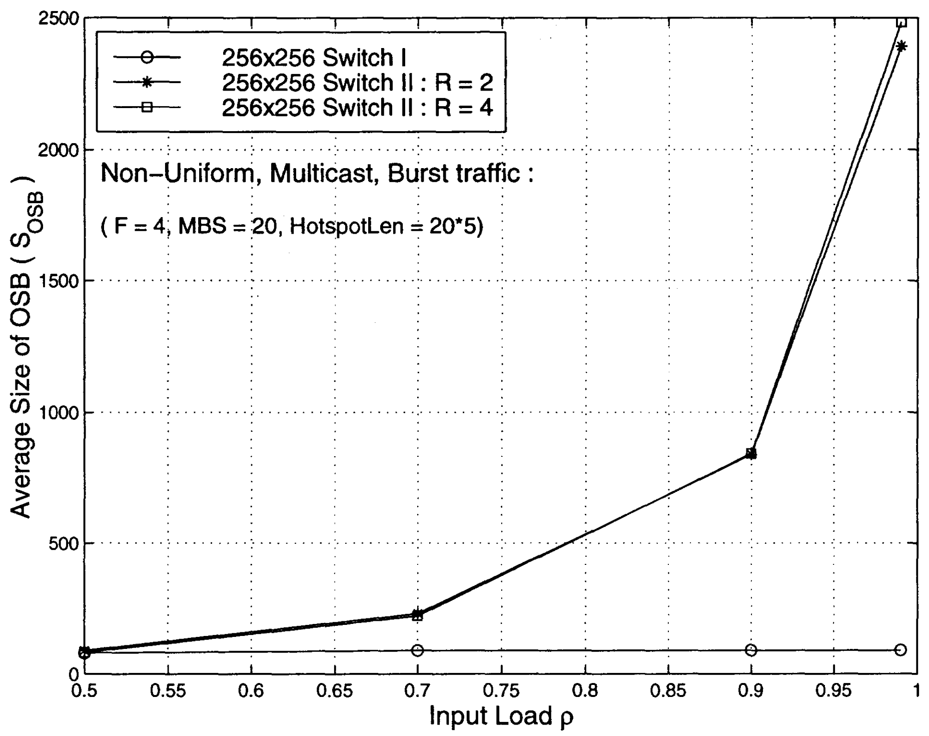


Figure 4.10 Switch II : average size of OSB (S_{OSB}) under "1 ISB → 1 OSB hopspot multicast traffic".

4.5.2 Disadvantages of Switch II

RR+POLR algorithm has a deficiency to achieve an efficient link resource allocation among ISBs, mainly due to following two reasons.

First, it may not be efficient for an ISB to reset its link reservation vector in every cell slot, because traffic patterns injected in an ISB usually will not change dramatically in every cell slot. In addition, the performance of Switch II is mainly depended on the link reservation rate, i.e. the ratio of Rsv_Slot and Cell_Slot. If Rsv_Slot = Cell_Slot, Switch II is exactly the same as Switch I because an ISB only has the opportunity to reserve links to its Mater_OSB in a cell slot.

Moreover, prioritized link reservation may hinder an ISB to reserve links for a starved OSB if the ISB has already reserved M links to other OSBs. Fig 4.11 shows an example of this scenario. In the 1^{st} Rsv_Slot, according to the one-to-one mapping, ISB 0 and ISB 1 respectively reserve links to their Master-OSBs. After that, no idle link is left for OSB 0, but 2 links to OSB 1 are still available since ISB 1 does not have cells destined to OSB 1. However, when receiving $Token_1$ in the 2^{nd} Rsv_Slot, ISB 0 is not able to reserve any more links (refer to $*$ in Fig 4.11). Eventually, OSB 1 is not served with any cells even though there are cells in ISBs which want to go to OSB 1 (ref $\sharp$ in Fig 4.11). This causes throughput degradation. The problem is due to prioritized link reservation without the knowledge of traffic load of other OSBs.

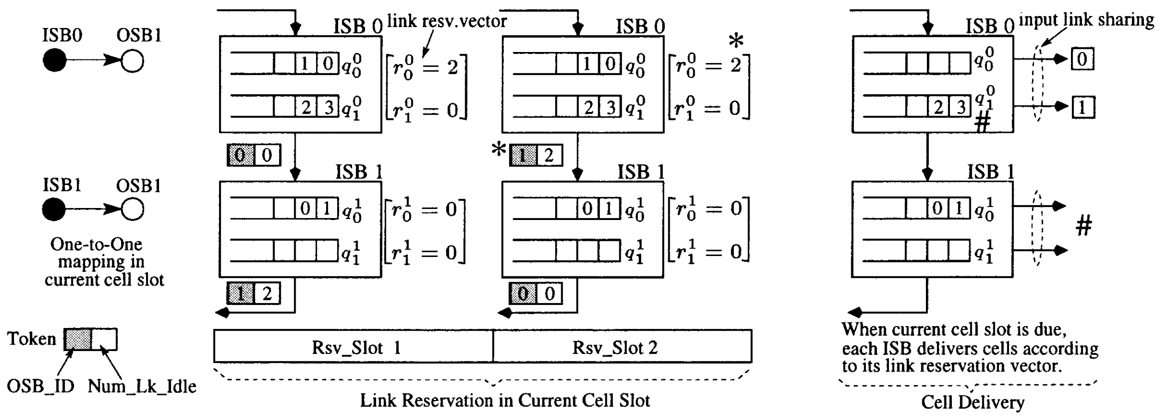


Figure 4.11 RR+POLR causes starvation of OSBs (refer to $\sharp$) in an example 4x4 switch, 2x2 ISBs/OSBs (N=4,M=2,K=2).

To resolve this problem, in next chapter, we present an enhanced switch architecture using dual round robin dynamic link reservation to achieve a dynamic fast and fair link resource allocation among ISBs.

CHAPTER 5

SWITCH III: A SCALABLE TERABIT MULTICAST PACKET SWITCH WITH DUAL ROUND ROBIN DYNAMIC LINK RESERVATION

In this chapter, we propose Switch III as an enhanced switch design using link sharing and dual round robin dynamic link reservation. Unlike the previous two switches, ISBs are connected by dual rings on which K link request tokens (REQs) and K link release tokens (RELs) circulate in a round robin manner. Cell delivery is based on link reservation in every ISB. But, without reseting its link reservation vector in every cell slot, each ISB can dynamically increase/decrease its link reservation for a specific OSB by "borrowing" or "lending" links from/to other ISBs. We propose two Queue Occupancy Based Dynamic Link Reservation (QOBDLR) algorithms to achieve a fast and fair link resource allocation among ISBs. QOBDLR is a distributed link reservation scheme in the sense that every ISB utilizes its local available information to arbitrate a modification for its own link reservation. Arbitration complexity is $O(1)$. Performance evaluation shows that Switch III can achieve s comparable performance to OQ switches under any traffic pattern. Moreover, Switch III avoids the speedup problem which is involved in OQ switches. Hence, Switch III would be a good choice for high performance, scalable, large-capacity core switches.

5.1 Switch Architecture

Fig 5.1 shows the architecture of Switch III, which consists of K ISBs, ATMCSF, K OSBs, and dual round robin rings. Functions of ISB, OSB and ATMCSF are the same as what we presented in Switch I and Switch II.

ISBs are connected by dual rings : a down-ward ring conveys link request tokens (REQs); and an up-ward ring carries link release tokens (RELs). At any time, there

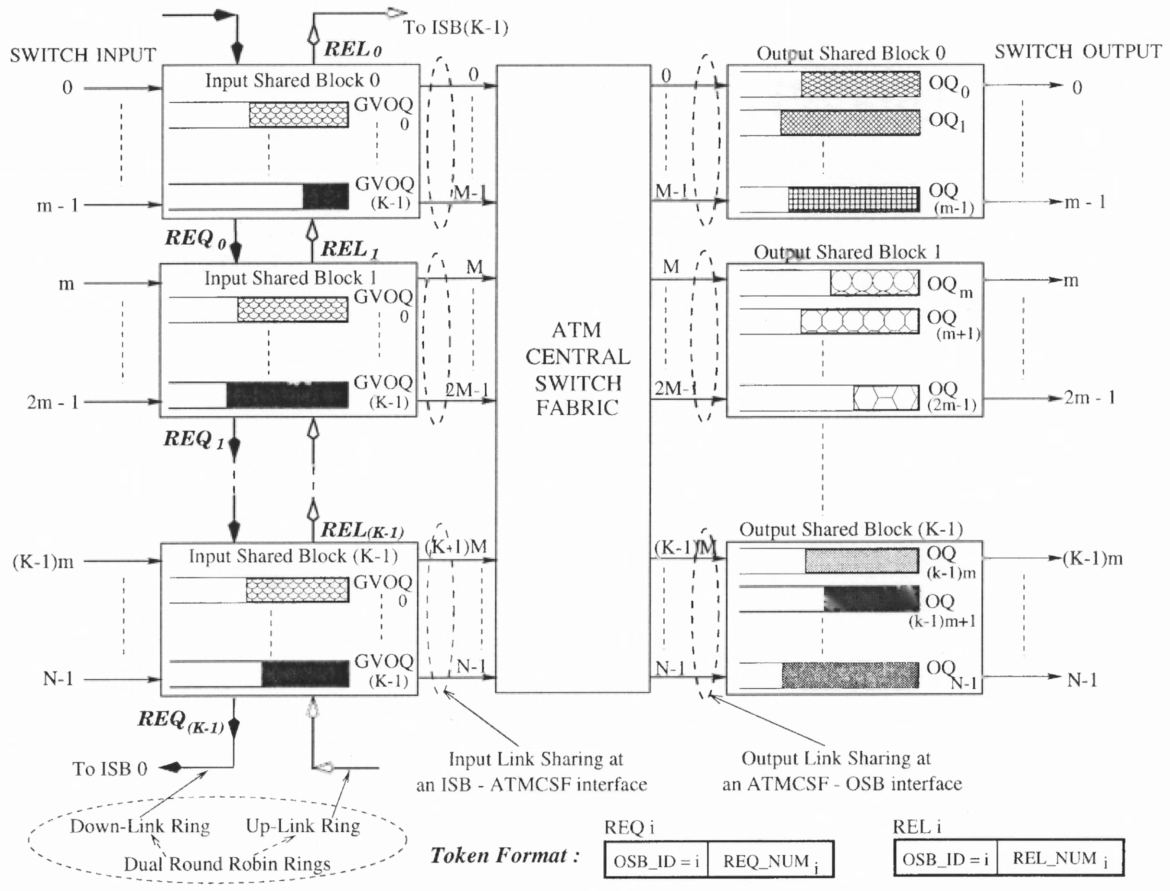


Figure 5.1 Switch III : an NxN switch consists of K ISBs, K OSBs, ATMCSF, and dual round robin rings; $K = \frac{N}{m}$, $m = M$. Cell delivery is based on link reservation. Dual round robin rings provide a mechanism for ISBs to dynamically "borrow" and/or "lend" links from each other.

are K REQ tokens and K REL tokens circulating on the dual rings respectively and passing ISBs one by one in a round robin manner. Each OSB (e.g. the i^{th} OSB) is correlated with a REQ token (e.g. REQ_i) and a REL token (e.g. REL_i).

As shown in Fig 5.1, REQ token and REL token have the same format containing two fields : (1) "OSB_ID" is the identification of an OSB; (2) "REQ_NUM" indicates how many link requests are issued for the identified OSB. Or, "REL_NUM" records the number of released links which are available to be reserved at the related ATMCSF-OSB interface.

5.2 Cell Scheduling

5.2.1 Cell Delivery

Cell delivery is based on link reservation. Each ISB has a *link reservation vector* and a *queue occupancy vector*. We use LK_RSV^i and Q^i to represent the two vectors in the i^{th} ISB $(0 \le i,\ j\ < K)$:

- LK_RSV^i : **link reservation vector in the i^{th} ISB.**

  $\text{LK\_RSV}^i = [r^i_0\ ,\ r^i_1\ ,\ \cdots,\ r^i_{(K-1)}]$, where r^i_j indicates how many links at the ATMCSF-OSB j interface are reserved by the i^{th} ISB, $0 \le r^i_j \le M$.

- Q^i : **queue occupancy vector in the i^{th} ISB.**

  $\text{Q}^i = [q^i_0\ ,\ q^i_1\ ,\ \cdots,\ q^i_{(K-1)}]$, where q^i_j is the queue length (# of cells) of the j^{th} GVOQ in the i^{th} ISB, $q^i_j \ge 0$.

In a cell slot, each ISB delivers cells to central switch fabric according to its link reservation. For example, as shown in Fig 5.2, if LK_RSV^i is $[4,\ 0,\ \cdots,\ 2]$ in current cell slot, the i^{th} ISB will send two cells to OSB 0 and four cells to OSB (K-1), but no cells are scheduled to other OSBs.

But, unlike Switch II, ISBs in Switch III do not need to reset their link reservation vectors in every cell slot. According to its queue occupancy vector, an ISB can dynamically modify its link reservation for a specific OSB when the ISB receives the related REQ token or REL token. We propose two link reservation algorithms, both of them are based on a *queue occupancy based dynamic link reservation (QOBDLR)* scheme.

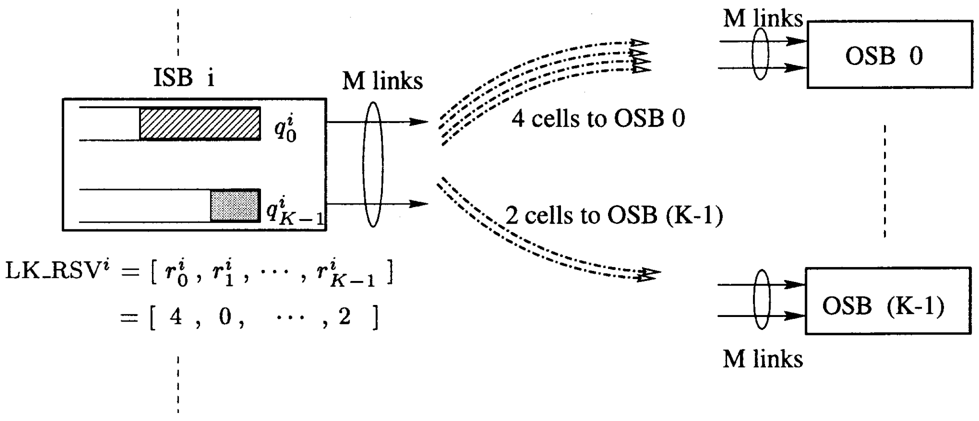


Figure 5.2 Cell delivery is based on link reservation.

5.2.2 Link Reservation

Link reservation in ISBs needs to resolve two contentions : (1) K GVOQs in an ISB contend for M links at the ISB-ATMCSF interface. (2) K ISBs contend for M links at every ATMCSF-OSB interface. To achieve a **fast and fair** link resource allocation among ISBs, we propose two algorithms :

• *REQ-QOBDLR algorithm* : Request-Motivated Queue Occupancy Based Dynamic Link Reservation algorithm.

• *REQREL-QOBDLR algorithm*: Request/Release-Motivated Queue Occupancy Based Dynamic Link Reservation algorithm.

To present the two link reservation algorithms, we provide following definitions.

Definition 1 : Link Reservation Rule.

Link reservation among K ISBs must satisfy two criteria :

(1) $\sum_{j=0}^{K-1} r_j^i \leq M$, i.e. the total links reserved by the i^{th} ISB can not exceed M which is the maximum number of links at an ISB-ATMCSF interface;

(2) $\sum_{i=0}^{K-1} r_j^i \leq M$, i.e. the total links reserved by all ISBs to the j^{th} OSB can not exceed M which is the maximum number of links at the ATMCSF-OSB interface.

Definition 2 : Link Reservation Slot, i.e. Rsv_Slot.

As shown in Fig 5.3, Rsv_Slot is defined as a small time interval during which an ISB receives a pair of REQ and REL tokens. In a Rsv_Slot, an ISB has the authority to modify its link reservation for the two OSBs which are identified by the received REQ and REL tokens.

Rsv_Slot is independent from Cell_Slot, usually, Rsv_Slot $\ll$ Cell_Slot. We define R as the ratio of the cell slot to the link reservation slot, i.e. $R = \frac{Cell\_Slot}{Rsv\_Slot}$. If $R = 1$, link reservation is performed in the slowest rate because a token only goes through one ISB in a cell slot; if $R \geq K$, a token can circulate a complete ring in a cell slot. Dynamic link reservation is operated in a Rsv_Slot. When a cell slot is due, every ISB delivers cells to ATMCSF according to its current link reservation vector.

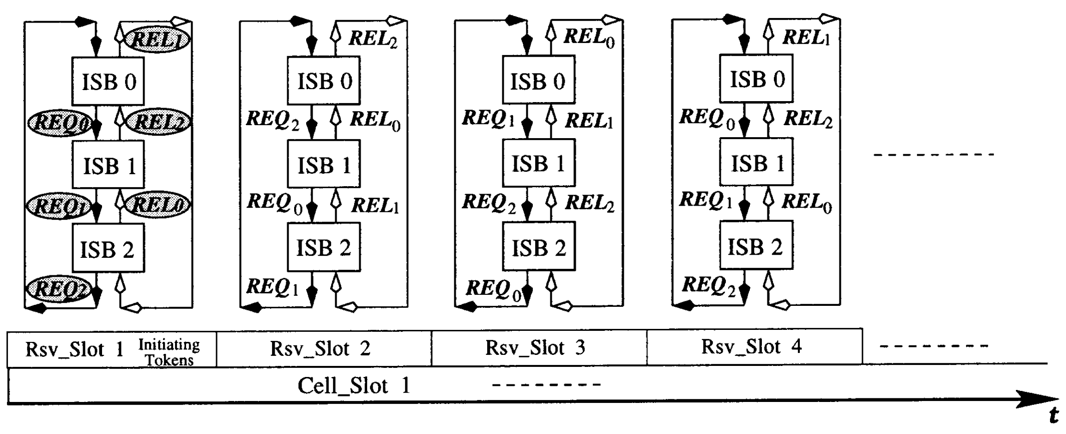
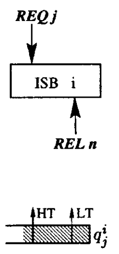


Figure 5.3 Rsv_Slot (link reservation slot) vs. Cell_Slot (cell delivery slot) in an example switch consisting of 3 ISBs and 3 OSBs.

5.3 REQ-QOBDLR Algorithm

In this section, we present the *Request-Motivated Queue Occupancy Based Dynamic Link Reservation (REQ-QOBDLR)* algorithm. As a common model shown in Fig 5.4, the i^{th} ISB ($0 \leq i < K$) is receiving REQ_j token and REL_n token in current Rsv_Slot, usually REQ_j and REL_n identify two different OSBs (i.e. $j \neq n$). The i^{th} ISB will only modify its link reservation, i.e. r_j^i and r_n^i, for the j^{th} OSB and the n^{th} OSB in current Rsv_Slot.



	Queue Occupancy $q_j^i < LT$ (area 1 : light load)	Queue Occupancy $LT \leq q_j^i \leq HT$ (area 2 : normal load)	Queue Occupancy $q_j^i > HT$ (area 3 : heavy load)
Operations Upon Receiving REQ j (QOBDLR)	the i^{th} ISB will request 0 extra link to the j^{th} OSB ;		the i^{th} ISB will request 1 extra link to the j^{th} OSB;
	if ((REQ_NUM$_j$ > 0) and ($r_j^i > 0$)) then the ISB will release a link for OSB j ;	if ((REQ_NUM$_j$ > 0) and ($r_j^i > Fair$)) then the ISB will release a link for OSB j ;	
Operations Upon Receiving REL n	if (REL_NUM$_n$ > 0) , and ($\sum_{l=0}^{(k-1)} r_l^i < M$) , and (the i^{th} ISB has requested an extra link to the n^{th} OSB) then the i^{th} ISB take a link from REL_n token ;		

Figure 5.4 REQ-QOBDLR algorithm which is performed in every Rsv_Slot.

REQ-QOBDLR algorithm is performed in every Rsv_Slot. For the received REQ_j token, the i^{th} ISB will refer the queue occupancy q_j^i to decide an intended modification of r_j^i. For the received REL_n token, the i^{th} ISB simply takes an extra link if the ISB had requested it. As the increasing/decreasing of link reservation for a specific OSB can only be triggered and accomplished as a result of an explicit request by the i^{th} OSB, this algorithm is so called **REQ-QOBDLR algorithm**.

The detail of REQ-QOBDLR algorithm is illustrated in Appendix A. In this section, we present the basic rule and main operations of REQ-QOBDLR algorithm.

5.3.1 Operations upon receiving REQ_j Token

When receiving REQ_j token, the i^{th} ISB will evaluate its queue occupancy q_j^i against two thresholds : a high threshold (HT) and a low threshold (LT). Then, the i^{th} ISB decides whether to request an extra link and/or release a link to the j^{th} OSB.

If $q_j^i > HT$, the i^{th} ISB will request an additional link for the j^{th} OSB. Note that, if the i^{th} ISB had sent a link request before but has not obtained the desired link yet, the i^{th} ISB will not issue an extra new-born link request but just keep asking for one additional link if $q_j^i > HT$. In addition, if the REQ_j token carries link requests (i.e. REQ_NUM$_j > 0$), the i^{th} ISB will schedule a single link release for the j^{th} OSB if the ISB reserves more than $Fair$ [1] links to the j^{th} OSB (i.e. $r_j^i > Fair$).

If $LT \leq q_j^i \leq HT$, the i^{th} ISB will release a link if REQ_j token carries link requests and if its link reservation for the OSB j is more than $Fair$ links; Otherwise, the ISB will keep the same reservation as before.

If $q_j^i < LT$, and if REQ_j token carries link requests, the i^{th} ISB will release one of its occupied link to the j^{th} OSB.

In general, a new-born link request for the j^{th} OSB will be added into REQ_j token, hence, REQ_NUM$_j$ will be increased by 1. On the other hand, if the i^{th} ISB releases a link to satisfy a link request in REQ_j token, REQ_NUM$_j$ will be reduced by 1. Usually, the link to be released for the j^{th} OSB can not be passed to other ISBs by REL_n token in current Rsv_Slot, if $j \neq n$. The i^{th} ISB needs to record this pending link release, and will add this released link into REL_j token when the i^{th} ISB receives REL_j token in some Rsv_Slot(s) later.

[1] $Fair = \lfloor \frac{M}{K} \rfloor$, i.e. the M links at an ATMCSF-OSB interface are fairly allocated to K ISBs

5.3.2 Operations upon receiving REL_n Token

When receiving REL_n token, the i^{th} ISB will decide whether to take a link from REL_n token, if REL_n token carries available links.

If the i^{th} ISB has issued a link request to the n^{th} OSB, and if the total links reserved by the i^{th} ISB is smaller than M (i.e. $\sum_{l=0}^{(k-1)} r_l^i < M$), then the i^{th} ISB will grab an available link and its link reservation to OSB n (i.e. r_n^i) will be increased by 1. Otherwise, the i^{th} ISB will not take any link from REL_n token.

REL_NUM$_n$ will be reduced by 1 if the i^{th} ISB reserves an available link from REL_n token. Moreover, before the i^{th} ISB forwards REL_n token to the next ISB, if the i^{th} ISB has a pending released link resulted from the operation of receiving REQ_n token, the released link will be inserted into REL_n token.

5.3.3 Remarks on REQ-QOBDLR Algorithm

During system initialization, all link resources are fully assigned to ISBs, i.e. $\sum_{i=0}^{(k-1)} r_l^i = M$ and $\sum_{l=0}^{(k-1)} r_l^i = M$ for $\forall i, j$. But, how to initialize link reservation vectors is not important because an ISB will dynamically "borrow" and/or "lend" links from/to other ISBs according to its traffic load. In our experiment, link reservation vector is initiated with a fair allocation, i.e. $r_j^i = \lfloor \frac{M}{K} \rfloor$ or $r_j^i = \lfloor \frac{M}{K} \rfloor + 1$.

In addition, when system starts, every ISB issues a REQ token and a REL token. There is no specific rule on how to establish the token sequence as long as each OSB is represented by a pair of REQ token and REL token. After that, K REQs and K RELs will keep circulating on the dual rings in a round robin manner.

Cell delivery and link reservation are independent operations. When a Cell_Slot is due, every ISB sends cells to ATMCSF based on its current link reservation vector. But, an ISB is able to change its link reservation in every Rsv_Slot, usually Rsv_Slot < Cell_Slot.

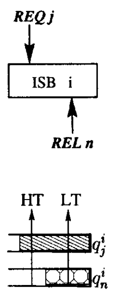
5.3.4 Conclusion on REQ-QOBDLR Algorithm

Dynamic link reservation in REQ-QOBDLR algorithm is triggered by issuing link requests. When receiving REQ_j token, the i^{th} ISB can ask for an extra link to the j^{th} OSB if $q_j^i > HT$. When receiving REL_n token, the i^{th} ISB will grab a released link if it had issued a link request and has been waiting for an available link. The advantage of REQ-QOBDLR algorithm is to ensure that requesting a link and/or releasing a link happens when necessary.

However, REQ-QOBDLR algorithm may have a potential problem especially when switch grows and K is large. The reason is that, the i^{th} ISB does not measure its queue occupancy q_n^i when receiving REL_n token. But, the traffic load in the n^{th} GVOQ (i.e. q_n^i) may have changed while the ISB is waiting for a preferred link. There are two possible circumstances when the i^{th} ISB receives REL_n token in current Rsv_Slot. The i^{th} ISB may desire an additional link to the n^{th} OSB (i.e. if $q_n^i > HT$) even though it did not issue a link request for the n^{th} OSB before. Or, the i^{th} ISB currently does not need the extra link even though it has sent out a link request before. To match the real traffic, it would be more effective if the i^{th} ISB evaluates queue occupancy q_n^i when dealing with REL_n token. This is the motivation for us to propose another competitive algorithm called REQREL-QOBDLR algorithm in next section.

5.4 REQREL-QOBDLR Algorithm

In this section, we present *Request/Release-Motivated Queue Occupancy Based Dynamic Link Reservation (REQREL-QOBDLR) algorithm.* As show in Fig 5.5, the i^{th} ISB is receiving REQ_j and REL_n token in current Rsv_Slot. For the received REQ_j token, the i^{th} ISB does the same operation as REQ-QOBDLR algorithm, i.e. the i^{th} ISB will evaluate the queue occupancy q_j^i to arbitrate a potential modification of r_j^i. Difference from REQ-QOBDLR algorithm exists in the operation upon receiving REL_n token : the i^{th} ISB will also measure the queue occupancy q_n^i to decide an intended modification of r_n^i. This algorithm is so called **REQREL-QOBDLR algorithm.**



Operation Upon Receiving REQ j (QOBDLR)	Queue Occupancy $q_j^i < LT$ (area 1 : light load)	Queue Occupancy $LT \leq q_j^i \leq HT$ (area 2 : normal load)	Queue Occupancy $q_j^i > HT$ (area 3 : heavy load)
	the i^{th} ISB will request 0 extra link to the j^{th} OSB;		the i^{th} ISB will request 1 extra link to the j^{th} OSB ;
	if ((REQ_NUM$_j > 0$) and ($r_j^i > 0$)) then the ISB will release a link for OSB j ;	if ((REQ_NUM$_j > 0$) and ($r_j^i > Fair$)) then the ISB will release a link for OSB j ;	
Operations Upon Receiving REL n (QOBDLR)	Queue Occupancy $q_n^i < LT$ (area 1 : light load)	Queue Occupancy $LT \leq q_n^i \leq HT$ (area 2 : normal load)	Queue Occupancy $q_n^i > HT$ (area 3 : heavy load)
	if (($r_n^i > Fair$) or ($r_n^i > q_n^i$)) then the ISB will release a link for OSB n ;	if (REL_NUM$_n > 0$) and ($\sum_{l=0}^{(k-1)} r_l^i < M$) and (the ISB has requested a link to the n^{th} OSB) then the i^{th} ISB will take a link from REL_n token ;	if (REL_NUM$_n > 0$) and ($\sum_{l=0}^{(k-1)} r_l^i < M$) then the i^{th} ISB will take a link from REL_n token ;

Figure 5.5 REQREL-QOBDLR algorithm which is performed in every Rsv_Slot. Assume that the i^{th} ISB is receiving REQ_j and REL_n token in current Rsv_Slot.

The detail of REQREL-QOBDLR algorithm is addressed in Appendix B. In this section, we present the basic idea of REQREL-QOBDLR algorithm.

5.4.1 Operations upon receiving REL_n Token

When receiving REL_n token, the i^{th} ISB will evaluate its queue occupancy q_n^i with HT and LT to decide an intended modification on r_n^i.

If $q_n^i > HT$, the i^{th} ISB will grab an additional link for the n^{th} OSB as long as following conditions are hold : (1) REL_n token carries available links (i.e. REL_NUM$_n > 0$); (2) the total number of links reserved by the i^{th} ISB is less than M (i.e. $\sum_{l=0}^{(k-1)} r_l^i < M$).

If $LT \leq q_n^i \leq HT$, the i^{th} ISB will take an extra link for the n^{th} OSB if following requirements are satisfied : (1) REL_n token carries available links (i.e. REL_NUM$_n > 0$); (2) the ISB has requested a link for the n^{th} OSB ; (3) the total number of links reserved by the ISB is smaller than M (i.e. $\sum_{l=0}^{(k-1)} r_l^i < M$).

If $q_n^i < LT$, the i^{th} ISB will schedule a link release if its current reservation for the n^{th} OSB is either greater than Fair (i.e. $r_n^i > Fair$) or greater than its current queue occupancy (i.e. $r_n^i > q_n^i$).

For the case of $q_n^i > HT$, it may happen that the i^{th} ISB takes a link from REL_n token but it had not sent a link request before. Under such circumstance, we term the i^{th} ISB as a 'thieving ISB', which "*steals*" an available link that may have been released for a link request issued by another ISB, namely the 'victim ISB'. The 'victim ISB' who has sent a link request and is waiting for an available link may never receive its desired link if there is a 'thieving ISB' on the way snatching an available link from REL_n token. To resolve this problem, the 'thieving ISB' should send a link request for the n^{th} OSB to motivate a link release not for itself but for the 'victim ISB'. Usually, this compensated link request for the n^{th} OSB can not be inserted into REQ_j token in current Rsv_Slot. The ISB has to record this pending link request and waits for receiving REQ_n token to send this link request to other ISBs.

For the case of $q_n^i < LT$, the i^{th} ISB's releasing a link for the n^{th} OSB is due to its own low traffic load and it does not need to be triggered by link requests in REQ_n token. It means that the i^{th} ISB releases a link for the n^{th} OSB without any knowledge of how many link requests are indicated in REQ_n token. In order to achieve an efficient link utilization, it is required that REL_NUM$_n \leq$ REQ_NUM$_n$, i.e. all available links will be used up by link requests and will be needed by some

ISBs. Bearing this in mind, when the i^{th} ISB releases a link for the n^{th} OSB due to $q_n^i < LT$, there are actually (REQ_NUM$_n$ $-$ 1)[2] or (REQ_NUM$_n$ $-$ 2) [3] link requests are expecting available links for the n^{th} OSB. Hence, the i^{th} ISB should decrease REQ_NUM$_n$ by either 1 or 2. However, the i^{th} ISB does not hold REQ_n token in current Rsv_Slot. The i^{th} ISB has to record this pending reduction of link requests and waits for receiving REQ_n token to modify REQ_NUM$_n$.

5.4.2 Operations upon receiving REQ_j Token

The operations for the received REQ_j token is very similar as that in REQ-QOBDLR algorithm. But, due to the operations for the received REL_j token in several Rsv_slot(s) before, the i^{th} ISB may need to first update REQ_NUM$_j$ with the pending increment/decrement of link requests for the j^{th} OSB. Hence, REQ_NUM$_j$ reflects the real number of link requests issued for the j^{th} OSB. After that, the i^{th} ISB follows the same operations as we presented in REQ-QOBDLR algorithm.

5.4.3 Remarks on REQREL-QOBDLR Algorithm

In a Ring_Cycle (i.e. $= K * Rsv\_Slot$), an ISB has two opportunities to evaluate its traffic load and to modify its link reservation for a specific OSB. Compared with REQ-QOBDLR algorithm, REQREL-QOBDLR algorithm is able to quickly adjust link resource allocation to adapt to the input traffic.

But, the efficiency of REQREL-QOBDLR algorithm is subject to the values of HT and LT. For example, if LT is given a large value but traffic load is not heavy, then every ISB will reduce its link reservation even though each ISB may have enough

[2]If the i^{th} ISB has not sent a link request for the n^{th} OSB, then (REQ_NUM$_n$ $-$ 1) link requests are demanding available links after the i^{th} ISB releases a link.

[3]If the i^{th} ISB has issued a link request for the n^{th} OSB, then (REQ_NUM$_n$ $-$ 2) link requests are demanding available links after the i^{th} ISB releases a link.

cells to be delivered (i.e. $q_j^i > r_j^i$). If so, released links are not going to be occupied by any ISB so that link resources are wasted. It will cause performance degradation. How to select HT and LT will be addressed in next section.

5.5 Analysis of QOBDLR Algorithms

5.5.1 Algorithm Complexity

Both REQ-QOBDLR algorithm and REQREL-QOBDLR algorithm are distributed link reservation schemes. In every Rsv_Slot, an ISB modifies its link reservation for only two OSBs which are identified by the received pair of REQ token and REL token. Arbitration on "borrowing" and/or "lending" a link to a specific OSB is based on the queue occupancy of two related GVOQs. Since an ISB modifies its link reservation according to its local available information, arbitration does not need to undergo multiple iterations and complexity is only $O(1)$.

5.5.2 The choice of HT and LT

In QOBDLR algorithms, the high threshold HT and the low threshold LT are predefined system parameters and are consistent after their initialization. In every Rsv_Slot, each ISB evaluates its queue occupancy of a GVOQ with HT and LT to decide whether to increase/decrease its link reservation for the related OSB. To select appropriate values of HT and LT is very important for QOBDLR algorithms to achieve a **fair and fast** link resource allocation among ISBs.

Notice that, ISBs tolerate two contentions when making link reservation for the targeted OSBs : (1) K GVOQs in a same ISB contend for M links at the ISB-ATMCSF interface; (2) K ISBs contend for M links at every ATMCSF-OSB interface.

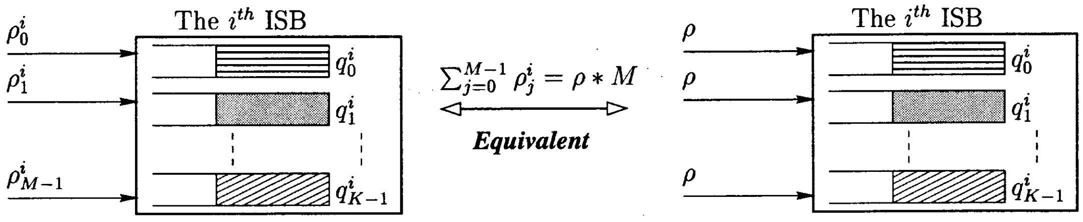


Figure 5.6 An ideal traffic scenario : the aggregated input load to the i^{th} ISB $(0 \leq i < K)$ uniformly targets K OSBs. Every ISB has the same traffic pattern.

As shown in Fig 5.6, if the aggregated input traffic to the i^{th} ISB $(0 \leq i < K)$ uniformly targets K OSBs, then the queue occupancy of every GVOQ in the i^{th} ISB will be the same. Hence, we have

$$q_0^i = q_1^i = \cdots = q_{(K-1)}^i = \frac{\rho_0^i + \rho_1^i + \rho_{(M-1)}^i}{K} = \frac{1}{K}\sum_{j=0}^{M-1} \rho_j^i \tag{5.1}$$

where, ρ_j^i $(0 \leq j < K)$ is the normalized input load contributed by the j^{th} switch input of the i^{th} ISB, $0 \leq \rho_j^i \leq 1$ [4].

If every ISB sustains the same aggregated input load at any time, then each ISB will have the same queue occupancy of every GVOQ. We use q_{avg} to represent the queue occupancy of GVOQ under this uniformly balanced traffic scenario. To obtain q_{avg}, we use an equivalent traffic model in which $\rho \times M = \sum_{j=0}^{M-1} \rho_j^i$, i.e. we assume that every switch input contributes the same input load ρ, then q_{avg} is :

$$q_{avg} = \frac{1}{K}\sum_{j=0}^{M-1} \rho_j^i = \frac{\rho \times M}{K} = \rho \times Fair \tag{5.2}$$

In above ideal case, ISBs do not need to borrow/lend links from each other because traffic pattern in every ISB is exactly the same. The M links at every ATMCSF-OSB interface will be evenly allocated to every ISB:

[4] ρ_j^i is a normalized input load which is formulated in detail in Section 5.6.1.

$$r_j^i = \frac{M}{K} = Fair; \qquad 0 \leq i,\ j\ < K \tag{5.3}$$

However, in real life, input load to an ISB is dynamically changed and different from each other's. In order to be **fair** for every ISB, the criteria to choose HT and LT should be :

$$\begin{aligned} \text{Lower Bound of HT :} \qquad & HT \geq q_{avg}; \\ \text{Upper Bound of LT :} \qquad & LT \leq q_{avg}; \end{aligned} \tag{5.4}$$

If the queue occupancy of a GVOQ in an ISB is larger than HT, the ISB can ask for an extra link to the related OSB because its traffic load to the OSB is heavier than the normal load q_{avg}. If the queue occupancy of a GVOQ in an ISB is less than LT, the ISB will release a link if other ISBs have link requests. Hence, QOBDLR algorithms provide a **fair** resource allocation among ISBs.

The values of HT and LT may have multiple choices. Table 5.1 shows an example which we apply to determine the values of HT and LT for Switch III if it is an 256x256 switch constructed by 8 ISBs and 8 OSBs, i.e. N = 256, K = 8, m = M = 32. Table 5.1 lists the possible choices of HT and LT based on different input load ρ. HT and LT should be suitable to handle most of the possible traffic loading. Since the input traffic loaded to a switch input is usually more than 50% (i.e. $\rho \geq 0.5$), we choose that LT = 2. When the traffic load is less than 50% (i.e. $\rho < 0.5$), the M links at an ATMCSF-OSB interface are most likely to be sufficient to support cell delivery of all ISBs. To determine HT, the joint set of values of HT to satisfy all possible traffic load is : HT $\in$ [4, ∞). If HT is very large, a link request will be activated much slowly because the queue occupancy of a GVOQ can not easily exceed HT. Therefore, an ISB may not be able to increase its link reservation timely to adapt to the increasing traffic. With this concern, we select HT as 4 above which an ISB starts requesting additional link. Hence, link reservation can adapt to the traffic quickly.

Table 5.1 The possible choices of HT and LT for an 256x256 switch consisting of 8 ISBs and 8 OSBs, i.e. $N = 256$, $K = 8$, $m = M = 32$. We select HT = 4, and LT = 2.

ρ	$q_{avg} = \frac{\rho * M}{K}$	LT	HT
1.0	4.0	$\leq$ 4.0 (i.e. 1,2,3,4)	$\geq$ 4.0 (i.e. 4,5, $\cdots$)
0.9	3.6	$\leq$ 3.6 (i.e. 1,2,3)	$\geq$ 3.6 (i.e. 4,5, $\cdots$)
0.8	3.2	$\leq$ 3.2 (i.e. 1,2,3)	$\geq$ 3.2 (i.e. 4,5, $\cdots$)
0.7	2.8	$\leq$ 2.8 (i.e. 1,2)	$\geq$ 2.8 (i.e. 3,4, $\cdots$)
0.6	2.4	$\leq$ 2.4 (i.e. 1,2)	$\geq$ 2.4 (i.e. 3,4, $\cdots$)
0.5	2.0	$\leq$ 2.0 (i.e. 1,2)	$\geq$ 2.0 (i.e. 2,3, $\cdots$)
0.4	1.6	$\leq$ 1.6 (i.e. 1)	$\geq$ 1.6 (i.e. 2,3, $\cdots$)
0.3	1.2	$\leq$ 1.2 (i.e. 1)	$\geq$ 1.2 (i.e. 2,3, $\cdots$)
0.2	0.8	$\leq$ 0.8 (i.e. 1)	$\geq$ 0.8 (i.e. 1,2, $\cdots$)
0.1	0.4	$\leq$ 0.4 (i.e. 1)	$\geq$ 0.4 (i.e. 1,2, $\cdots$)

It is worth to mention that, a theoretical work about the optimal choice of HT and LT, rather than the upper/lower bounds of the two thresholds, may be needed for different input traffic patterns. This would be our future work.

5.5.3 Scalability of QOBDLR Algorithms

To cooperate the modular switch architecture to achieve a good performance, QOBDLR algorithms should be scalable as well. The scalability of QOBDLR algorithms can be investigated from two aspects — *simplicity* and *efficiency*.

In section 5.5.1, we discussed that QOBDLR algorithms sustain a very low arbitration complexity of $O(1)$. Hence, QOBDLR algorithms will be able to afford switch growth without increasing arbitration complexity.

On the other hand, QOBDLR algorithms should be efficient to provide a fast and fair link resource allocation. One of the main factors to judge the efficiency of link reservation is D_{grant}, which is the average latency for a link request to be granted by a released link. For example, if a link request issued by an ISB has to travel the whole ring to find an available link at the fastest ISB, then the ISB will suffer a long delay to obtain its desired link. It will demote the efficiency of QOBDLR algorithms.

3.5.3.1 D_{grant} in REQ-QOBDLR Algorithm : In REQ-QOBDLR algorithm, an ISB may issue a link request for a specific OSB upon receiving the related REQ token. Fig 5.7 depicts an example in which ISB 0 is generating a new link request for the n^{th} OSB ($0 \leq n < K$) when it receives REQ_n token in Rsv_Slot 0.

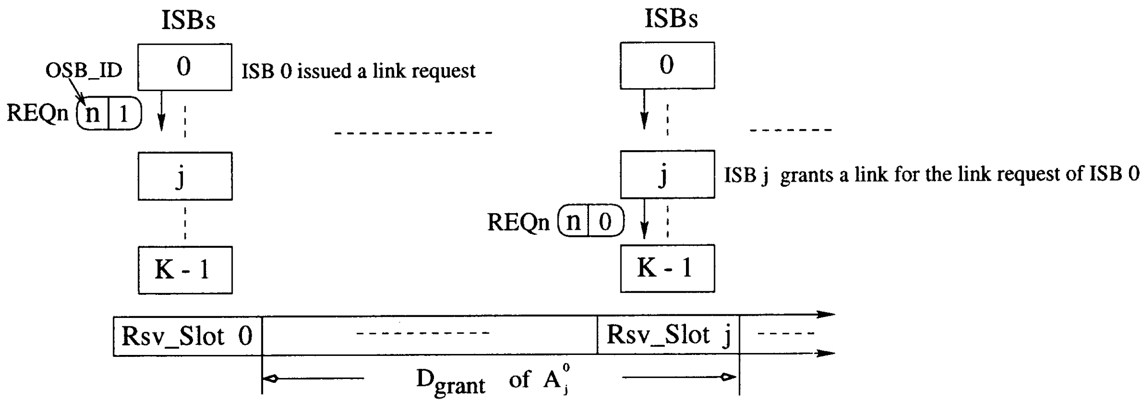


Figure 5.7 D_{grant} in REQ-QOBDLR Algorithm

We assume that in a Ring_Cycle (i.e. $K * Rsv\_Slot$), there is at least one ISB on the ring who can grant a link for the link request of ISB 0. Otherwise, ISB 0 is destined not to be able to obtain its desired link because other ISBs do not have available links to be released. If it is the j^{th} ISB that eventually releases a link to satisfy the link request of ISB 0, the latency for the link request to be granted by an available link is j Rsv_Slot(s). Statistically, we have the average latency D_{grant} in terms of the number of Rsv_Slot(s) as follows :

$$D_{grant} = \sum_{j=1}^{K-1} j \, P_r(A_j^0) \qquad (Rsv\_Slots) \tag{5.5}$$

where A_j^i indicates the event that a link request of the i^{th} ISB finds an available link in the j^{th} ISB. $P_r(A_j^i)$ is the probability that A_j^i happens.

We first derive D_{grant} under uniform traffic in which input load injected into an ISB uniformly targets K OSBs. In this scenario, every ISB has the same probability p $(0 \leq p \leq 1)$ to grant a link request for a certain OSB. Since a token passes ISBs one by one and only visits an ISB in a Rsv_Slot, we have :

$$P_r(A_j^0) = (1-p)^{j-1} p \tag{5.6}$$

therefore, D_{grant} will be :

$$\begin{aligned} D_{grant} &= \sum_{j=1}^{(K-1)} j P_r(A_j^0) = \sum_{j=1}^{(K-1)} j(1-p)^{j-1} \, p \\ &= \sum_{l=0}^{(K-2)} (l+1)(1-p)^l \, p \\ &= \underbrace{p \sum_{l=0}^{(K-2)} l(1-p)^l}_{PartA} + \underbrace{p \sum_{l=0}^{(K-2)} (1-p)^l}_{PartB} \end{aligned} \tag{5.7}$$

we can derive $PartA$ and $PartB$ of Eq. 5.7 individually :

$$\begin{aligned} Part\ A &= p \sum_{l=0}^{(K-2)} l(1-p)^l \\ &\overset{if K \to \infty}{=} p \times [(1-p) + 2(1-p)^2 + 3(1-p)^3 + \cdots] \\ &= p \times (1-p)[1-(1-p)]^{-2} \\ &= \frac{1-p}{p} \end{aligned} \tag{5.8}$$

$$\begin{aligned}
Part\ B &= p \sum_{l=0}^{(K-2)} (1-p)^l \\
&\overset{if K \to \infty}{=} p \times [1 + (1-p) + (1-p)^2 + (1-p)^3 + \cdots] \\
&= p \times [1 - (1-p)]^{-1} \\
&= 1 \qquad (5.9)
\end{aligned}$$

According to Eq. 5.7, Eq. 5.8 and Eq. 5.9, we eventually have :

$$\begin{aligned}
D_{grant} &= Part\ A + Part\ B \\
&\overset{if K \to \infty}{=} \frac{1-p}{p} + 1 \\
&= \frac{1}{p} \quad (\ \text{Rsv\_Slot(s)}\ ) \qquad (5.10)
\end{aligned}$$

Eq. 5.10 explains that, under uniform traffic, D_{grant} is not effected by K but is determined by p. It implies that large switch size (i.e. large K) will not incur an increasing delay for a link request to meet an available link.

In Eq. 5.10, p is the probability that an ISB is able to grant a link for a link request. If $p = 1$, then D_{grant} is 1, i.e. a link request will be satisfied by the next neighboring ISB. But, if $p \ll 1$, then $D_{grant} \gg 1$, i.e. it will take several Rsv_Slots for a link request to encounter an available link. Since an ISB needs to check its related queue occupancy to arbitrate whether to release a link for a link request, p can be expressed as follows (for $0 \leq i, n < K$) :

$$p \ = \ P_r(q_n^i < LT) \ + \ P_r(q_n^i \geq LT)\, P_r(r_n^i > Fair) \qquad (5.11)$$

Eq. 5.11 is engaged with a queueing problem modeled in Fig 5.8. The input traffic from every switch input is an ON-OFF traffic stream multiplexing several VC connections. Moreover, the input load injected into every GVOQ is in fact an

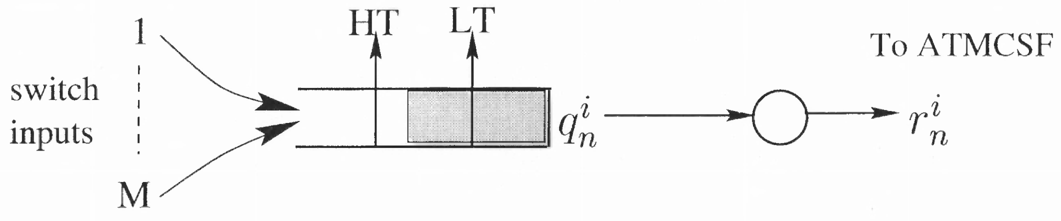


Figure 5.8 How to get $P_r(q_n^i > c)$, where c is a constant value.

aggregation of multiple ON-OFF streams carrying multicast cells. The outgoing traffic rate is identified by r_n^i which can be interpreted as the dynamic service rate. The queueing model illustrated in Fig 5.8 is similar to the traffic model consisting of batch arrival and batch departure. However, in our model, theoretical analysis is too complicated to achieve because departure process is correlated with arrival process and queue length.

Independent batch arrival/batch departure traffic model, in general, is a difficult analyzing problem due to multiplexing of typically a large number of connections and burstiness of individual cell streams at possibly different time scales [61]. Sohraby et al. presented several solutions based on M/G/1-Type Markov Chains in [61] [62] [63]. However, their approaches are very computation-consuming. Moreover, those solutions are not applicable in our model where arrival and departure and correlated. Hence, a further effort to achieve a comprehensive theoretical analysis is still our ongoing work.

But, we may do some approximation and intuitively interpret the meaning beyond the equation. From Eq. 5.11, we derive an upper bound of D_{grant}.

$$D_{grant} = \frac{1}{p} = \frac{1}{P_r(q_n^i < LT) + P_r(q_n^i \geq LT)\, P_r(r_n^i > Fair)}$$

$$\leq \frac{1}{P_r(q_n^i < LT)} \tag{5.12}$$

It is observed from Eq. 5.12, if LT reduces, $P_r(q_n^i < LT)$ will become smaller and D_{grant} will increase. It makes sense in a way that the queue occupancy of a GVOQ may not easily go below LT, if LT is small. Thus, an ISB will not be able to grant a link for a link request. It will incur a longer delay for a link request to be satisfied.

Under non-uniform burst traffic, situation may become more complicated. If we use p_i $(0 \leq i < K)$ to indicate the probability that the i^{th} ISB is able to release a link for a link request to the n^{th} OSB, then we may have :

$$p_i = P_r(q_n^i < LT) + P_r(q_n^i \geq LT)\, P_r(r_n^i > Fair) \tag{5.13}$$

Moreover, with non-uniform traffic, we have :

$$p_0 \neq p_1 \neq p_2 \cdots \neq p_{K-1} \tag{5.14}$$

hence, D_{grant} will be expressed as :

$$\begin{aligned} D_{grant} &= \sum_{j=1}^{K-1} j * P_r(A_j^0) \\ &= \sum_{j=1}^{K-1} j * [(1-p_1)(1-p_2)\cdots(1-p_{j-1})p_j] \end{aligned} \tag{5.15}$$

Due to the difficulty to analyze the model as shown in Fig 5.8, we have not been able to obtain a close-form expression of Eq. 5.15. But, we would like to present following discussions to intuitively evaluate D_{grant} . First, a REQ token passes ISBs one by one in a round robin manner, so that the nearest ISB has the highest responsibility to release a link for a link request. Intuitively, an ISB who had issued a link request for a specific OSB should obtains the desired link from its down-stream neighboring ISBs in several Rsv_Cycle(s). Second, if switch size is very large (i.e. K is very large), traffic pattern in each ISB may change during the period that a link request is seeking for an available link. Therefore, an ISB who previously does not want to grant a link may be able to release a link when the link request token stops at its block. In our performance simulations, it rarely happens that a link request needs to traverse a complete ring to obtain an available link.

3.5.3.2 D_{grant} in REQREL-QOBDLR Algorithm : In REQREL-QOBDLR algorithm, link request operation and link release operation are more independent than those in REQ-QOBDLR algorithm. Even though there is no link request issued by any ISB, an ISB may release a link if its queue occupancy is less than LT. On the other hand, an ISB who has issued a link request to a certain OSB may be able to catch an available link immediately without polling neighbor ISBs one by one because there may already be released links circulating on the ring. Hence, in REQREL-QOBDLR algorithm, D_{grant} is smaller than that in REQ-QOBDLR algorithm. However, the delay variation in REQREL-QOBDLR may be larger than that in REQ-QOBDLR algorithm in which a link release is only stimulated by an explicit link request.

5.6 Performance Evaluation

In this section, we evaluate the performance of Switch III and compare it with Switch I, Switch II and the OQ switch under same traffic scenarios.

5.6.1 Traffic Model

The switch performance is investigated under both *uniform* and *non-uniform* traffic. As shown in Fig 5.9, cells coming from different VCs are multiplexed in bursts which are interleaved to contribute as the arrival traffic at every switch input. We employ the ON(active)/OFF(idle) model to describe the burst-idle process. The back-to-back cells in an ON duration belong to the same VC so that they have same destinations. No cells arrive in idle period.

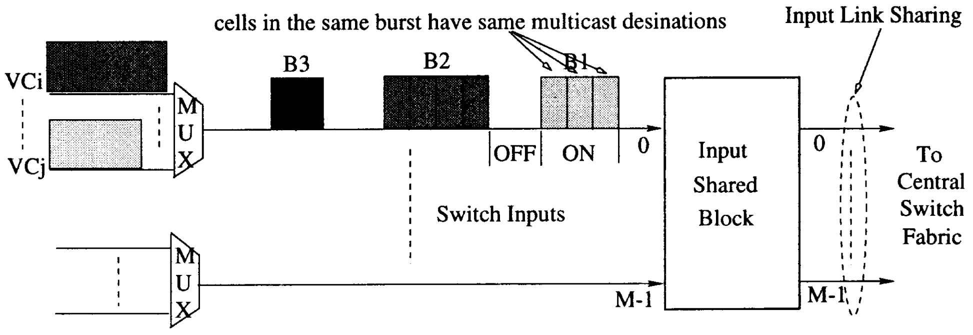


Figure 5.9 Traffic Model : Multicast Burst Traffic

Under **uniform traffic**, the aggregated input load to every ISB is the same and uniformly targets all switch outputs. We set a small value of MBS which is the maximum burst size (i.e. the number of cells in an ON duration). Cells' destinations are uniformly distributed among N switch outputs.

On the contrary, **non-uniform traffic** is featured by "hot spot" phenomenon : cells accommodated in an ISB prefer to go to some switch outputs, but rarely go to other output ports. Three scenarios are likely to build a non-uniform burst traffic in an ISB : (1) If MBS (i.e. maximum burst size) is very large, cells arriving in an ON period will keep targeting the same destinations for a long time. Input traffic is not uniformly destined to N switch outputs in this time duration. (2) If bursts are correlated with each other, i.e. cells arriving in successive ON periods keep going to the same destinations. Even though MBS may be small, cells accumulated in several ON periods will generate a non-uniform traffic in an ISB. (3) In an extreme case, an

ISB only has cells to go to a specific OSB, but has no cells to other OSBs. Different ISB is dedicated to different OSB. This is so called "1 ISB $\rightarrow$ 1 OSB hot spot" traffic.

In this dissertation, VBR source is used to generate the ON-OFF traffic. Following are traffic parameters : MBS is the maximum burst size; PCR is the peak cell rate (i.e. the number of cells per second) which satisfies that $PCR \leq LCR$; LCR is the line cell rate which approximates $\frac{155,520,000}{53*8} = 366,793(cells/sec)$ for an OC-3 link; ACR is the average cell rate. We define F_{out} as the fanout of a cell. F_{out} has a uniform distribution from 0 to C_{max}. The average fanout load $F = (C_{max} + 1)/2$, where C_{max} is the maximum copies allowed for a multicast cell. The effective input load is defined as $\rho = (ACR \times F)/LCR$, $0 \leq \rho \leq 1$.

5.6.2 Switch Performance

The switch performance is evaluated through simulations by using OPNET/MIL3 simulation platform [65]. For our switch designs, we apply an 256x256 (N = 256) switch consisting of 8 ISBs and 8 OSBs (K = 8). Each ISB/OSB is of size 32x32 (m = M = 32). **For Switch II** using RR+POLR, we assume that the link reservation is carried out with a rate of $R = \frac{Cell\_Slot}{Rsv\_Slot} = 4$, i.e. a token can pass four ISBs in a cell slot. As we discussed in chapter 4, faster link reservation rate (i.e. $R = \frac{Cell\_Slot}{Rsv\_Slot} > 4$) results in better performance; and vice verser. Note that Switch II will be the same as Switch I if $R = 1$. **For Switch III** with QOBDLR algorithms, we assume that the link reservation is performed at the slowest rate, i.e. $R = \frac{Cell\_Slot}{Rsv\_Slot} = 1$. Hence, a token only goes through one ISB in a cell slot. Switch III with any faster link reservation rate (i.e. $R > 1$) will obtain a better performance than what we simulated here. **For the OQ switch**, we assume that the OQ switch can support N times speedup. Therefore, cells arriving at switch inputs can be transmitted to the related output queues in a cell slot. Output buffers are infinite so that no cells could be lost. The OQ switch proved to be able to achieve the best performance under any traffic pattern.

We investigate the switch performance under both uniform and non-uniform traffic. Following performance statistics are estimated :

- **Throughput :** switch throughput which is statistically measured on N switch outputs ;

- D_{E-to-E} : the average end-to-end cell delay in terms of the number of cell slots. D_{E-to-E} is the latency for a cell going through the switch. Delay jitter is measured by (Min, Max) of D_{E-to-E}.

- D_{ISB} : the average cell delay in ISBs in terms of the number of cell slots. We assume that ATMCSF forwards cells from input shared links to output shared links in a cell slot, hence, ATMCSF does not cause any cell delay. D_{E-to-E} is resulted from two parts : the *cell delay in ISBs (i.e.* D_{ISB}), and the *cell delay in OSBs.* The vector of (Min, Max) of D_{ISB} is the minimum delay and the maximum delay incurred in ISBs.

- S_{OSB} : the average occupancy of an OSB measured by the number of cells accommodated in the OSB. Since each OSB consists of M output queues, S_{OSB} indicates the total number of cells waiting in an OSB. The vector of (Min, Max) of S_{out} estimates the lower bound and the upper bound of the occupancy of an OSB.

Table 5.2 compares the switch performance under **uniform traffic** with different input load ρ. Multicast uniform traffic is applied. For Switch III, we select the values of HT and LT as 4 and 2 as what we discussed in previous section.

Table 5.2 Switch III : performance comparison under uniform multicast traffic with different input load ρ. The observed performance statistics are : (1) throughput; (2) average end-to-end cell delay and delay jitter (D_{E-to-E}, (Min, Max)); (3) average cell delay in ISB and delay jitter (D_{ISB}, (Min, Max)); (4) average occupancy of OSB (S_{OSB} and (Min, Max)).

		Uniform, Multicast, Burst Traffic (F = 4 , Cmax = 8, MBS = 32)				
		OQ Switch *(256x256 Switch)*	***Switch I*** ***GVOQ RR***	***Switch II*** ***RR+POLR*** $R = \frac{Cell\_Slot}{Rsv\_Slot} = 4$	***Switch III*** *(HT = 4 , LT = 2)* ***REQ-QOBDLR*** $R = \frac{Cell\_Slot}{Rsv\_Slot} = 1$	***REQREL-QOBDLR*** $R = \frac{Cell\_Slot}{Rsv\_Slot} = 1$
ρ = **0.99**	Throughput	0.99	0.98898	0.98958	0.98930	0.98950
	$\mathbf{D}_{\text{E-to-E}}$ (Min, Max)	24.6 (1.0 , 153)	34.3 (5.7 , 203)	30 (2.8 , 184)	32.5 (2.0 , 190)	31 (2.0 , 184)
	$\mathbf{D}_{\text{ISB}}$ (Min, Max)	N/A	7.6 (1.0 , 18.0)	5.8 (1.0 , 16.4)	6.4 (1.0 , 14.2)	6.0 (1.0 , 13.9)
	$\mathbf{S}_{\text{OSB}}$ (Min, Max)	N/A	764 (586 , 965)	816 (582 , 998)	788 (572, 964)	804 (574 , 1040)
ρ = **0.90**	Throughput	0.90	0.89917	0.89938	0.89932	0.89948
	$\mathbf{D}_{\text{E-to-E}}$ (Min, Max)	9.5 (1.0 , 95)	17.2 (5.6 , 146)	12.7 (2.0 , 108)	13.4 (1.5 , 117)	12.0 (1.5 , 112)
	$\mathbf{D}_{\text{ISB}}$ (Min, Max)	N/A	5.9 (1.0 , 17)	4.6 (1.0 , 14)	4.7 (1.0 , 11.8)	4.2 (1.0 , 11.6)
	$\mathbf{S}_{\text{OSB}}$ (Min, Max)	N/A	364 (201 , 483)	382 (208 , 476)	372 (172 , 465)	388 (174, 465)
ρ = **0.70**	Throughput	0.70	0.69941	0.69983	0.69988	0.69984
	$\mathbf{D}_{\text{E-to-E}}$ (Min, Max)	3.6 (1.0 , 38)	8.1 (2.0 , 76)	5.4 (2.5 , 60)	4.6 (2.0 , 63)	4.8 (2.0 , 66)
	$\mathbf{D}_{\text{ISB}}$ (Min, Max)	N/A	4.5 (1.0 , 12.8)	2.8 (1.0 , 11.4)	1.3 (1.0 , 10.0)	1.3 (1.0 , 10.2)
	$\mathbf{S}_{\text{OSB}}$ (Min, Max)	N/A	86.5 (60 , 117)	82 (56 , 112)	75 (52 , 102)	78 (54, 110)
ρ = **0.50**	Throughput	0.50	0.49967	0.49991	0.49994	0.49991
	$\mathbf{D}_{\text{E-to-E}}$ (Min, Max)	2.0 (1.0 , 22)	3.6 (1.0 , 47)	2.5 (1.0 , 35)	2.3 (1.0 , 33)	2.4 (1.0 , 3 3)
	$\mathbf{D}_{\text{ISB}}$ (Min, Max)	N/A	2.1 (1.0 , 11)	1.35 (1.0 , 11)	1.05 (1.0 , 9.5)	1.0 (1.0 , 9.6)
	$\mathbf{S}_{\text{OSB}}$ (Min, Max)	N/A	38.3 (22.3, 57)	35 (21 , 52)	29.6 (18.8 , 48.8)	30 (18.4 , 46)

In chapter 3 and chapter 4, we have shown that Switch I and Switch II are able to provide good performance under uniform traffic. Comparison in Table 5.2 indicates that, Switch III, as well as Switch I and Switch II, can achieve a comparable performance as the OQ switch under uniform traffic. On **throughput** performance, the OQ switch always leads to the maximized throughput ρ, and our switches have less than 0.5% throughput degradation. In general, the **end-to-end cell delay** D_{E-to-E} increases with input load ρ. Compared with the lower bound of D_{E-to-E} achieved in the OQ switch, D_{E-to-E} in our switch designs causes 2$\sim$15 more cell slots. Longer cell delay is due to lower throughput.

It is also observed from Table 5.2 that, Switch II and Switch III obtain a little bit better performance than Switch I because the former ones utilize link reservation to avoid starvation of OSBs. However, under uniform traffic, link reservation is not necessary so that Switch I still can obtain a similarly good performance as the other two switches. When input load is heavy such as $\rho = 0.99$ or 0.90, Switch II results in a little bit better performance than Switch III but it happens with the condition that the link reservation rate in Switch II ($R = \frac{Cell\_Slot}{Rsv\_Slot} = 4$) is faster than that in Switch III ($R = \frac{Cell\_Slot}{Rsv\_Slot} = 1$). If link reservation performs at the same rate of $R = 1$, Switch II will yield the same performance as Switch I so that Switch III can defeat Switch II.

The two QOBDLR algorithms proposed for Switch III are very competitive to each other. REQREL-QOBDLR algorithm exceeds REQ-QOBDLR algorithm with better performance when input load ρ is heavy. The reason for that is, dynamic link reservation achieved by REQREL-QOBDLR is faster than that in REQ-QOBDLR algorithm. But, when input load ρ is less than 0.7, REQ-QOBDLR algorithm outperforms REQREL-QOBDLR algorithm because REQREL-QOBDLR intends to release more links which may not be utilized by any ISBs. In general, both REQ-QOBDLR and REQREL-QOBDLR algorithms are capable of providing good performance for uniform traffic.

Moreover, Table 5.2 shows that D_{E-to-E} in our designs is mainly due to the latency in OSBs (i.e. $D_{E-to-E} - D_{ISB}$) rather than the delay in ISBs (i.e. D_{ISB}). In addition, S_{OSB} indicates that cells are forwarded to OSBs in a fast manner because most of the cells are backlogged in OSBs. This is a good feature of our switch designs because OSBs may be able to incorporate per VC queueing with appropriate cell schedulers to provide QoS guarantees as the OQ switch does. It is the subject of our ongoing work.

Table 5.3 compares the switch performance under **non-uniform traffic**. Fig 5.10 $\sim$ Fig 5.13 illustrate the performance of throughput, D_{E-to-E}, D_{ISB} and S_{OSB} individually. We apply unicast "1 ISB $\rightarrow$ 1 OSB HotSpot Traffic" : the input load injected into an ISB only targets a specific OSB, but no cells go to other OSBs.

Performance comparison shows that Switch I fails to offer a good performance for non-uniform traffic. The reason is that, in Switch I, an ISB is only allowed to deliver cells to its matched OSB according to the one-to-one mapping in a cell slot. If an ISB does not have cells to go to its assigned OSB, other ISBs do not have authority to send cells to the starved OSB. Under "1 ISB $\rightarrow$ 1 OSB HotSpot Traffic", an ISB only has cells to be delivered in 1 out of every K cell slots. Switch I suffers a significant performance degradation and only approaches 13% throughput even though input load is 99%. Since more and more cells are blocked in ISBs, the cell delay in ISBs (i.e. D_{ISB}) continues to increase. It, therefore, causes an ever-increasing end-to-end cell delay (i.e. D_{E-to-E}).

Table 5.3 Switch III : performance comparison under unicast "1 ISB $\rightarrow$ 1 OSB HotSpot" traffic. The observed performance statistics are : (1) throughput; (2) average end-to-end cell delay and delay jitter (D_{E-to-E}, (Min, Max)); (3) average cell delay in ISB and delay jitter (D_{ISB}, (Min, Max)); (4) average occupancy of OSB (S_{OSB} and (Min, Max)).

		1 ISB - 1 OSB HotSpot, Unicast, Burst Traffic (F = Fout = 1 , MBS = 20, HotSpotLen = 20*5)				
		OQ Switch *(256x256 Switch)*	***Switch I*** ***GVOQ RR***	***Switch II*** ***RR+POLR*** $R = \frac{Cell\_Slot}{Rsv\_Slot} = 4$	***Switch III*** *(HT = 4, LT = 2)*	
					REQ-QOBDLR $R = \frac{Cell\_Slot}{Rsv\_Slot} = 1$	***REQREL-QOBDLR*** $R = \frac{Cell\_Slot}{Rsv\_Slot} = 1$
ρ = **0.99**	Throughput	0.99	0.12949	0.51807	0.98902	0.98933
	D_{E-to-E} (Min, Max)	88 (1.0 , 335)	2260 (32.5 , 2454)	1260 (6.4 , 1388)	127 (2.5 , 425)	103 (2.5 , 378)
	D_{ISB} (Min, Max)	N/A	2197 (8.3 , 2386)	1198 (0.1 , 1309)	58.2 (1.0 , 224)	55.5 (1.0 , 159)
	S_{OSB} (Min, Max)	N/A	14.2 (5.5 , 26.8)	112 (59.0 , 156)	1788 (1586 , 2070)	1744 (1554 , 2038)
ρ = **0.90**	Throughput	0.90	0.12926	0.51805	0.89967	0.89971
	D_{E-to-E} (Min, Max)	33.8 (1.0 , 248)	1860 (30 , 2012)	875 (5.8 , 912)	72 (2.1 , 354)	56 (2.1 , 316)
	D_{ISB} (Min, Max)	N/A	1833 (7.0 , 1985)	445 (0.1 , 863)	34.0 (1.0 , 127)	31.5 (1.0 , 88.0)
	S_{OSB} (Min, Max)	N/A	11.5 (5.5 , 17.4)	105 (62.5 , 145)	1305 (1194 , 1510)	1238 (1162 , 1492)
ρ = **0.70**	Throughput	0.70	0.12883	0.50567	0.69995	0.69985
	D_{E-to-E} (Min, Max)	11.2 (1.0 , 112)	1730 (25.0 , 1887)	24.3 (5.4 , 156)	28.9 (2.05 , 197)	29.1 (2.0 , 168.5)
	D_{ISB} (Min, Max)	N/A	1705 (5.8 , 1791)	5.6 (0.1 , 18.6)	8.7 (1.0 , 92.3)	8.6 (1.0 , 67.1)
	S_{OSB} (Min, Max)	N/A	8.0 (6.0 , 12.4)	97 (70 , 129)	274 (208 , 349)	258.5 (191 , 338.5)
ρ = **0.50**	Throughput	0.50	0.12806	0.49925	0.49987	0.49981
	D_{E-to-E} (Min, Max)	5.0 (1.0 , 65)	1610 (21.5 , 1812)	13.5 (5.0 , 75)	18.5 (2.0 , 158.3)	18.5 (2.0 , 157.4)
	D_{ISB} (Min, Max)	N/A	1594 (4.0 , 1670)	2.4 (0.1 , 16.2)	3.6 (1.0 , 58.3)	3.6 (1.0, 45.7)
	S_{OSB} (Min, Max)	N/A	7.4 (6.0 , 9.6)	97.0 (70.5 , 130)	83.5 (51.5 , 124.5)	84 (51.0 , 127.5)

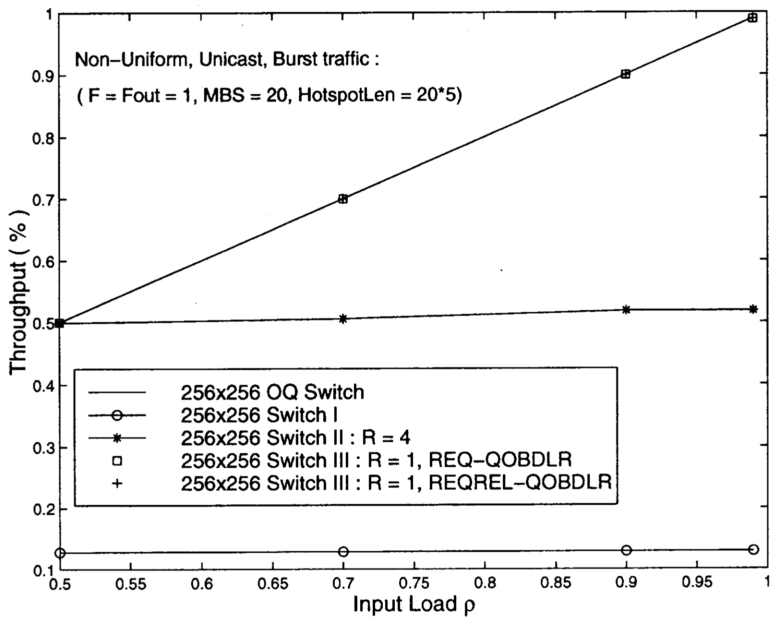


Figure 5.10 Switch III : throughput performance under unicast "1 ISB → 1 OSB HotSpot" traffic.

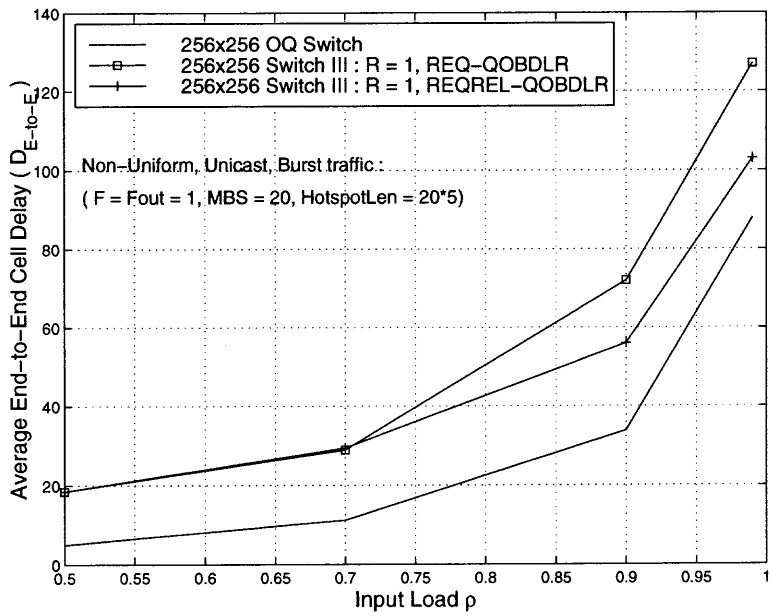


Figure 5.11 Switch III : average end-to-end cell delay (D_{E-to-E}) under unicast "1 ISB → 1 OSB HotSpot" traffic.

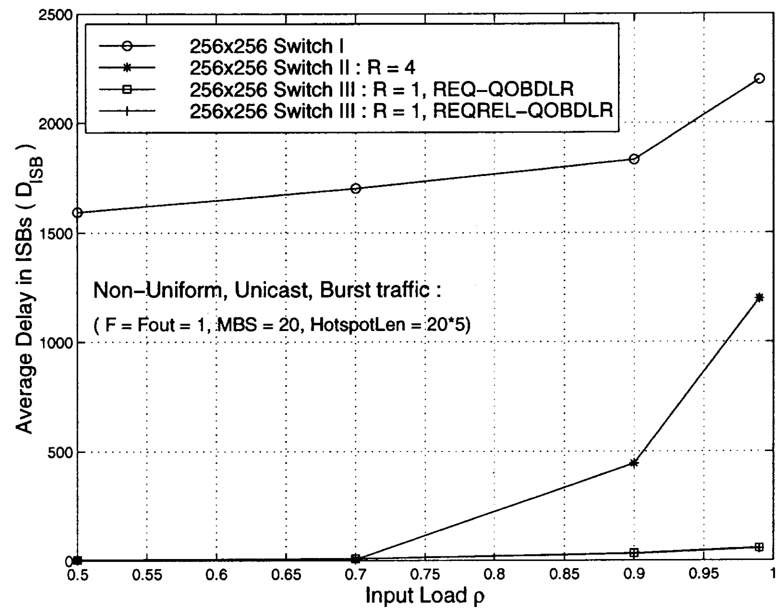


Figure 5.12 Switch III : average delay in ISBs (D_{ISB}) under unicast "1 ISB → 1 OSB HotSpot" traffic.

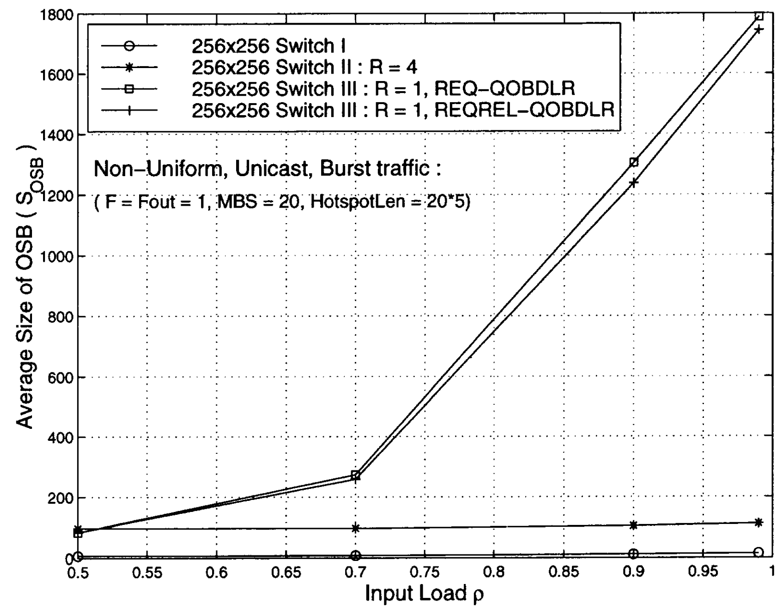


Figure 5.13 Switch III : average size of OSBs (S_{OSB}) under unicast "1 ISB → 1 OSB HotSpot" traffic.

Switch II will endure the same performance decline as Switch I if link reservation is performed at the slowest rate, i.e. $R = \frac{Cell\_Slot}{Rsv\_Slot} = 1$. But, Switch II can lead to an improved performance with any faster link reservation rate such as $R = 4$ (refer to Table 5.3). However, Switch II has a weakness that an ISB has to reset its link reservation vector in every cell slot. Hence, the performance of Switch II is mainly determined by the link reservation rate which would be a bottleneck for Switch II to achieve the high performance. For example, Switch II with $R = 4$, though achieving better performance than Switch I, can not approach to a comparable performance as the OQ switch.

Being an enhanced switch design, Switch III outperforms the other two switches and achieves a comparable performance to the OQ switch under non-uniform traffic. Switch III benefits from the dynamic link reservation schemes so that an ISB does not need to reset its link reservation vector in every cell slot. Even though the dynamic link reservation is operated at the slowest rate (i.e. $R = \frac{Cell\_Slot}{Rsv\_Slot} = 1$), Switch III using QOBDLR algorithms can adapt to the input traffic quickly and perform a fast and fair link resources allocation among ISBs. Fig 5.10 shows that Switch III leads to a very similar throughput as the OQ switch. Fig 5.11 indicates that Switch III causes no more than 30 cell slots longer delay of D_{E-to-E} if compared to the OQ switch. We also observed that most of the cells are forwarded to and buffered in OSBs, hence, D_{E-to-E} is mainly resulted from the cell delay in OSBs. As we mentioned before, it is a good feature of the proposed switch because OSBs, which look like the output queues in the OQ switch, are able to incorporate per VC queueing with appropriate cell schedulers to provide QoS guarantees.

Under multicast traffic, our switch designs can yield better performance than under unicast traffic, because ISBs can take the advantage of GVOQs to deliver multicast cells to ATMCSF. Thus, a faster cell forwarding can be gained when switches handle multicast input traffic. Table 4.2 and Table 4.3 have evaluated Switch I and Switch II under multicast " 1 ISB → 1 OSB HotSpot traffic". Here, we will not further examine Switch III under multicast traffic because Switch III has already proved to be able to achieve a good performance under unicast traffic as shown in Table 5.2. Obviously, performance of Switch III under multicast traffic will be even better.

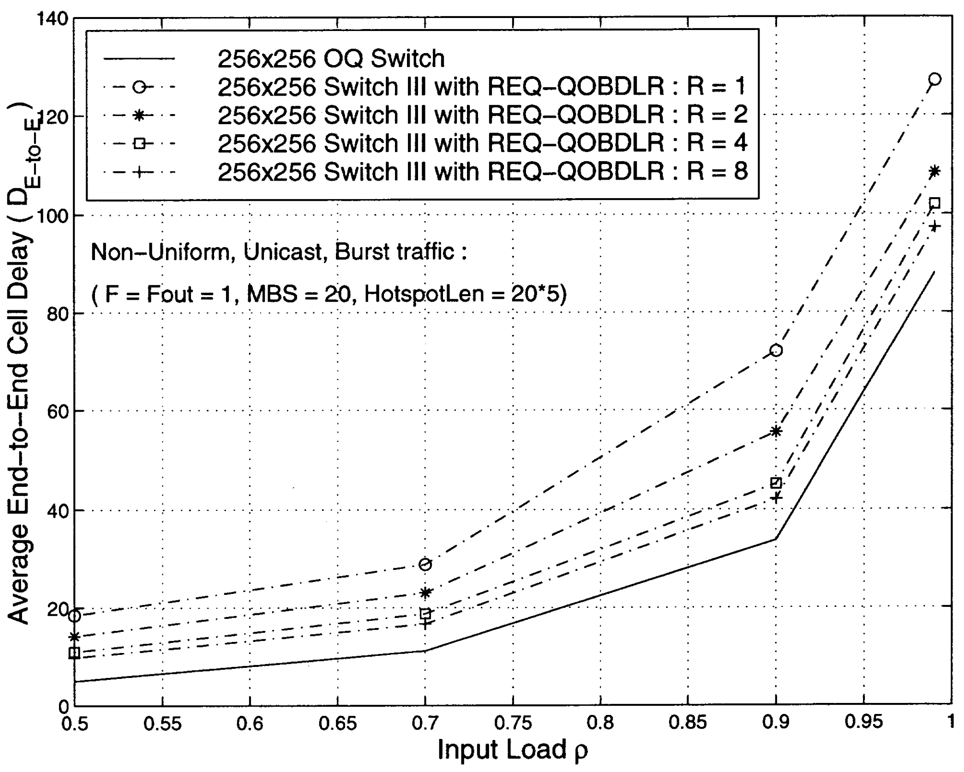


Figure 5.14 $D_{E\text{-}to\text{-}E}$ in Switch III using REQ-QOBDLR algorithm with different link reservation rate (i.e. $R = \frac{Cell\_Slot}{Rsv\_Slot}$) under unicast "1 ISB → 1 OSB HotSpot" traffic.

Moreover, we investigate the impact of the link reservation rate (i.e. $R = \frac{Cell\_Slot}{Rsv\_Slot}$) on the performance of Switch III. Table 5.3 has shown that Switch III with

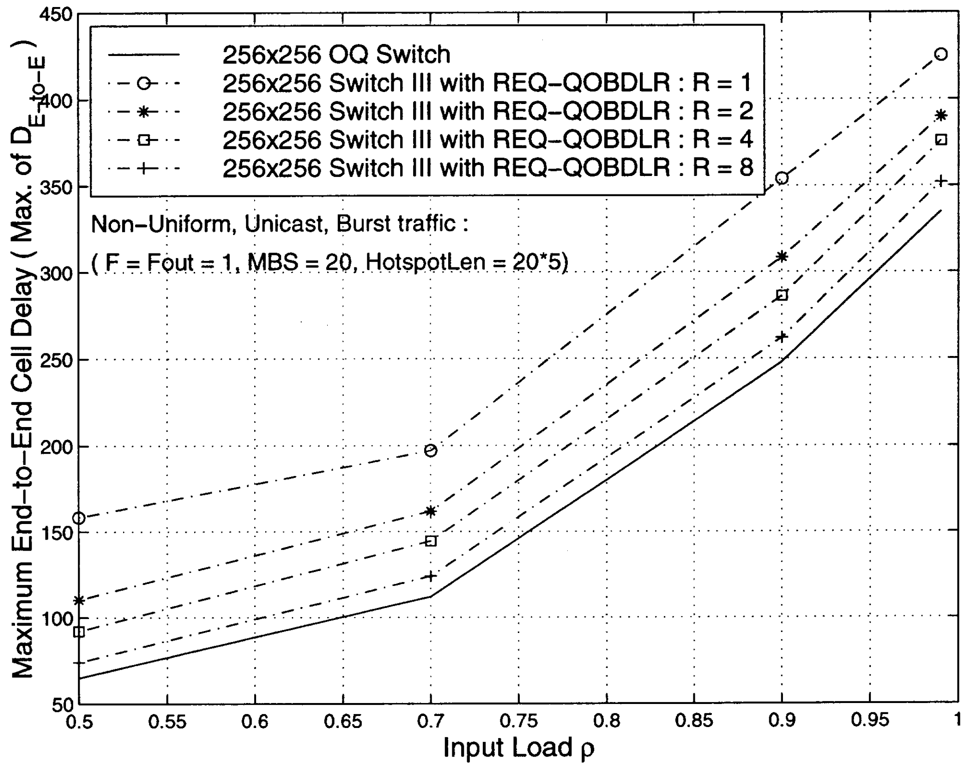


Figure 5.15 Max. of D_{E-to-E} in Switch III using REQ-QOBDLR algorithm with different link reservation rate (i.e. $R = \frac{Cell\_Slot}{Rsv\_Slot}$) under unicast "1 ISB → 1 OSB HotSpot" traffic.

$R = 1$ is capable of obtaining a high throughput which is comparable to that of the OQ switch. The choice of R does not affect the throughput performance significantly. But, Fig 5.14 and Fig 5.15 shows that, for Switch III using REQ-QOBDLR algorithm, D_{E-to-E} and delay variance (i.e. Max of D_{E-to-E}) will be reduced if R increases. The same conclusion can be drawn for REQREL-QOBDLR algorithm : the faster link reservation rate, the better performance.

In summary, Switch III exhibits the capability to pursue a high performance under both uniform traffic and non-uniform traffic. Compared to the OQ switch,

Switch III can be claimed as a competitive design in the sense that Switch III not only can achieve a comparable performance to the OQ switch but also can eliminate the N times speedup which is necessary in the OQ switch.

5.7 Conclusion

In this chapter, we present a novel switch design for scalable terabit multicast packet switches. The proposed switch enjoys a modular architecture consisting of ISBs, OSBs and a central switch fabric. Dual round robin rings provide a mechanism for ISBs to dynamically "borrow" and/or "lend" links from/to each other. The switch benefits from input and output link sharing so that no speedup is needed in the central switch fabric.

To resolve input and output contentions, cell delivery is based on link reservation in every ISB. We propose two Queue Occupancy Based Dynamic Link Reservation algorithms, both of them are able to provide a fast and fair link resource allocation among ISBs. QOBDLR is a distributed link reservation scheme in which an ISB can dynamically increase/decrease its link reservation for a specific OSB according to its local available information. Arbitration complexity is $O(1)$.

Performance evaluation demonstrates that Switch III can achieve a comparable performance to the OQ switch under any traffic pattern. But, our switch design can scale easily without requiring speedup, while the OQ switch supporting similar performance needs N times speedup (N is the switch size) which in large scale switches is impractical.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The aim of this dissertation is the design of a scalable, large-capacity, high performance core switch for broadband networks. The issues addressed for the switch design include multicasting, architecture scalability, and arbitration complexity. In this dissertation, we proposed three novel scalable terabit multicast packet switches — Switch I, Switch II and Switch III.

From an architectural point of view, all the proposed switches are characterized by a modular configuration using ISBs, OSBs and ATMCSF. Furthermore, all switches employ a novel co-operative input and output link sharing so that no speedup is necessary in the central switch fabric. Thus, the bottleneck in memory access rate and architecture expansion is avoided. Multicast function is achieved by cell splitting along with cell delivery. The novel scheme of grouped virtual output queue (GVOQ) provides a fast cell forwarding and simple cell scheduling, especially for multicast traffic. Because of the modular architecture, the proposed switches can easily scale to a large size and high capacity.

Instead of using a centralized scheduler to resolve input and output contentions, we proposed various distributed resource allocation algorithms for each switch design. In Switch I, two round robin scheduling algorithms — IVOQ RR and GVOQ RR, are presented. The arbitration complexity of IVOQ RR is in a range of $[O(1)$, $O(M)]$, while GVOQ RR sustains a low complexity of $O(1)$. Switch II applies a prioritized link reservation algorithm RR+POLR to eliminate the starvation of OSBs. This results in substantial improvement in switch performance especially for non-uniform traffic. For the enhanced Switch III, we proposed two dual round robin dynamic link reservation algorithms — REQ-QOBDLR and REQREL-QOBDLR.

A fast and fair link resource allocation among ISBs is achieved by "borrowing" and/or "lending" links from each other through REQ tokens and REL tokens. Both algorithms are distributed link reservation schemes in which every ISB, according to its local available information, can dynamically modify its own link reservation. As arbitration complexity is $O(1)$, scheduling complexity is not an obstacle any more for switch growing to a large scale.

Comparison studies on switch performance show that Switch I performs well for uniform traffic but it is not suitable for non-uniform traffic. Though Switch II yields an improved performance under non-uniform traffic, RR+POLR algorithm is not flexible enough to quickly adapt to the input traffic. Switch III benefits from dynamic link reservation which provides a fast and fair resource allocation. Hence, Switch III achieves a high performance as good as the OQ switch, while at the same time eliminating the N times speedup of central switch fabric required in the OQ switch. Thus, Switch III is a good choice for a scalable terabit multicast packet switch.

The following issues need to be further addressed for practical implementation of Switch III. First, the optimal choice of HT and LT may need to be investigated rather than the bounds of HT and LT. A more comprehensive theoretical work on REQ-QOBDLR and REQREL-QOBDLR algorithms is needed, and is ongoing work. In addition, a detailed study on QoS features of Switch III might be necessary, even though it appears that OSBs can incorporate per VC queueing with appropriate cell schedulers to provide QoS guarantee.

APPENDIX A

REQ-QOBDLR ALGORITHM

We describe REQ-QOBDLR algorithm in detail. Every ISB has following four vectors, $0 \leq i\,,\, j\, < K$:

• LK_RSV^i : link reservation vector in the i^{th} ISB. LK_RSV^i = $[r_0^i\,,\, r_1^i\,,\, \cdots,\, r_{(K-1)}^i]$, where r_j^i indicates how many links at the ATMCSF-OSB j interface are reserved by the i^{th} ISB, $0 \leq r_j^i \leq M$.

• Q^i : queue occupancy vector in the i^{th} ISB. Q^i = $[q_0^i\,,\, q_1^i\,,\, \cdots,\, q_{(K-1)}^i]$, where q_j^i is the queue occupancy of the j^{th} GVOQ in the i^{th} ISB, $q_j^i \geq 0$.

• LK_REQ^i : link request vector in the i^{th} ISB. LK_REQ^i = $[s_0^i\,,\, s_1^i\,,\, \cdots,\, s_{(K-1)}^i]$, where s_j^i indicates whether the i^{th} ISB is requesting a link to the j^{th} OSB, $s_j^i = 0$ or 1. If $s_j^i = 0$, the i^{th} ISB is not desiring any more link to the j^{th} OSB; if $s_j^i = 1$, the i^{th} ISB is asking for an extra link to the j^{th} OSB.

• LK_REL^i : link release vector in the i^{th} ISB. LK_REL^i = $[l_0^i\,,\, l_1^i\,,\, \cdots,\, l_{(K-1)}^i]$, where l_j^i indicates whether the i^{th} ISB has a pending released link for the j^{th} OSB, $l_j^i = 0$ or 1.

REQ-QOBDLR algorithm is performed in every Rsv_Slot. Assume that the i^{th} ISB is receiving REQ_j token and REL_n token $(j \neq n)$ in current Rsv_Slot.

A.1 Operations upon receiving REQ_j token

When receiving REQ_j token, the i^{th} ISB will evaluate its queue occupancy q_j^i against two thresholds : a high threshold (HT) and a low threshold (LT). Then, the i^{th} ISB decides whether to request an extra link and/or release a link to the j^{th} OSB.

Step 1 : The i^{th} ISB decides whether to request an additional link for the j^{th} OSB ?

<u>**Arbitration :**</u> If $q_j^i > HT$, the i^{th} ISB will request an extra link to the j^{th} OSB. If $q_j^i \leq HT$, the i^{th} ISB will not ask for any more link because the j^{th} GVOQ is not heavy loaded.

<u>**Operation :**</u> If the i^{th} ISB decides to request an additional link to the j^{th} OSB, s_j^i in link request vector LK_REQ^i will be set to 1. To be understandable, we use $s_j^{i,old}$ to represent the old value of s_j^i just before the arbitration, and we use s_j^i to present the updated value after arbitration in current Rsv_Cycle.

	IF		*THEN*	
	q_j^i	$s_j^{i,old}$	s_j^i	**REQ_NUM$_j$**
case 1	> HT	0	1	In Step 3, REQ_NUM$_j$ ++;
case 2		1	1	REQ_NUM$_j$
case 3	≤ HT	0	0	REQ_NUM$_j$
case 4		1	if (REQ_NUM$_j$ > 0) $s_j^i = 0$; REQ_NUM$_j$– –; else $s_j^i = 1$;	

Figure A.1 When receiving REQ_j token, the i^{th} ISB decides whether to request an extra link for the j^{th} OSB. The i^{th} ISB uses s_j^i to record a link request for the j^{th} OSB.

As shown in Fig A.1, there are four possible circumstances according to the values of q_j^i and $s_j^{i,old}$. In case 1, the i^{th} ISB generates a new link request which will be inserted into REQ_NUM$_j$ in Step 3. In case 2, the i^{th} ISB had sent a link request but has not obtained the desired link yet. The i^{th} ISB will keep waiting for an available link but will not issue a new link request again. In case 4, the i^{th} ISB will cancel its current link request if REQ_NUM$_j > 0$.

Step 2 : The i^{th} ISB decides whether to release a link if REQ_j token carries link requests ?

Arbitration : Now, if REQ_NUM$_j > 0$, REQ_j token carries link requests for OSB j. The i^{th} ISB will evaluate its queue occupancy q_j^i to decide whether to release a link for OSB j. If $q_j^i < LT$, the i^{th} ISB will unconditionally release one of its reserved links to the j^{th} OSB. If $q_j^i \geq LT$, the i^{th} ISB will release a link for OSB j if the ISB occupies more than $Fair$ links to the j^{th} OSB (i.e. $r_j^i > Fair$). REQ_NUM$_j$ will be reduced by 1 if the i^{th} ISB decides to release a link to the j^{th} OSB.

Operation : Usually, the i^{th} ISB can not utilize REL_n token to pass a released link for the j^{th} OSB, if $j \neq n$. Therefore, the i^{th} ISB does not need to reduce its link reservation r_j^i in current Rsv_Slot, even though it decides to release a link for the j^{th} OSB. The i^{th} ISB only needs to reduce REQ_NUM$_j$ by 1, and records this pending link release for the j^{th} OSB in l_j^i. This pending link release will be activated when the i^{th} ISB receives REL_j token later.

$$l_j^i = \begin{cases} 1 & \text{if } ((q_j^i < LT) \text{ and } (r_j^i > 0)) \\ 1 & \text{if } ((q_j^i \geq LT) \text{ and } (r_j^i > Fair)) \\ 0 & others; \end{cases} \qquad (A.1)$$

Step 3 : The i^{th} ISB updates REQ_j token, then passes REQ_j token to next down-steam ISB.

If a new-born link request for the j^{th} OSB was generated by the i^{th} ISB in Step 1 (i.e. case 1 in Fig A.1), the i^{th} ISB will insert this new request in REQ_j token. Hence, REQ_NUM$_j$ will be increased by 1. Finally, the i^{th} ISB forwards REQ_j token to its down-stream adjacent ISB.

A.2 Operations upon receiving REL_n token

When receiving REL_n token, the i^{th} ISB will decide whether to take a link from REL_n token if REL_n token carries available links.

Step 1 : The i^{th} ISB decides whether to take an available link from REL_n token.

Arbitration : If REL_NUM$_n > 0$, REL_n token accommodates available links to the n^{th} OSB. If the i^{th} ISB has sent a link request for the n^{th} OSB before (i.e. $s_n^i = 1$) and if the total links reserved by the ISB is less than M (i.e. $\sum_{l=0}^{(K-1)} r_l^i \leq M$), the i^{th} ISB will reserve an available link from REL_n token .

Operation : If the i^{th} ISB can grab a link from REL_n token, its link reservation r_n^i will be increased by 1, but REL_NUM$_n$ will be reduced by 1.

Step 2 : The i^{th} ISB updates REL_n token, then passes REL_n token to its up-link neighboring ISB.

If the i^{th} ISB held a pending link release for the n^{th} OSB (i.e. $l_n^i = 1$), **now**, the i^{th} ISB can really release the link for the n^{th} OSB through REL_n token. The i^{th} ISB reduces r_n^i by 1, and increases REL_NUM$_j$ by 1. Finally, the i^{th} ISB passes REL_n token to its up-link neighboring ISB. ⟨END⟩

APPENDIX B

REQREL-QOBDLR ALGORITHM

REQREL-QOBDLR algorithm is illustrated in detail in this appendix. The i^{th} ISB $(0 \leq i < K)$ needs another vector LK_REQ_Modify^i besides the four vectors such as Q^i, LK_RSV^i, LK_REL^i, LK_REQ^i.

• LK_REQ_Modify^i : Link Request Modification Vector in the i^{th} ISB. LK_REQ_Modify^i = $[s_0^{i,modify}, s_1^{i,modify}, \cdots, s_{(K-1)}^{i,modify}]$, where $s_j^{i,modify}$ records the number of link requests that the i^{th} ISB intends to increase/decrease to the total link requests for the j^{th} OSB. If $s_j^{i,modify} > 0$, the i^{th} ISB results in an increase on total link requests for the j^{th} OSB; if $s_j^{i,modify} < 0$, link requests for the j^{th} OSB will be reduced.

Assume that the i^{th} ISB is receiving REQ_j and REL_n token in current Rsv_Slot. Operation for REQ_j token is more similar to than different from that in REQ-QOBDLR algorithm. But, operation for REL_n token is unlike that in REQ-QOBDLR algorithm.

We first describe the operation upon receiving REL_n token. Then we focus on what is the impact of such difference in the operation of REL_n token on the operation for REQ_j token.

B.1 Operations upon receiving REL_n token

$\Longrightarrow$ Step 1 : The i^{th} ISB decides whether to take a released link from REL_n token, if REL_NUM$_n > 0$?

Arbitration : If REL_NUM$_n > 0$, there are available links to the n^{th} OSB. The i^{th} ISB will take a released link from REL_n token only if its queue occupancy $q_n^i \geq LT$.

If $(LT \leq q_n^i \leq HT)$ and $(\sum_{l=0}^{k-1} r_l^i < M)$, the i^{th} ISB will take a link from REL_n token if it has requested a link to the n^{th} ISB (i.e. $s_n^i = 1$).

If $(q_n^i > HT)$ and $(\sum_{l=0}^{k-1} r_l^i < M)$, the i^{th} ISB will grab a link from REL_n token no matter whether it has requested a link for the n^{th} OSB (i.e. $s_n^i = 0$ or 1).

Operation : If the i^{th} ISB decides to take an available link from REL_n token, its link reservation r_n^i will increase 1 but REL_NUM$_n$ will decrease 1.

But, in the circumstance of $q_n^i > HT$, if the i^{th} ISB takes an available link from REL_n token without sending a request in advance (i.e. $s_n^i = 0$), it implies that the i^{th} ISB is snatching a link which is supposed to satisfy another ISB's link request. To compensate the 'stolen' link, the i^{th} ISB will issue a link request for the n^{th} OSB to trigger a new released link not for itself but for another ISB who is still waiting for its desired link. Since the i^{th} ISB can not use REQ_j token to carry a link request for the n^{th} OSB in current Rsv_Slot, hence, the ISB records this pending link request in LK_REQ_Modify by setting $s_n^{i,modify} = s_n^{i,modify} + 1$. As soon as the i^{th} ISB receives REQ_n token in some Rsv_Slot(s) later, ISB i will first add the pending link request into REQ_NUM$_n$.

$\Longrightarrow$ Step 2 : The i^{th} ISB decides whether to release an occupied link to the n^{th} OSB?

Arbitration : If $(q_n^i < LT)$ and $(r_n^i > 0)$, the i^{th} ISB will release one of its occupied link to the n^{th} OSB.

Operation : If the i^{th} ISB decides to release a link, its link reservation r_n^i will reduce 1. The released link will be inserted into REL_n token right away so that REL_NUM$_n$ will increase 1.

Since the i^{th} ISB releases a link based on its own traffic load but it does not know whether other ISBs are demanding this available link for the n^{th} OSB. The potential problem may cause more than necessary links to be released. To avoid this problem, the i^{th} ISB should have a way to reduce the total link requests for the n^{th} OSB if it can release an available link to the n^{th} OSB. In another word, if the i^{th} ISB can release a link for the n^{th} OSB, from a globe point of view of the link requests carried in REQ_n token, only (REQ_NUM$_n$ $- s_j^i - 1$) link requests are demanding available links. As the i^{th} ISB does not have REQ_n token in handy in current Rsv_Slot, the i^{th} ISB will record the pending reduction on link requests for the n^{th} OSB by setting $s_n^{i,modify} = s_n^{i,modify} - s_n^i - 1$. As soon as the i^{th} ISB received REQ_n token in some Rsv_Slot(s) later, the i^{th} ISB will carry out the intending reduction on REQ_NUM$_n$ as follows : REQ_NUM$_n$ = REQ_NUM$_n$ + $s_n^{i,modify}$.

$\Longrightarrow$ Step 3 : The i^{th} modifies REL_n token, and passes it to next up-stream ISB.

Arbitration : If the i^{th} ISB has a record about a pending link release for the n^{th} OSB (i.e. $l_n^i = 1$), now, the ISB will really reduce its link reservation r_n^i by 1 in order to release a link for the n^{th} OSB. Then, the i^{th} ISB will use REL_n token to pass the released link to other OSBs.

Operation : If $l_n^i = 1$, then r_n^i reduces 1 and REL_NUM$_n$ increases 1. Finally, the i^{th} ISB sends REL_n token to its up-link ISB.

B.2 Operations upon receiving REQ_j token

In REQREL-QOBDLR algorithm, operations upon receiving REQ_j token is very similar to that in REQ-QOBDLR algorithm. But, before starting the 3 steps shown in REQ-QOBDLR algorithm (Appendix A), the i^{th} ISB first will update REQ_NUM$_j$ with $s_j^{i,modify}$, i.e. REQ_NUM$_j$ = REQ_NUM$_j$ $+s_j^{i,modify}$. The reason for that is the operations upon receiving REL_j token (which may not be scheduled in current

Rsv_Slot) may cause a potential pending increase/decrease on total link requests carried in REQ_j token. Hence, when the i^{th} ISB receives REQ_j token in present, REQ_NUM$_j$ will be adjusted to be more realistically reflect the number of link requests for the j^{th} OSB. REQ_NUM$_j$ may be negative, it implies that the available links to the j^{th} OSB is more than link requests to the j^{th} OSB in current time point. A negative REQ_NUM$_j$ will not trigger any more link release for the j^{th} OSB. Hence, from long-term point of view, the released links for the j^{th} ISB will keep a balance to the link requests for the j^{th} OSB (i.e. REL_NUM$_j$ $\leq$ REQ_NUM$_j$). This is a characteristic of the REQREL-QOBDLR algorithm. $\langle$END$\rangle$

REFERENCES

1. M. H. Guo, R. S. Chang, *Multicast ATM Switches : Survey and Performance Evaluation*, Computer Communication Review, Vol 28, No. 2, April 1998, pp. 98-131.

2. J. Turner, N. Yamanaka, *Architecture Choices in Large Scale ATM Switches*, WUCS 97-21, May 1997.

3. A. Huang and S. Knauer, *STARLITE: A Wideband Digital Switch*, Proc. IEEE GLOBECOM'84, pp. 121-125, Dec. 1984.

4. J. S. Turner, *Design of a Broadcast Packet Switching Network*, IEEE Trans. on Commun., Vol. 36, June 1988, pp. 734-743.

5. T. T. Lee, *Nonblocking Copy Networks for Multicast Packet Switching*, IEEE J. on Select. Areas in Commun. Vol. 6, December 1988, pp. 1445-1467.

6. J.S. Turner, *A Practical Version of Lee's Multicast Switch Architecture*, IEEE Trans. on Commun., Vol 41, No.8, August 1993, pp. 1166-1169.

7. J. Kim, J. Park, H. Yoon, J.W. Cho, *Fault-Tolerant Multicasting in MIN's for ATM Switches*, IEEE Commun. Letters, Vol. 2, No. 12, Dec. 1998, pp.331-333.

8. K.Y. Eng, M.G. Hluchyj, Y.S. Yeh, *Multicast and Broadcast Services in a Knockout Packet Switch*, Proc. of INFOCOM'88, 1988, pp.29-34.

9. C.K. Kim, T.T. Lee, *Call Scheduling Algorithms in a Multicast Switch*, IEEE Trans. on Commun., Vol. 40, No. 3, March 1992, pp. 625-635.

10. D. X. Chen, J. W. Mark, *SCOQ : a Fast Packet Switch with Shared Concentration and Output Queueing*, IEEE/ACM Trans. on Networking, Vol. 1, 1993, pp.142-151.

11. H. J. Chao, B. S. Choe, *Design and Analysis of A Large-Scale Multicast Output Buffered ATM Switch*, IEEE/ACM Trans. on Networking, Vol. 3, No. 2, April 1995, pp. 126-138.

12. H. J. Chao, B. S. Choe, J. S. Park, N. Uzun, *Design and Implementation of Abacus Switch : A Scalable Multicast ATM Switch*, IEEE J. on Select. Areas in Commun., Vol. 15, No. 5, June 1997, pp. 830-843.

13. K. Wang, M.H. Cheng, *Design and Performance Analysis of a Growable Multicast ATM Switch*, Proc. of INFOCOM'97, pp.934-940.

14. D.J. Marchok, C.E. Rohrs, R.M. Schafer, *Multicasting in a Growable Packet (ATM) Switch*, INFOCOM'91, 1991, pp. 850-858.

15. Y.Yang, G.M. Masson, *Broadcast Ring Sandwich Networks*, IEEE Trans. on Computers, Vol 44, Oct. 1995, pp. 1169-1180.

16. K.Y. Eng, M.J. Karol, G.J. Cyr, M.A. Pashan, *Design and Prototype of a Terabit ATM Switch Using a Concentrator-Based Growable Switch Architecture*, Proc. of International Switching Symposium (ISS), 1995, pp. 404-408.

17. M.R. Hashemi, A. Leon-Garcia, *The Single-Queue Switch : A Building Block for Switches with Programmable Scheduling*, IEEE J. on Select. Areas on Commun., Vol. 15, No. 5, June 1997, pp. 785-793.

18. A.K. Choudhury, E.L. Hahne, *A New Buffer Management Scheme for Hierarchical Shared Memory Switches*, IEEE/ACM Transactions on Networking, Vol. 5, No. 5, October 1997, pp. 728-738.

19. R.C.Chang, C.Y. Hsieh, *Design of Multicast ATM Switch*, IEE Electronics Letters, Vol. 34, No. 22, October 29, 1998, pp. 2089-2091.

20. Y. Xiong, L. Mason, *Multicast ATM Switches Using Buffered MIN Structure : A Performance Study*, IEEE INFOCOM'97, pp. 926-933.

21. F. M. Chiussi, Y. Xia, V. P. Kumar. *Performance of Shared-Memory Switches Under Multicast Bursty Traffic*, IEEE J. on Select. Areas in Commun., Vol. 15, No. 3, April 1997, pp. 473-487.

22. A. Racz, G. Fodor, Z. Turanyi, *Weighted Fair Early Packet Discard at an ATM Switch Output Port*, IEEE INFOCOM'99, S 8E.

23. J.S. Wu, C.C. Ke, *ATM Shared Memory Switch with Multicasting Balancing*, IEICE Trans. on Commun., Vol. E78-B, No. 9, September 1995, pp. 1262-1268.

24. H.J.Chao, N. Uzun, *An ATM Queue Manager Handling Multiple Delay and Loss Priorities*, IEEE/ACM Trans. on Networking, Vol. 3, No. 6, December 1995, pp. 652-659.

25. A.K.Choudhury, E.L.Hahne, *A New Buffer Management Scheme for Hierarchical Shared Memory Switches*, IEEE/ACM Trans. on Networking, Vol. 5, No. 5, October, 1997, pp.728-738.

26. S.H.Byun, D.K.Sung, *A General Expansion Architecture for Large-Scale Multicast ATM Switches*, IEEE GLOBECOM'97, S7-7.

27. S. Kumar, D.P. Agrawal, *On Multicast Support for Shared-Memory-Based ATM Switch Architecture*, IEEE Network, Jan/Feb 1996, pp. 34-39.

28. M. R. Hashemi, A. Leon-Garcia, *A Multicast Single-Queue Switch with a Novel Copy Mechanism*, Proc. of IEEE INFOCOM'98.

29. M. R. Hashemi, A. Leon-Garcia, *A Multicast Single-Queue Switch with a Novel Copy Mechanism*, IEEE INFOCOM'98, S07A-2.

30. N. Mckeown, V. Anantharam, J. Walrand, *Achieving 100% Throughput in an Input-Queued Switch*, Proc. of IEEE INFOCOM'96, March 1996.

31. A. Mekkittikul, N. McKeown, *A Starvation-free Algorithm For Achieving 100% Throughput in an Input-Queued Switch*, Proc. of ICCCN'96.

32. A. Mekkittikul, N. Mckeown, *A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches*, Proc. of IEEE INFOCOM'98, April 1998.

33. B. Prabhakar, N. McKeown, and R. Ahuja, *Multicast Scheduling for Input-Queued Switches*, IEEE J. on Select. Areas in Commun., Vol. 15, No. 5, pp. 855-866, 1997.

34. B. Prabhakar, N. Mckeown, *Designing A Multicast Switch Scheduler*, Proc. of the 33th Annual Allerton Conference on Communication, Control and Computing, October 1995.

35. S-T. Chuang, A. Goel, N. Mckeown, B. Prabhakar, *Matching Output Queueing with Combined Input and Output Queueing*, Proc. of IEEE INFOCOM'99, March, 1999.

36. M. Andrews, S.Khanna, K. Kumaran, *Integrated Scheduling of Unicast and Multicast Traffic in an Input-Queued Switch*, IEEE INFOCOM'99, S-8E.

37. J. S-C. Chen, R. Guerin, *Performance Study of an Input Queueing Packet Switch with Two Priority Classes*, IEEE Trans. on Commun., Vol. 39, No. 1, pp.117-126.

38. L. Jacob, A. Kumar, *Delay Performance of Some Scheduling Strategies in an Input Queuing ATM Switch with Multiclass Bursty Traffic*, IEEE/ACM Trans. on Networking, Vol. 4, No. 2, April 1998, pp. 258-271.

39. M.J. Karol, M.G. Hluchyj, S.P.Morgan, *Input versus Output Queueing on a Space-Division Packet Switch*, IEEE Trans. on Commun., Vol. Com-35, No. 12, December 1987, pp. 1347-135.

40. T. T. Lee, *A Modular Architecture for Very Large Packet Switches*, Proc. of GLOBECOM'89, Dec. 1989, pp. 1801-1809.

41. J. N. Giacopelli, J. J. Hickey, W. S. Marcus, W. D. Sincoskie, M. Littlewood, *Sunshine : A High-Performance Self-Routing Broadband Packet Switch Architecture*, IEEE J. Select. Areas in Commun., Vol. 9, No. 8, Oct. 1991, pp. 1289-1298.

42. Bellcore, *Broadband Switching System (BSS) Generic Requirements, BSS Performance*, GR-110-CORE, Issue 1, Sept. 1994.

43. F. Sestini, *Recursive Copy Generation for Multicast ATM Switching*, IEEE/ACM Trans. on Networking, Vol. 5, No. 3, June 1997, pp.329-335.

44. S. C. Liew, *A General Packet Replication Scheme for Multicasting in Interconnection Networks*, IEEE INFOCOM'95, 3d.4.1-3d.4.8.

45. J. F. Hayes, R. Breault, M. K. Mehmet-Ali, *Performance Analysis of a Multicast Switch*, IEEE Trans. on Commun., Vol 39, No. 4, pp. 581-587.

46. J.S-C, Chen, T.E.Stern, *Throughput Analysis, Optimal Buffer Allocation, and Traffic Imbalance Study of a Generic Nonblocking Packet Switch*, IEEE J. on Select. Areas in Commu., Vol. 9, No. 3, April 1991, pp. 439-449.

47. W. D. Zhong, Y. Onozato, J. Kaniyil. *A Copy Network with Shared Buffers for Large-Scale Multicast ATM Switching*, IEEE/ACM Trans. on Networking, Vol. 1, No. 2, April 1993, pp. 157-165.

48. R. P. Bianchini Jr., H. S. Kim. *Design of a Nonblocking Shared-Memory Copy Network for ATM*, Proc. of INFOCOM'92, pp. 6D.3.1-6D.3.10.

49. P. S. Min, M, V. Hegde, and H. S. Saidi, A. Chandra. *Nonblocking Copy Networks in Multi-Channel Switching*, IEEE/ACM Trans. on Networking, Vol. 3, No. 6, December 1995, pp. 857-871.

50. X. Liu, H. T. Mouftah. *Queuing Performance of Copy Networks With Dynamic Cell Splitting for Multicast ATM Switching* IEEE Trans. on Commun., Vol. 45, No. 4, April 1997, pp. 464-472.

51. F. M. Chiussi, Y. Xia, V. P. Kumar, *Performance of Shared-Memory Switches Under Multicast Bursty Traffic*, IEEE/ACM Trans. on Networking, Vol. 1, No. 2, April 1993, pp. 157-165.

52. W.T. Chen, Y.L. Chang, W.Y. Hwang, *A High Performance Cell Scheduling Algorithm in Broadband Multicast Switching Systems*, IEEE GLOBECOM'97, S5B-3.

53. C.K. Kim, *Performance Analysis of a Duplex Multicast Switch*, IEEE Trans. on Commun., Vol. 40, No. 10, October 1992, pp. 1615-1624.

54. N. Uzun, A. Blok, *Ten Terabit Multicast Packet Switch with SRRM Scheduling Algorithm*, BSS'99, Kingston, Canada, June 1999.

55. R. O. LaMaire, D. N. Serpanos, *Two-Dimensional Round-Robin Schedulers for Packet Switches with Multiple Input Queues*, IEEE/ACM Trans. on Networking, Vol. 2, No. 5, October 1994, pp. 471-482.

56. J.F. Hayes, R. Breault, M. K. Mehmet-Ali, *Performance Analysis of a Multicast Switch*, IEEE Trans. on Commun., Vol. 39, No. 4, April 1991, pp. 581-587.

57. C.S. Wu, G.K. Ma, B-S. P. Lin, *A Cell Scheduling Algorithm for VBR Traffic in an ATM Multiplexer*, IEEE, 1995, pp. 632-637.

58. S.Q. Li, J.W. Mark, *Traffic Characterization for Integrated Services Networks*, IEEE Trans. on Commun., Vol 38, No. 8, August 1990, pp. 1231-1243.

59. A.K. Choudhury, E.L. Hahne, *Dynamic Queue Length Thresholds for Shared-Memory Packet Switches*, Vol. 6, No. 2, April 1998, pp.130-140.

60. M. Murata, Y. Oie, T. Suda, H. Miyahara, *Analysis of a Discrete-Time Single-Server Queue with Bursty Inputs for Traffic Control in ATM Networks*, IEEE J. on Select. Areas in Commun., Vol. 8, No. 3, April 1990, pp. 447-458.

61. N. Akar, N.C. Oguz, K. Sohraby, *Matrix-Geometric Solutions of M/G/1-Type Markov Chains : A Unifying Generalized State-Space Approach*, IEEE J. on Select. Areas in Commun., Vol. 16, No. 5, June 1998, pp. 626-639.

62. R. Jafari, K. Sohraby, *Performance Analysis of a Priority based ATM Multiplexer with Correlated Arrivals*, IEEE INFOCOM'99, pp. 1036-1043.

63. R. Jafari, K. Sohraby, *General Discrete-Time Queueing Systmes with Correlated Batch Arrivals and Departures*, IEEE INFOCOM'00, March 2000.

64. N. Matsufuru, R. Aibara, *Efficient Fair Queueing for ATM Networks using Uniform Round Robin*, IEEE INFOCOM'99, pp. 389-397.

65. OPNET by MIL3 Inc., Washington, DC 20008.