Fall 2007

Architecture design and performance analysis of practical buffered-crossbar packet switches

Ziqian Dong
New Jersey Institute of Technology

ABSTRACT

ARCHITECTURE DESIGN AND PERFORMANCE ANALYSIS OF PRACTICAL BUFFERED-CROSSBAR PACKET SWITCHES

by
Ziqian Dong

Combined input crosspoint buffered (CICB) packet switches were introduced to relax input-output arbitration timing and provide high throughput under admissible traffic. However, the amount of memory required in the crossbar of an $N \times N$ switch is $N^2 \times k \times L$, where k is the crosspoint buffer size and needs to be of size RTT in cells, L is the packet size. RTT is the round-trip time which is defined by the distance between line cards and switch fabric. When the switch size is large or RTT is not negligible, the memory amount required makes the implementation costly or infeasible for buffered crossbar switches. To reduce the required memory amount, a family of shared memory combined-input crosspoint-buffered (SMCB) packet switches, where the crosspoint buffers are shared among inputs, are introduced in this thesis. One of the proposed switches uses a memory speedup of m and dynamic memory allocation, and the other switch avoids speedup by arbitrating the access of inputs to the crosspoint buffers. These two switches reduce the required memory of the buffered crossbar by 50% or more and achieve equivalent throughput under independent and identical traffic with uniform distributions when using random selections.

The proposed mSMCB switch is extended to support differentiated services and long RTT. To support P traffic classes with different priorities, CICB switches have been reported to use $N^2 \times k \times L \times P$ amount of memory to avoid blocking of high priority cells. The proposed SMCB switch with support for differentiated services requires $\frac{1}{mP}$ of the memory amount in the buffered crossbar and achieves similar throughput performance to that of a CICB switch with similar priority management, while using no speedup in the shared memory.

The throughput performance of SMCB switch with crosspoint buffers shared by inputs (I-SMCB) is studied under multicast traffic. An output-based shared-memory crosspoint-buffered (O-SMCB) packet switch is proposed where the crosspoint buffers are shared by two outputs and use no speedup. The proposed O-SMCB switch provides high performance under admissible uniform and nonuniform multicast traffic models while using 50% of the memory used in CICB switches. Furthermore, the O-SMCB switch provides higher throughput than the I-SMCB switch.

As SMCB switches can efficiently support an RTT twice as long as that supported by CICB switches and as the performance of SMCB switches is bounded by a matching between inputs and crosspoint buffers, a new family of CICB switches with flexible access to crosspoint buffers are proposed to support longer RTTs than SMCB switches and to provide higher throughput under a wide variety of admissible traffic models. The CICB switches with flexible access allow an input to use any available crosspoint buffer at a given output. The proposed switches reduce the required crosspoint buffer size by a factor of N, keep the service of cells in sequence, and use no speedup. This new class of switches achieve higher throughput performance than CICB switches under a large variety of traffic models, while supporting long RTTs.

Crosspoint buffered switches that are implemented in single chips have limited scalability. To support a large number of ports in crosspoint buffered switches, memory-memory-memory (MMM) Clos-network switches are an alternative. The MMM switches that use minimum memory amount at the central module is studied. Although, this switch can provide a moderate throughput, MMM switch may serve cells out of sequence. As keeping cells in sequence in an MMM switch may require buffers be distributed per flow, an MMM with extended memory in the switch modules is studied. To solve the out of sequence problem in MMM switches, a queuing architecture is proposed for an MMM switch. The service of cells in sequence is analyzed.

ARCHITECTURE DESIGN AND PERFORMANCE ANALYSIS OF PRACTICAL BUFFERED-CROSSBAR PACKET SWITCHES

by
Ziqian Dong

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 in Electrical Engineering

Department of Electrical and Computer Engineering

January 2008

APPROVAL PAGE

ARCHITECTURE DESIGN AND PERFORMANCE ANALYSIS OF PRACTICAL BUFFERED-CROSSBAR PACKET SWITCHES

Ziqian Dong

Dr. Roberto Rojas-Cessa, Dissertation Advisor — Date
Associate Professor, Department of Electrical and Computer Engineering, New Jersey Institute of Technology

---

Dr. Nirwan Ansari, Committee Member — Date
Professor, Department of Electrical and Computer Engineering, New Jersey Institute of Technology

---

Dr. Sotirios G. Ziavras, Committee Member — Date
Professor, Department of Electrical and Computer Engineering, New Jersey Institute of Technology

---

Dr. Jie Hu, Committee Member — Date
Assistant Professor, Department of Electrical and Computer Engineering, New Jersey Institute of Technology

---

Dr. Aleksandar Kolarov, Committee Member — Date
Senior Scientist, Applied Research, Telcordia Technologies

BIOGRAPHICAL SKETCH

Author: Ziqian Dong

Degree: Doctor of Philosophy

Date: January 2008

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2008
- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ 2002
- Bachelor of Science in Electrical Engineering,
  Beijing University of Aeronautics and Astronautics, Beijing, China, 1999

Major: Electrical Engineering

Publications:

Z. Dong and R. Rojas-Cessa, "Output-Based Shared-Memory Crosspoint-Buffered Packet Switch for Multicast Services," *to appear in IEEE Communications Letters, December 2007.*

R. Rojas-Cessa, Z. Dong, and Z. Guo, "Load-Balanced Combined Input-Crosspoint Buffered Packet Switch and Long Round-Trip Times," *IEEE Communications Letters*, Vol. 4, issue 7, pp. 661 - 663, July 2005.

Z. Dong and R. Rojas-Cessa, "Efficient Packet Replication and Switching of Multicast Traffic by Shared-Crosspoint Packet Switches," *Proc. IASTED International Conference on Communication Systems, Networks, and Applications, CSNA 2007,* October 8-10, 2007.

R. Rojas-Cessa, L. Ramesh, Z. Dong, B. D'Alessandro and N. Ansari, "Implementation of A Parallel-Search Trie-Based Scheme for Fast IP Lookup," *Proc. IASTED International Conference on Communication Systems, Networks, and Applications, CSNA 2007,* October 8-10, 2007.

R. Rojas-Cessa, L. Ramesh, Z. Dong, L. Cai and N. Ansari, "Parallel-Search Trie-based Scheme for Fast IP Lookup," *Proc. IEEE IEEE Global Telecommunications Conference,* Washington D.C., 6 pages, November 26 - 30, 2007.

Z. Dong and R. Rojas-Cessa, "Packet Switching and Replication of Multicast Traffic by Crosspoint Buffered Packet Switches," *Proc. IEEE Workshop on High Performance Switching and Routing*, Brooklyn, NY, 6 pages, May 30 - Jun 1, 2007.

Z. Dong and R. Rojas-Cessa, "Shared-Memory Combined Input-Crosspoint Buffered Packet Switch for Differentiated Services," *Proc. IEEE Global Telecommunications Conference*, San Francisco, CA, 5 pages, November 27 - December 1, 2006.

R. Rojas-Cessa and Z. Dong, "Combined Input-Crossspoint Buffered Packet Switch with Flexible Access to Crosspoints Buffers: One-Cell Size Case," *Proc. 6th IEEE International Caribbean Conference on Devices, Circuits and Systems*, Playa del Carmen, Mexico, 5 pages, April 26 - 28, 2006.

R. Rojas-Cessa, Z. Dong, and S. Ziavras, "Load-Balanced-CICB Packet Switch and Long Round-Trip Time Support," *Proc. of IEEE IEEE Global Telecommunications Conference*, St. Luis, MO, pp. 1002-1006, November 28 - December 2, 2005.

Z. Dong and R. Rojas-Cessa, "Long Round-Trip Time Support with Shared-Memory Crosspoint Buffered Packet Switch," *Proc. IEEE 13th Symposium on High Performance Interconnects*, Palo Alto, CA, pp. 138-143, August 17 - 19, 2005.

R. Rojas-Cessa and Z. Dong, "Combined Input-Crosspoint Buffered Packet Switch with Shared Crosspoint Buffers," *Proc the 39th Conference on Information Sciences and Systems*, Baltimore, MD, March 16 - 18, 2005.

To my parents and my husband

ACKNOWLEDGMENT

I wish to express my sincere gratitude to my advisor, Dr. Roberto Rojas-Cessa, for giving me the opportunity to work with him and guiding me through this learning process. I am greatly indebted to my advisor for his inspiration and encouragement over the last five years. He is always open to new ideas and I really appreciate the support he gave me while working on my research. This research could not have been possible without his brilliant ideas. Not only was he readily available for me when I have questions, as he generously is for all his students, but also he carefully goes through the drafts of my work. Moreover, Dr. Rojas-Cessa also provided valuable facilities and brilliant guidelines for testing and evaluation of my dissertation. I owe much for his unending help to do the research and for his financial support during my graduate study.

I would also like to thank Dr. Nirwan Ansari, Dr. Sotirios Ziavras, Dr. Jie Hu, and Dr. Aleksandar Kolarov for serving on my committee, reviewing this dissertation and making constructive comments.

My thanks go out to my lab mates Chuan-bi Lin, Zhen Guo, Zhen Qin and Lin Cai who provided help and suggestions for my research. All my friends in and out of NJIT have been great source of support and fun. I thank them for making this journey productive and enjoyable.

I am grateful to the donors of the Hashimoto Fellowship and the National Science Foundation which provide financial support for my study.

I am forever indebted to the unconditional love and trust of my family. My parents and my brother have always been a source of inspiration, support, advice and happiness without which I would not have gone this far. Most importantly, I would like to thank my husband for his patience, love and support throughout the years.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter Page

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter Page

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure Page

LIST OF FIGURES
(Continued)

Figure | Page

LIST OF FIGURES
(Continued)

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Packet Switches

The exponential growth of Internet traffic requires high speed packet switches and routers to perform switching at higher speed, low latency and to support traffic with multiple priorities. Packet switches are found at the backbone and gateway levels of a network where one network connects with another as well as at the subnetwork level, where data is forwarded in a direction close to its destination. Packet switches operate at layer two of the Open System Interconnection (OSI) model. Those switches that operate at both layer two and three are referred to as routers [2]. Packet switches determine the output port a packet is to be forwarded based on the information from the destination IP address in the packet header and its own routing table. In this dissertation, packets conform with the Internet Protocol (IP).

A packet switch consists of input/output line interface cards (LICs), which perform processes on incoming IP packets, such as segmentation, IP lookup, scheduling, and of a switch fabric which is responsible for transferring fixed-length cells from inputs to outputs [3]. An IP packet is processed in the input line cards and then forwarded to its destination port as retrieved by IP lookup in the LIC through the switch fabric. To synchronize the internal operation within the switch core, incoming variable-size IP packets are segmented into fixed-length packets, called cells, at the ingress side of a switch and re-assembled at the egress side, before the packets depart from the switch. In this dissertation, the use of cells is considered. Time slot is defined as the time it takes for a switch to transfer a cell from an input to an output.

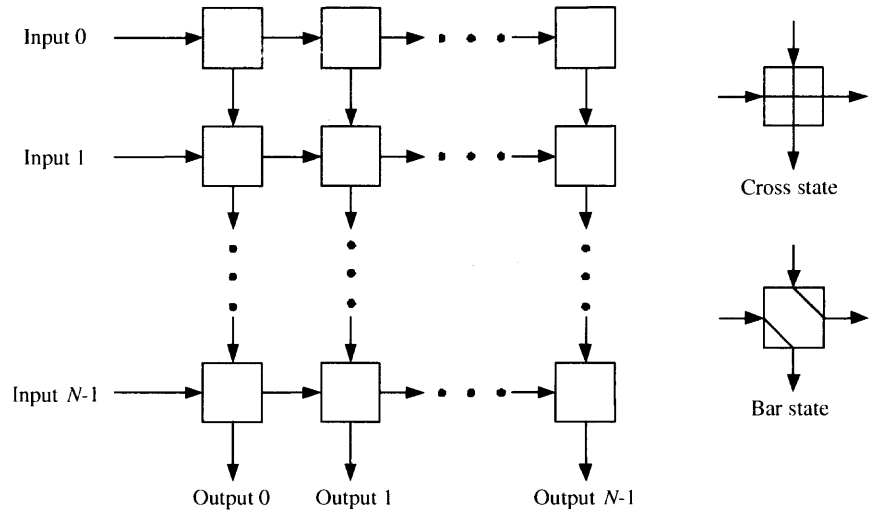

Figure 1.1 An $N \times N$ crossbar switch.

Different designs of switch fabrics were introduced in [4] [5]. In general, the design of a switch fabric can be summarized into two categories: single-stage switch fabric and multiple-stage switch fabric.

For single-stage switch fabrics as in [6]-[7], there is one path available for a cell to go through from an input to an output. A typical implementation of single-stage switch is the bufferless and buffered crossbar switch. In an $N \times N$ bufferless crossbar switch, as shown in Figure 1.1, there are N^2 crosspoints, each of which represents an input-output pair. Each crosspoint has two states, cross state and bar state. The connection of input i to output j is established when crosspoint from i to j is at bar state. There is no connection between input i and output j when crosspoint from i to j is set at cross state. A bufferless crossbar switch has the advantage of implementation simplicity and of being internally nonblocking. However, the scheduling process at the inputs of selecting which output to send a cell has to be done in a very limited time otherwise it becomes a bottleneck of the system when the switch size, N, is large. Buffered crossbar switches, as shown in Figure 1.2, employ buffers at the crosspoints to store internally blocked cells to reduce loss rate [8].

For multiple-stage switch fabrics, there are multiple paths available for a cell to traverse to its destination. Banyan switches [9], multiplane switches [10], and Clos switches

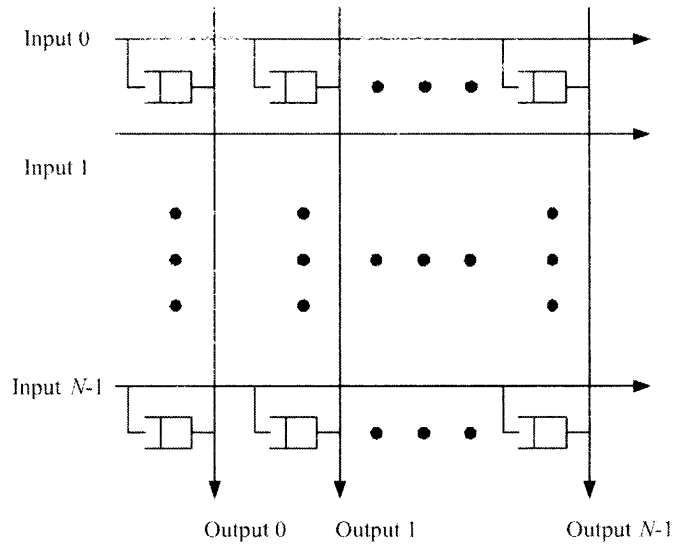

Figure 1.2 An $N \times N$ buffered crossbar switch.

[11] are examples of multiple-stage switch fabrics. These switches have the advantage of being scalable, but have the problem of blocking by internal contentions. To avoid this blocking condition, the switches need centralized scheduling process to resolve contention, which can result in a complicated implementation since different modules can be located on different chips and boards.

1.1.1 Output Buffered vs. Input Buffered Switches

Single staged packet switches can be categorized as output buffered (OB) switches, iutput buffered (IB) switches and crosspoint buffered switches based on the buffering strategies [12].

Output buffered (OB) switches, as shown in Figure 1.3 (a), employ buffers at the output ports. OB switches can achieve 100% throughput since cells from the inputs will not be blocked by the head-of-line (HOL) cells. However, output buffers need to store N cells in each time slot for an $N \times N$ switch. The memory needs to run N times faster than the line rate (speedup of N) for dual access memory. In practical backbone packet switches, port speed can be tens Gbits/s. Memory speedup in OB switches makes implementation infeasible when switch size is large because the speed of memory limits the switch size.

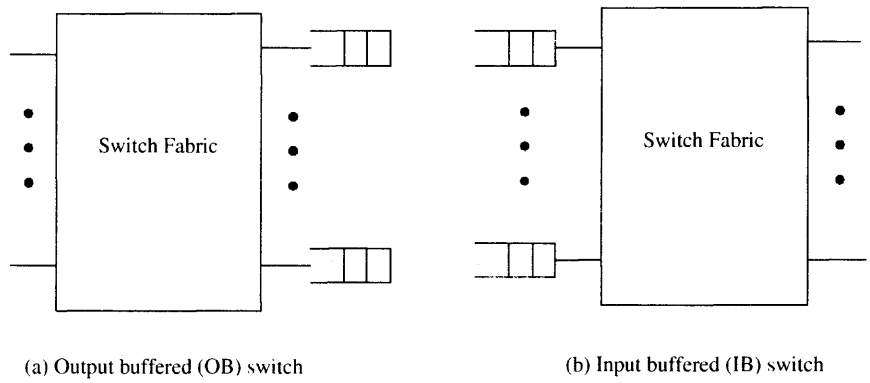

Figure 1.3 Output buffered (a) and Input buffered (b) switch.

IB switches, as shown in Figure 1.3 (b), employ first-in first-out (FIFO) buffers at the inputs. To resolve output contention, IB switches need a scheduler to arbitrate cells from different inputs to each output [13] [14] [15]. Scheduling schemes used by the scheduler can be categorized as weighted and weightless scheduling schemes [16][17]. Random selection and round-robin selection are examples of weightless scheduling schemes. Longest queue first (LQF), oldest cell first (OCF) are examples of weighted scheduling schemes. IB switches are easy to implement and scalable. However, IB switches suffer from HOL blocking where cells are prevented from reaching a free output port because of other cells are ahead of it in the buffer and those cannot be transmitted over the switch fabric due to contention. HOL blocking limits the throughput of the IB switch to 58.6% under uniform traffic [18]. The throughput of an IB switch also depends on the scheduling schemes used. Virtual output queues (VOQs) are implemented to resolve the problem of HOL blocking, where N virtual output queues are used to store cells destined to different outputs at each input. For a crossbar switch with VOQs at the inputs, the throughput can reach 100% using maximum matching algorithms [19] [20].

1.1.2 Combined-Input Crosspoint Buffered Packet Switches

Combined input-crosspoint buffered (CICB) switches are an alternative to input-buffered switches to relax arbitration timing and to provide high-performance switching for packet

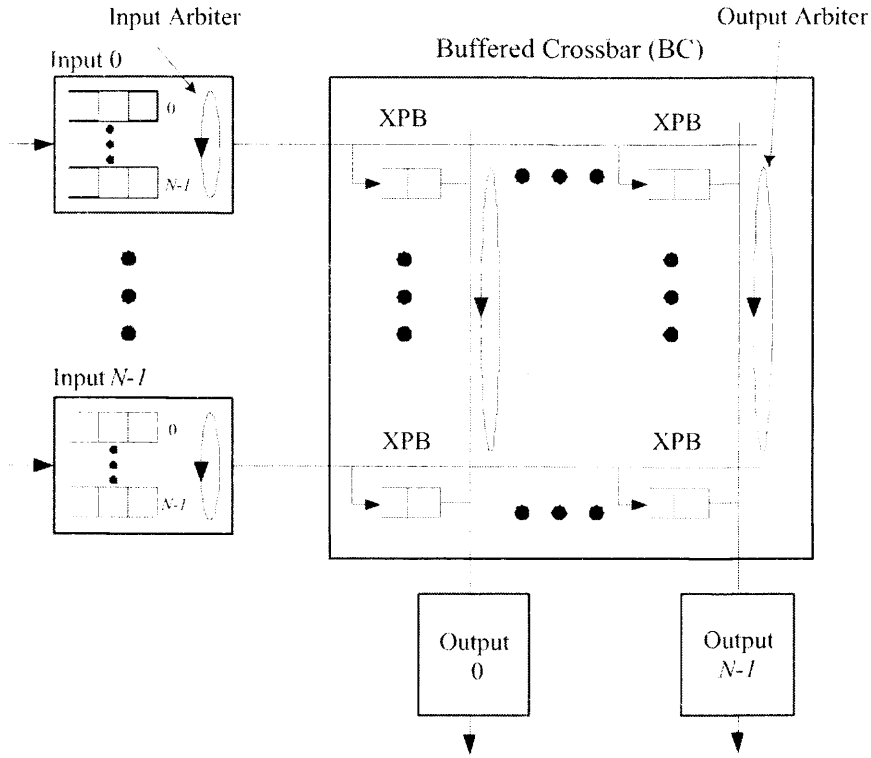

Figure 1.4 Combined input crosspoint buffered switch architecture.

switches with high-speed ports [1] [21] [22] [23]. These packet switches use time efficiently as input and output port arbitrations are performed independently [8]-[14]. Figure 1.4 shows the architecture of a CICB switch. For an $N \times N$ CICB switch, there are N VOQs at each input. Each VOQ stores cells for one of the N outputs. There are N^2 buffered crosspoints, each of which stores cells from input i to output j, where $0 \leq i, j \leq N-1$. A credit-based flow control mechanism is used to indicate the inputs of the crosspoint buffer availability. Different scheduling schemes were proposed in [6] [24] [25] to improve the performance under non-uniform traffic.

1.2 Problem Statement

1.2.1 Effect of Long Round Trip Times

In practical high-speed core packet switches, line cards and the switch fabric can be located on different racks as seen in Figure 1.5 [26]. The distance between line cards and switch

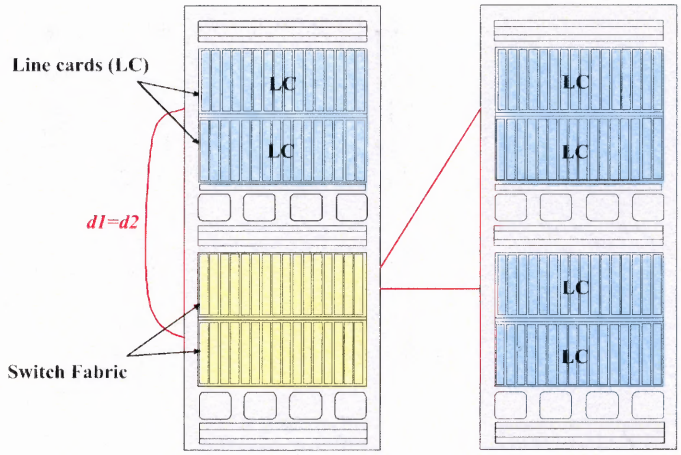

Figure 1.5 Multi-Rack Packet Switch.

fabric can be tens to hundreds of meters. When the port speed is several Gbits/s, the cell transmission delay through the cable connecting the line cards and the switch fabric becomes non-negligible [27]. In this dissertation, round trip time (RTT) is defined as as the sum of the delays of 1) the input arbitration IA, 2) the transmission of a cell from an input to the crossbar $d1$, 3) the output arbitration OA, and 4) the transmission of the flow-control information back from the crossbar to the input, $d2$. Cell and bit alignments are included in the transmission times [28] [29]. This dissertation considers equal distance between line cards and switch fabric.

To keep up with high data rates, switch ports must be able to handle flows of up to R_c b/s,[1] where R_c is the data-rate capacity of a port in a switch or router. In a CICB switch (e.g., the CIXB switch presented in [1]), the maximum flow rate that the switch can handle is $R_c \frac{k}{RTT}$. Note that when $r_{f(i,j)} = R_c$, where $r_{f(i,j)}$ is the rate of $f(i,j)$, the maximum flow rate that the CIXB switch can transfer from inputs to outputs is equivalent to its achievable throughput. A method is proposed in [30] to reduce memory size in the buffered crossbar.

The amount of memory in a buffered crossbar is

$$N^2 \times k \times L, \tag{1.1}$$

[1] In contrast, switches unable to support such flows can only handle aggregated data rates of R_c b/s, where each flow might have a data rate r_{single}, such that $r_{single} < R_c$.

where N is the number of input/output ports, k is the crosspoint buffer size in number of cells, and L is the cell size in bytes. The value of k is defined by the length of the round-trip time (RTT). For example, the switch proposed in [1] requires the size of k to be equal to or larger than the RTT to avoid throughput degradation or crosspoint-buffer underflow for flows (here defined as the data arriving at input i and destined to output j, where $0 \leq i, j \leq N-1$) with high data rates.

1.2.2 Fast Memory in Switch Fabric

As stated in Subsection 1.1.2, CICB switches use memory in the buffered crosspoints. For practical high-speed core packet switches, the line speed is in the scale of Gbit/s. The memory access speed needs to be at least the same as the line speed. As the switch fabric is implemented on a single chip, the amount of memory that can be allocated is limited by the real estate of the chip. Therefore, fast memory is needed with the speed of static random access memory (SRAM) and the density of dynamic random access memory (DRAM) in the switch fabric for the CICB switches [31]. SRAM is suitable for implementation in the buffered switch fabrics for its access speed. However, SRAM is more expensive and less dense than DRAM. New switch architectures that use less memory are needed to reduce implementation cost and support long RTT.

CHAPTER 2

SHARED MEMORY CROSSPOINT BUFFERED SWITCHES

2.1 Introduction

This chapter addresses the need of saving memory amount in the buffered crossbar while supporting long RTTs by two shared memory switch architectures. It is first shown that CICB switches suffer from throughput degradation under long RTTs. To support long RTTs, CICB switches need the amount of memory proportional to the switch size and RTT. The implementation of CICB switches becomes infeasible when switch size is large and RTT is long.

The throughput of the CICB switch is observed under different k and RTT values in a 32×32 switch to validate the traffic model. Different from [1], $RTT > 0$ is considered in this dissertation. Here, it is assumed that the distances between input ports and the buffered crossbar are identical.[2] To model flows with different rates, unbalanced traffic model is used [1]. The unbalanced traffic model uses a probability, w, as the fraction of input load directed to a single predetermined output, while the rest of the input load is directed to all outputs with uniform distribution. For an $N \times N$ CICB switch with input port s, output port d, and the offered input load for each input port ρ, the traffic load from input port s to output port d, $\rho(s, d)$ is given by,

$$\rho(s,d) = \begin{cases} \rho\left(w + \frac{1-w}{N}\right) & \text{if } s = d \\ \rho\frac{1-w}{N} & \text{otherwise.} \end{cases}$$

When $w = 0$, the offered traffic is uniform and can be represented as the matrix below,

[2]The results in this dissertation also apply for non-identical distances.

$$\bar{\rho} = \rho \begin{pmatrix} \frac{1}{N} & \cdots & \frac{1}{N} \\ \vdots & \ddots & \vdots \\ \frac{1}{N} & \cdots & \frac{1}{N} \end{pmatrix}$$

On the other hand, when $w = 1$, it is completely directional, from input i to output j, where $i = j$. This means that all traffic of input port s is destined for only output port d, where $s = d$.

The unbalanced traffic can be represented as

$$\bar{\rho} = \rho \begin{pmatrix} w + \frac{1-w}{N} & \cdots & \frac{1-w}{N} \\ \vdots & \ddots & \vdots \\ \frac{1-w}{N} & \cdots & w + \frac{1-w}{N} \end{pmatrix}$$

As explained in 1.2.1, R_c is the data-rate capacity of a port in a switch or router and $r_{f(i,j)}$ is the rate of $f(i,j)$. Under unbalanced traffic, the fraction of R_c that $f(i,j)$ uses is $r_{f(i,j)} = w + \frac{1-w}{N}$. The maximum data rate of $f(i,j)$ is represented by setting $w = 1$ or $r^{max}_{f(i,j)} = R_c$, and the minimum data rate is represented when $w = 0$ or $r^{min}_{f(i,j)} = \frac{1}{N}$. These two w values of the unbalanced traffic model are the critical points in analyzing the throughput performance.

Figure 2.1 shows that when flows have a rate $r_{f(i,j)} = r^{min}_{f(i,j)}$ (i.e., w=0) for different k values, such that $RTT - k < N$, the throughput is 100%, as shown by curves 1) and 5), where $RTT - k = 0$, and as shown by curves 4) and 6), where $RTT - k = 31$. The uniform distribution of traffic relaxes the demand for buffer space, resulting in high throughput. The figure also shows that when $RTT - k \geq N$, the throughput is less than 100%, as shown by curve 2), where $RTT - k = 32$.

Furthermore, as the data rate of the flow increases (i.e., w), throughput degradation occurs. The worst-case scenario is observed when $r_{f(i,j)} = R_c$ b/s (i.e., w=1) where the

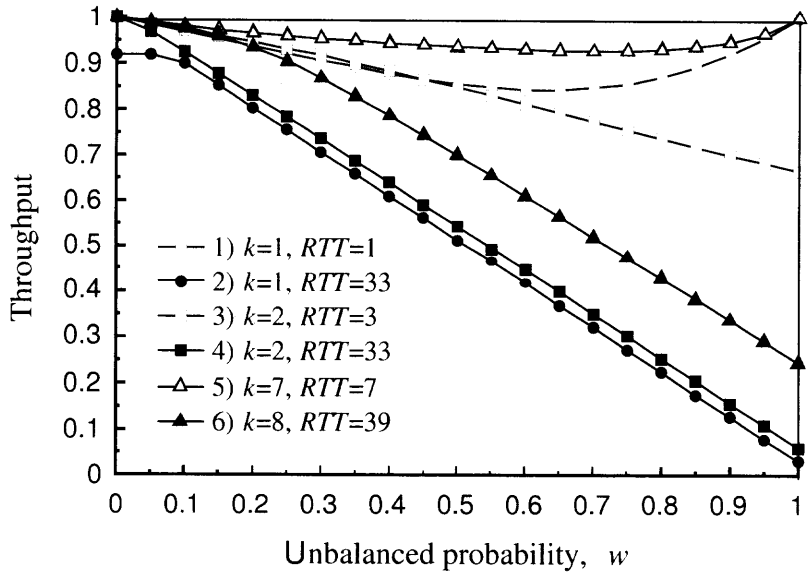

Figure 2.1 Throughput performance of a CICB switch [1] with $RTT > 0$.

achieved throughput is $\frac{k}{RTT}$ for $RTT - k > 0$, as curves 2)-6) show. In this case, up to k cells can be sent back-to-back in RTT time slots.

The amount of memory in a buffered crossbar is

$$N^2 \times k \times L, \tag{2.1}$$

where N is the number of input/output ports, k is the crosspoint buffer size in number of cells, and L is the cell size in bytes. The value of k is defined by the length of the round-trip time (RTT). For example, the switch proposed in [1] requires the size of k be equal to or larger than the round-trip time to avoid throughput degradation or crosspoint-buffer underflow for flows (here defined as the data arriving at input i and destined to output j, where $0 \leq i, j \leq N - 1$) with high data rates.

In a CICB switch, the crosspoint-buffer size to avoid underflow by flows of data rate R_c b/s, where R_c is the port speed, is

$$RTT = d1 + OA + d2 + IA \leq k, \tag{2.2}$$

such that cells are transmitted continuously every time slot [1].

Furthermore, as the buffered crossbar can be physically located far from the input ports in a multi-rack router implementation, actual RTTs can be long. To support long RTTs in a buffered-crossbar switch, the crosspoint-buffer size needs to be increased [32], such that up to RTT cells can be buffered. However, the memory amount that can be allocated in a chip is limited, and therefore, it can make the implementation costly or infeasible when the distance between line cards and the buffered crossbar is long, or else to provide k such that $k < RTT$ without supporting high data rates. An interesting scheme using limited memory is presented in [30] for a switch with p traffic classes, where the crosspoint buffer size is larger than RTT for a single class, and smaller than $p \times RTT$.

A solution to keep the crosspoint buffer small while supporting long RTTs and high data rates is needed.

In this chapter, two shared memory combined input crosspoint buffered switches (SMCB) are proposed to reduce the required memory amount in the buffered crossbar as in CICB switches and support long RTT. These switches use shared memory in the crosspoint buffers to reduce the total crosspoint buffer size such that flows with high data rates can be handled with smaller memory than a switch with dedicated buffers. This chapter shows that a SMCB switch supports a given round-trip time with half or less memory than a buffered crossbar with dedicated crosspoint buffers and deliver equivalent switching performance. Furthermore, it is shown that no speedup is needed when using the shared-memory approach.

2.2 Shared Memory Combined Input Crosspoint Buffered Switches (SMCB)

As discussed in Section 2.1, the largest throughput degradation occurs when the $r_{f(i,j)} = R_c$ b/s, or $w = 1$ in the unbalanced traffic model. Under these conditions, all traffic at input i goes to the crosspoint that connects to output j and the other crosspoints receive no traffic. This motivates the sharing of the crosspoint memory by two or more inputs. In

an SMCB switch, the crosspoint buffer is shared by m inputs, where $1 \leq m \leq N$. Two SMCB switch architectures are proposed, SMCB with dynamic memory allocation, or the SMCB$\times m$ switch, and speedup $=m$ and an SMCB switch with no speedup and input access schedulers, or the mSMCB switch. The SMCB$\times m$ switch requires memory speedup and a dynamic partitioning of the amount of memory for each input of the shared memory based on the VOQ occupancy. The mSMCB switch implements input-to-crosspoint matching to eliminate speedup in the shared crosspoint memory.

2.2.1 Shared-Memory Crosspoint Buffered Switch with Speedup m (SMCB$\times m$)

The SMCB$\times m$ switch has N VOQs at each input. A crosspoint in the shared-memory buffered crossbar that connects input port i to output j is also denoted as $XP(i,j)$. To minimize the speedup of the shared memory, the crosspoint buffer is only shared by two inputs. The buffer for $XP(i,j)$ and $XP(i',j)$, where $0 \leq i, i' \leq N-1$ and $i \neq i'$, that stores cells for output port j and is shared by these two crosspoints (or inputs i and i') is denoted as $SMB(q,j)$, where $0 \leq q \leq \frac{N}{2}-1$. An even N is assumed for the sake of clarity. However an odd N can be used.[3] For a switch where each crosspoint buffer is shared by m inputs, there are $\frac{N^2}{m}$ shared-memory crosspoint buffers in the buffered crossbar. The speedup of the shared crosspoint memory is m.

Figure 2.2 shows the architecture of the switch with two inputs sharing the buffered crosspoint. The arbitration scheme for both inputs and outputs is round-robin. The switch works as following. Each input sends its VOQ occupancy $Z_{i,j}$ to the crossbar. The sharing control unit (SCU) at every SMB sets up a threshold for partition of the shared memory, one for each input, based on the occupancy a VOQ has. This dynamically allocates the memory between the two inputs sharing the buffered crosspoint. The threshold sets the maximum value $C_{i,j}^{max}$of credit-based flow control counter at the input. In this architecture,

[3]For switches with odd number of ports, one port is left with dedicated buffers of size 0.5 the capacity of a SMB.

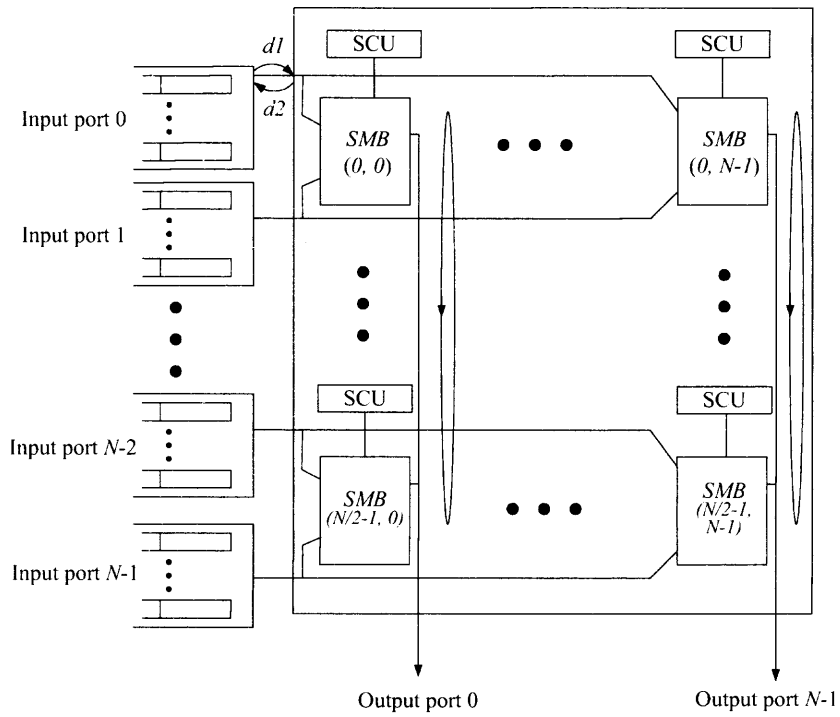

Figure 2.2 $N \times N$ shared-memory buffered crossbar switch with speedup m.

each shared crosspoint buffer size is k_s, where $k_s \geq RTT$. The threshold value based on the input occupancy of two input sharing the buffered crosspoint is presented in Table 2.1.

The occupancy of the two VOQs that share a crosspoint buffer is represented as $Z_{i,j}$ and $Z_{i',j}$. There are four occupancy states - when $Z_{i,j} = 0$, $Z_{i,j} \leq \frac{RTT}{2}$, $\frac{RTT}{2} < Z_{i,j} \leq RTT$ and $Z_{i,j} > RTT$. Since m inputs may need to access the shared memory at the same time, this architecture requires the shared memory have a speedup of m. To minimize the speedup, m is set to two in this dissertation.

2.2.2 Shared-Memory Crosspoint Buffered Switch with Input-Crosspoint Matching (mSMCB)

To eliminate the speedup at SMBs, only one input is allowed to access an SMB at a time. An input-access scheduler is used among the m inputs that share the same $SMB(q, j)$ to schedule the SMB access between two inputs that are physically separated. The size of an

Table 2.1 Threshold Setup of SMCB$\times m$ Switch

$Z_{i,j}$	$Z_{i',j}$	$C_{i,j}^{max}$	$C_{i',j}^{max}$
0	0	0	0
$[0, RTT]$	0	$Z_{i,j}$	0
$[RTT, \infty]$	0	RTT	0
$[0, RTT/2]$	$[0, RTT/2]$	$RTT/2$	$RTT/2$
$[RTT/2, \infty)$	$[0, RTT/2]$	$RTT - Z_{i',j}$	$Z_{i',j}$
$[RTT/2, \infty)$	$[RTT/2, \infty)$	$RTT/2$	$RTT/2$

SMB, in number of cells that can be stored, is k_s. There are $\frac{N}{m}$ input-access schedulers in the buffered crossbar. An input-access scheduler matches non-empty inputs to the SMBs that have room for storing a new cell. As shown in figure 2.3, there are $N/2$ input-access schedulers with $m = 2$.

The input-access scheduler performs a matching process among the shared-crosspoint buffers and the inputs that share them. Figure 2.4 shows the inputs and the shared crosspoint buffers that take place in the matching. The matching follows a three-phase process, as performed for IB switches. The matching scheme used in this switch is round-robin based [33] to have a valid comparison with the CIXB switch. However, any matching scheme can be used.

At each output j in the buffered crossbar, there is an output arbiter to select the outgoing cell from non-empty crosspoint buffers as shown in figure 2.3. The transmission delays between ports and the crosspoint are denoted by $d1$ and $d2$. An output arbiter can consider up to two cells from an SMB, where each cell belongs to one different input.

The way the mSMCB switch works is as follows. Cells destined to output j arrive at $VOQ(i, j)$ and wait for dispatching. Input i notifies the input-access scheduler about the new cell arrival. The input access scheduler selects the next cells to be forwarded to the

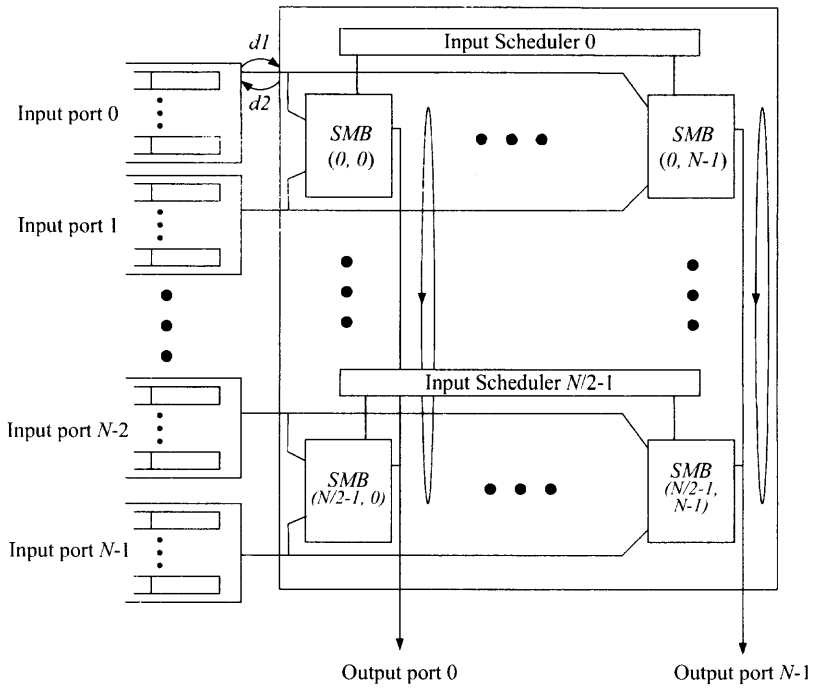

Figure 2.3 $N \times N$ Shared-memory buffered crossbar switch.

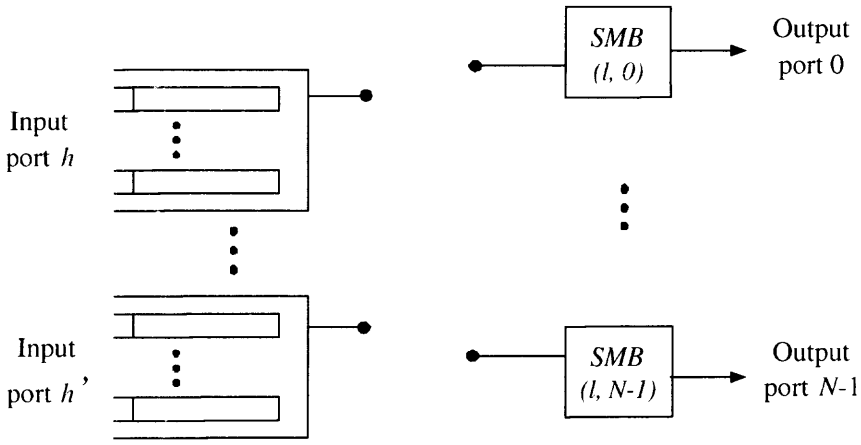

Figure 2.4 Bipartite matching in an input-access scheduler between inputs and SMBs.

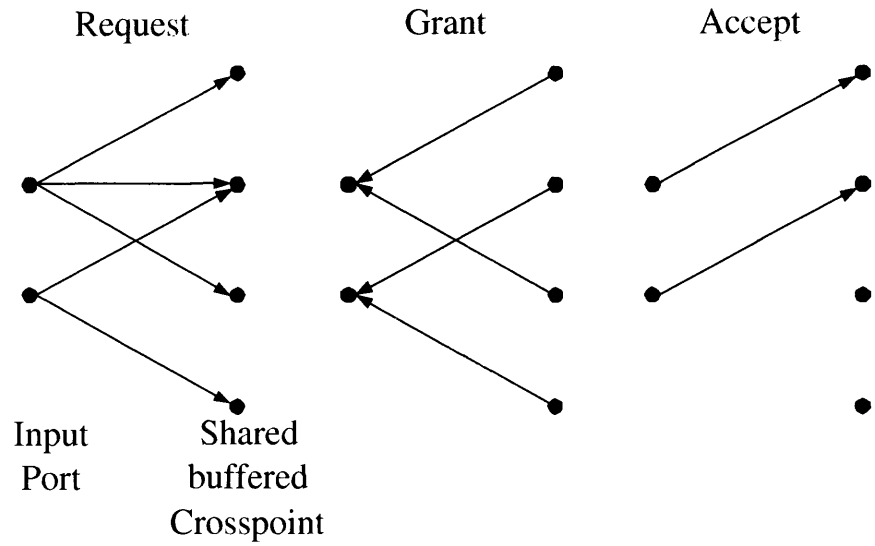

Figure 2.5 Example of the matching process in a 4×4 switch.

crossbar by matching the m inputs to the SMBs. Figure 2.5 shows an example of a 2×4 bipartite match as used in this input-SMB matching.

A cell going from input i to output j enters the buffered crossbar and is stored in $SMB(q, j)$. Cells leave output j after being selected by the output arbiter. The output arbiter uses round-robin selection.

A flow control mechanism that behaves like the credit-based flow control [1] is applied by input-access schedulers to avoid crosspoint-buffer overflow. Input access scheduler considers VOQs for which the SMBs that are not full as eligible. The input-access scheduler information is sent from the SMB to the corresponding VOQ (i.e., $d2$). Cells and flow-control data experience transmission delays between input ports and the buffered crossbar.

2.3 Performance Analysis of the SMCB Switch with Random Selection

In this section, the throughput of the mSMCB and SMCBxm switches with random-based selection schemes, under independent and identically distributed (i.i.d.) Bernoulli traffic at each input is analyzed. It is demonstrated that under these conditions, speedup is not

necessary. The emphasis is made on switches with $m = 2$, and it is shown that the 2SMCB switch provides equivalent performance to that of the SMCB$\times$2 switch. At the end, the case of $m = N$ for the mSMCB switch is analyzed to discuss the trade-off between switching performance and memory efficiency. The throughput of data switches with and without speedup is presented in [34] using fluid model approach.

The analysis focuses on the probability that a VOQ receives service. The service is defined in function of the flow of cells through the crosspoint buffer, which is defined by the service rate given by an output arbiter. Therefore, the analysis shows the effect that the matching process has over the switch performance.

In an SMCBx2 switch, SMBs are partitioned before the cells are forwarded from the input to the crosspoints, and therefore, a partition can be considered as a dedicated buffer for input i. A VOQ must be selected by the input arbiter to forward a cell to the buffer partition. It is considered that an SMB of size k_s stores one or more cells for the mSMCB switch and two or more cells for the SMCB$\times m$ switch. The probability that a buffer is full is denoted as P_f and the probability that a VOQ is blocked (from sending a cell to the corresponding crosspoint buffer) is denoted as P_b. The probability that a VOQ sends the HOL cell to the buffer is defined by the nonblocking probability of a VOQ, $1 - P_b$. For the sake of clarity, the variables for the mSMCB switch are represented with the superscript M and those for the SMCB$\times m$ switch with the superscript X.

2.3.1 mSMCB Switch with $k_s = 1$

The blocking probability of a VOQ in the mSMCB switch with $k_s = 1$, denoted as P_{b1}^M, is defined by two cases: a) when the SMB is full with probability $P_{fk_s}^M$, the blocking probability is the probability that there is cell to be forwarded to the corresponding VOQ. The probability that there is a cell destined to this specific output is $\frac{1}{N}$. b) When the SMB is available, there may be t inputs contending with a specific input, where $0 \leq t \leq m - 1$, requesting access to the SMB, and only one input is granted. Therefore, the probability that

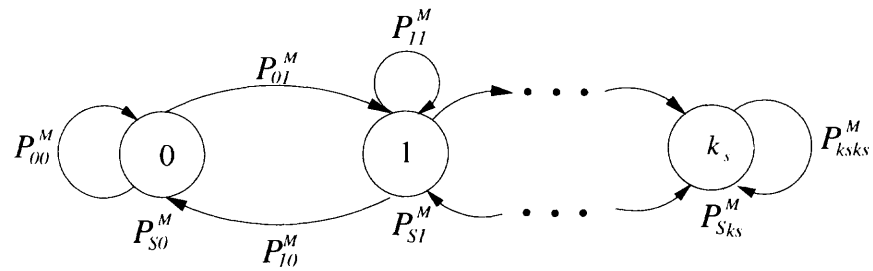

Figure 2.6 Diagram of the Markov chain of a buffer of size k_s.

an input receives no grant is $\frac{t}{t+1}$. Considering these two cases, the blocking probability for mSMCB switch is stated as

$$P_{b1}^M = \frac{1}{N}P_{f1}^M + \sum_{t=0}^{m-1}\binom{m-1}{t}(\frac{1}{N})^t(1-\frac{1}{N})^{m-1-t}\frac{t}{t+1}(1-P_{f1}^M). \tag{2.3}$$

A Markov chain is used to model $SMB(i,j)$ of the mSMCB switch [35] [36]. Figure 2.6 shows the Markov chain of an SMB of size k_s. P_{Sr}^M is the state probability where $0 \leq r \leq k_s$. $P_{r,u}^M$ is the transition probability from state r to state u. $P_{fk_s}^M$ is the probability that the SMB is full, and it is equivalent to the state probability P_{Sk_s}.

For any k_s, P_{01}^M is then defined by the product of input arrival rate $\rho_{i,j}$ and the matching probability between the input and the SMBs,

$$P_{01}^M = \rho_{i,j}\sum_{t=0}^{m-1}\binom{m-1}{t}(\frac{1}{N})^t(1-\frac{1}{N})^{m-1-t}\frac{1}{t+1} \tag{2.4}$$

The service rate $P_{service}^M$ is the probability the output arbiter selects a non-empty $SMB(i,j)$ for forwarding a cell to the output. It is easy to see that for $m=2$, $P_{service}^M = \frac{2}{N}$ for any state. P_{10}^M, which occurs when input i has no requests and $SMB(i,q)$ is selected by the output arbiter, is defined by

$$P_{10}^M = (1-\rho_{i,j})P_{service}^M. \tag{2.5}$$

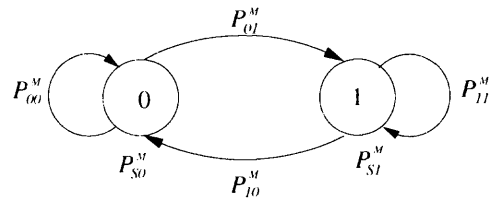

Figure 2.7 Diagram of the Markov chain of a buffer of size $k_s = 1$.

Figure 2.7 shows the state diagram when $k_s = 1$ for mSMCB switch. From this state diagram, the following two balance equations can be obtained,

$$\begin{cases} P^M_{01} P^M_{S0} = P^M_{10} P^M_{S1}; \\ P^M_{S0} + P^M_{S1} = 1; \end{cases} \tag{2.6}$$

and from these, it is defined that

$$P^M_{f1} = P^M_{S1} = \frac{P^M_{01}}{P^M_{01} + P^M_{10}}. \tag{2.7}$$

P^M_{00} is the probability that input i has no request. P^M_{11} is the probability that input i has no request and $SMB(i,j)$ is not selected by output arbiter or input i has request and $SMB(i,j)$ is selected by output arbiter to forward a cell. These two transition probabilities have no effect in our balance equations.

2.3.2 mSMCB Switch with $k_s = 2$

To analyze the performance of the two SMCB switches when they have the same amount of memory in the SMBs, k_s is set to two.

The probability that an SMB is full follows (2.3),

$$P^M_{b1} = \frac{1}{N} P^M_{f1} + \sum_{t=0}^{m-1} \binom{m-1}{t} (\frac{1}{N})^t (1 - \frac{1}{N})^{m-1-t} \frac{t}{t+1} (1 - P^M_{f1}). \tag{2.8}$$

Figure 2.8 shows the Markov chain diagram of the mSMCB switch when $k_s = 2$. From this figure, the following balance equations are obtained:

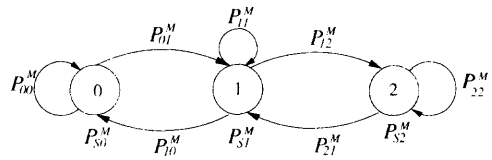

Figure 2.8 Diagram of the Markov chain of a buffer of size $k_s = 2$.

$$\begin{cases} P_{01}^M P_{S0}^M = P_{10}^M P_{S1}^M; \\ P_{12}^M P_{S1}^M = P_{21}^M P_{S2}^M; \\ P_{S0}^M + P_{S1}^M + P_{S2}^M = 1; \end{cases} \tag{2.9}$$

and from these equations,

$$P_{f_2}^M = P_{S2}^M = \frac{P_{01}^M P_{12}^M}{P_{01}^M P_{12}^M + P_{01}^M P_{21}^M + P_{10}^M P_{21}^M}. \tag{2.10}$$

Here, the transition probabilities are defined as:

$$\begin{cases} P_{01}^M = P_{12}^M = \rho_{i,j} \sum_{t=0}^{m-1} \binom{m-1}{t} (\frac{1}{N})^t (1 - \frac{1}{N})^{m-1-t} \frac{1}{t+1}; \\ P_{10}^M = (1 - \rho_{i,j}) \frac{1}{N-1}; \\ P_{21}^M = (1 - \rho_{i,j}) \frac{2}{N}; \end{cases} \tag{2.11}$$

2.3.3 SMCB$\times m$ Switch with $k_s = 2$

Without losing generality, let us assume that all VOQs have a backlog longer than RTT, such that in the SMCBxm switch, where an SMB is shared by m inputs (i.e., VOQs), an SMB is partitioned into m equally sized parts, one part for each VOQ. In this case, the blocking probability of a VOQ is in function of the occupancy of the allocated portion for an SMB, such that the portion is either full or available.

For the SMCB$\times$2 switch, the SMB is partitioned into two equally divided parts of size $\frac{k_s}{2}$. One part for each VOQ sharing this SMB. This is equivalent to having each VOQ with a designated crosspoint buffer with a variable buffer size. The blocking probability of a VOQ is determined by considering whether the allocated portion of the SMB is full or

available. Under uniform traffic, the average size of allocated portion of the SMB for each VOQ is equal to each other. P_b^X is represented as

$$P_b^X = \rho_{i,j} P_f^X. \tag{2.12}$$

The allocated portion with size of one cell in an SMB for the SMCB$\times$2 is also modeled as a Markov chain, as the model in Figure 2.6 shows for the mSMCB switch (where the superscripts Ms can be substituted by Xs). The state transition probability P_{01}^X for the SMCBx2 switch is the product of input arrival rate $\rho_{i,j}$ and the probability that the input arbiter selects $VOQ(i,j)$, $\frac{1}{N}$. Then,

$$P_{01}^X = \rho_{i,j} \frac{1}{N}. \tag{2.13}$$

The service rate $P_{service}^X$ is the probability that the output arbiter selects $SMB(i,j)$ to forward a cell to the output, and this is $\frac{1}{N}$. The transition probability P_{10}^X is the probability of an SMB being selected by the output arbiter while having no request from the input, or

$$P_{10}^X = (1 - \rho_{i,j}) \frac{1}{N}. \tag{2.14}$$

P_f^X is calculated as in (2.7):

$$P_f^X = P_{S1}^X = \frac{P_{01}^X}{P_{01}^X + P_{10}^X}. \tag{2.15}$$

2.3.4 Maximum Throughput of the 2SMCB and SMCBx2 switches with Random Selection

Considering the blocking probabilities of the VOQs in both switches, the throughput of each switch can be determined as i.i.d. traffic is assumed. Next, it is demonstrated that these two switches have similar performance for large N by looking at the blocking probability of VOQs.

The following equations show the limits of the P_b functions for the 2SMCB switch when k_s is 1 and 2 cells, respectively, and for SMCBx2 switch when k_s is 2 cells.

In the case for the 2SMCB switch with $k_s = 1$,

$$\lim_{N\to\infty} P_{f_1}^M = \lim_{N\to\infty} \frac{1 - \frac{1}{2N}}{3 - \frac{5}{2N}} = \frac{1}{3} \tag{2.16}$$

therefore,

$$\lim_{N\to\infty} P_{b1}^M = \lim_{N\to\infty} \{\frac{1}{N} P_{f1}^M + \frac{1}{2N}(1 - P_{f1}^M)\} = 0. \tag{2.17}$$

For the case where $k_s = 2$:

$$\lim_{N\to\infty} P_{f_2}^M = \lim_{N\to\infty} \frac{1 - \frac{1}{N} + \frac{1}{4N^2}}{3 - \frac{4}{N} + \frac{5}{4N^2} + \frac{2N}{N-1} - \frac{4}{N-1} + \frac{2}{N^2-N}} = \frac{1}{3} \tag{2.18}$$

then, the limit follows (2.16),

$$\lim_{N\to\infty} P_{b2}^M = \lim_{N\to\infty} \{\frac{1}{N} P_{f2}^M + \frac{1}{2N}(1 - P_{f2}^M)\} = 0. \tag{2.19}$$

For the SMCBx2 switch with $k_s = 2$, using (2.13), (2.14), and (2.15),

$$\lim_{N\to\infty} P_b^X = \lim_{N\to\infty} \frac{1}{N^2} = 0. \tag{2.20}$$

Therefore, because the minimum k_s values are addressed, it can be concluded that in all cases, the nonblocking probability, $1 - P_b$, for these two switches with any k_s value approaches 1 as N grows. Since the switch throughput is defined by the non-blocking probability, both switches achieve 100% throughput for a large N.

The reason why this high throughput is achieved with random selection is because the matching size in the 2SMCB switch is $2 \times N$, which means that there is an expansion gain, and because the speedup is two in the SMCBx2. However, in the case of the mSMCB switch with no expansion, or $m = N$, and $k_s = 1$, the performance of the mSMCB switch follows that of an IB switch using random selection.

Considering that the limit of the non-matching probability is,

$$\lim_{N\to\infty}\sum_{t=0}^{N-1}\binom{N-1}{t}(\frac{1}{N})^t(1-\frac{1}{N})^{N-1-t}\frac{t}{t+1} \tag{2.21}$$

$$=\lim_{N\to\infty}(1-\frac{1}{N})^N=0.368, \tag{2.22}$$

the probability that the SMB is full is:

$$\lim_{N\to\infty}P_{f_1}^M=\lim_{N\to\infty}\frac{(1-\frac{1}{N})^N}{(1-\frac{1}{N})^N+N-1}=0. \tag{2.23}$$

Therefore, the nonblocking probability when all N inputs share the SMBs converges to

$$1-\lim_{N\to\infty}P_b^M=1-(\lim_{N\to\infty}\{\frac{1}{N}P_f^M+(1-\frac{1}{N})^N(1-P_f^M)\})=0.632 \tag{2.24}$$

Therefore, the throughput of this switch is 63.2% (as in [37], [38] for an $N\times N$ IB switch with random selection. This proves that the performance of the mSMCB switch is dominantly defined by the matching scheme, and therefore when m is smaller than the switch size, the mSMCB switch delivers higher throughput performance. From this, while a larger m saves more crossbar memory, the throughput performance becomes compromised.

2.3.5 Non Blocking Probability for Small Switch Sizes

To observe the throughput of smaller switch sizes, the nonblocking probabilities of switches with sizes from 2 to 128 ports are computed as Figure 2.9 shows, using (2.7) and (2.15).

Figure 2.9 shows the nonblocking probability of a VOQ when $k_s=1$ for 2SMCB and $k_s=2$ for SMCB$\times$2 switch under uniform traffic. The result shows that the SMCB$\times$2 switch achieves higher throughput than the 2SMCB switch when switch size is small (when $N<16$). When switch size increases, the throughput of both switches approaches 100%.

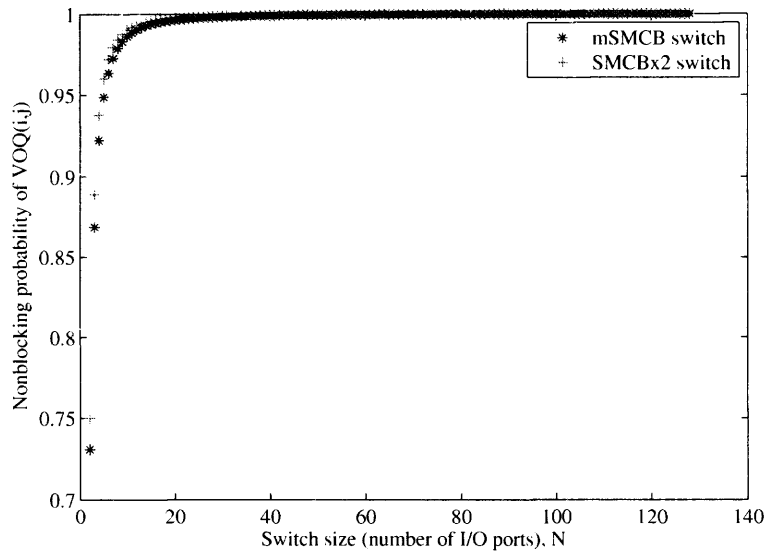

Figure 2.9 Calculated throughput of a VOQ under uniform traffic, $\rho = 1$, $k_s = 1$ for mSMCB switch, $k_s = 2$ for SMCB$\times$2 switch.

The throughputs of both switches are compared when they have the same amount of memory in the SMBs (i.e., the 2SMCB and SMCB$\times$2 switches with $k_s = 2$). Figure 2.10 shows the nonblocking probability when $k_s = 2$ for the 2SMCB and SMCB$\times$2 switches. The curves in this figure were generated using (2.7) for the SMCB$\times$2 switch and (2.10) for the 2SMCB switch. The results show that the throughput of the 2SMCB switch is higher than that of the SMCB$\times$2 switch when the switch size $N < 16$. However, when the switch size increases, the throughputs of both switches become similar, as these approach 100%.

2.4 Simulation of the SMCB Switch under Different Traffic Models

In this section the switching performances of two 32×32 switches is compared. It also presents the throughput performances of a CIXB switch and SMCB switches with buffered crosspoint shared by m inputs. The throughput and average cell delay under traffic with Bernoulli and bursty arrivals with uniform distribution, and the throughput under Bernoulli traffic with unbalanced distribution are studied.

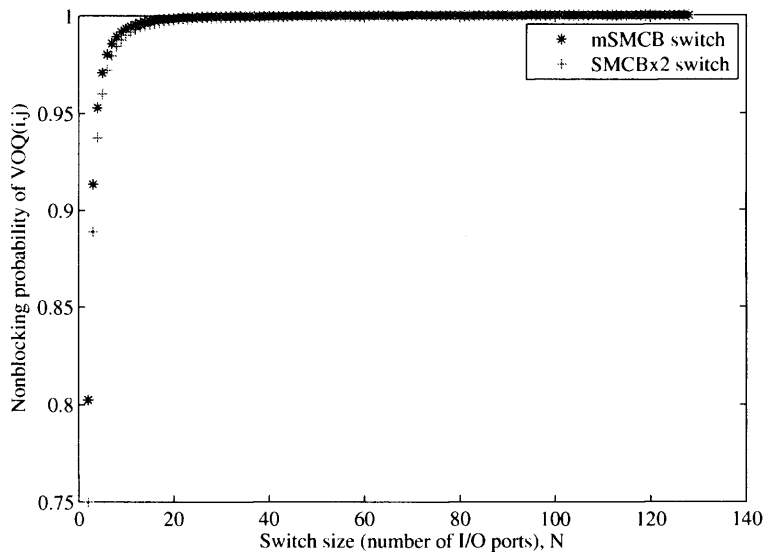

Figure 2.10 Calculated throughput of a VOQ under uniform traffic, $\rho = 1$, $k_s = 2$ for mSMCB and SMCB$\times$2 switch.

2.4.1 Performance of SMCB Switches with Round-Robin Selection

This section compares the switching performance of 32$\times$32 CICB, SMCB$\times$2 and mSMCB, all using round-robin selection schemes via computer simulation. It presents the throughput performance and average cell delay of all these switches. The considered traffic models have Bernoulli and bursty arrivals with uniform and nonuniform distributions. Simulation results are obtained with a 95% confidence interval, not greater than 5% standard error for the average cell delay.

2.4.1.1 Uniform Traffic. Figure 2.11 shows the average cell delay of the CICB, SMCB$\times$2, and 2SMCB switches under uniform traffic. Here, the crosspoint buffer size for CICB, k, is 2, and the crosspoint buffer size for the SMCB$\times$2 and 2SMCB switch, k_s, is also 2. By using these crosspoint buffer sizes, the amount of memory used in the SMCB switches is half the amount of memory of that in the CICB switch. Figure 2.11 shows that the average cell delay of the SMCB$\times$2 and 2SMCB switches are similar. Furthermore, the average cell delay of the SMCB switches shows similar magnitude to that of the CICB switch without

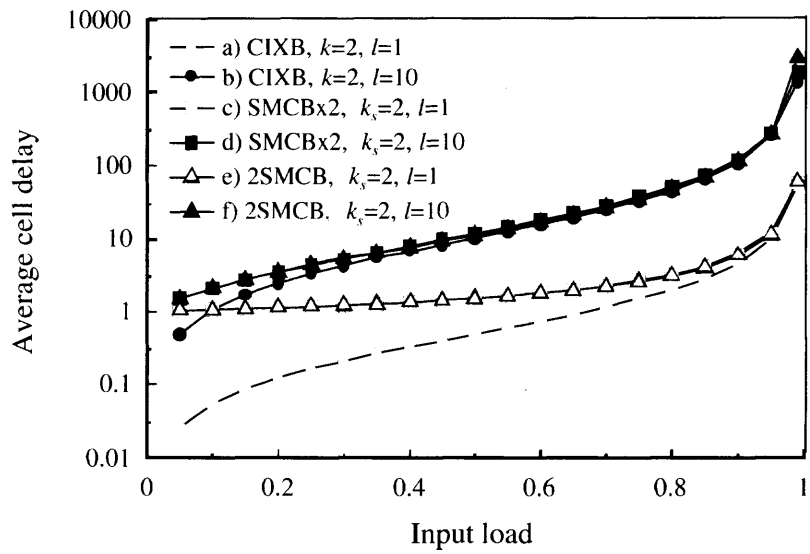

Figure 2.11 Average cell delay of 32×32 CICB, SMCB$\times$2, and 2SMCB switches with round-robin selection, under uniform traffic.

the effects of RTT (i.e, $RTT \leq 1$). The larger average delay shown by SMCB switches at loads from 0.1 to 0.8 are a constant time slot because the VOQs notify the input-access scheduler when a new cell arrives in SMCB$\times$2, and a VOQ is matched before forwarding a cell to the crossbar. These processes always take at least one time slot. For loads over 0.9, the SMCB switches have the average delay similar to that of the CICB switch. The results for bursty traffic, with the average burst length l of 10 cells, is similar for the three switches. It can be observed that these three switches have similar performance under uniform traffic. Therefore, the SMCB switches with $m = 2$ have equivalent performance under uniform traffic while using half memory amount of the CICB switch.

2.4.1.2 Nonuniform Traffic: Unbalanced Distribution. As described in Section 2.1, the unbalanced traffic model distributes the load of an input in two different parts: one fraction (i.e. w) to one output port and the rest to all output ports. This traffic model is becoming widely used because switches have been shown to be sensitive to this nonuniform distribution.

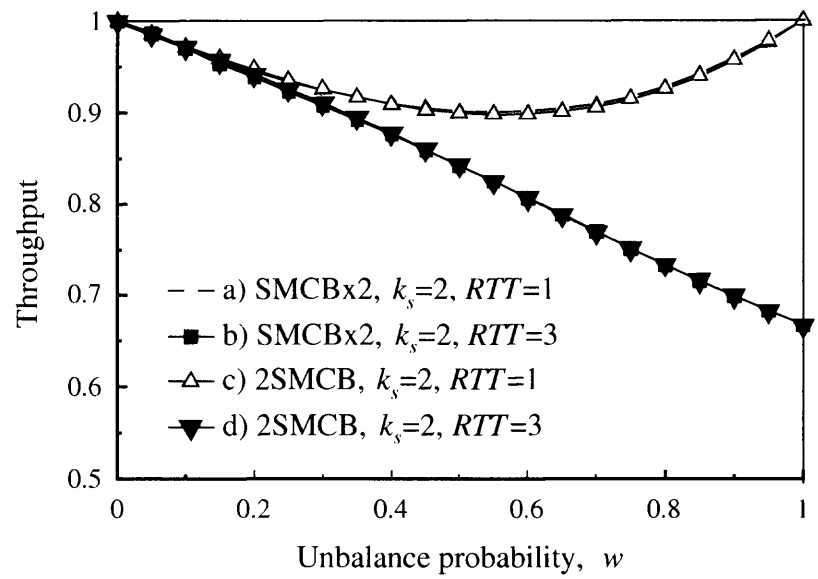

Figure 2.12 Throughput of SMCB×2 and 2SMCB switches with round-robin selection and the same amount of memory.

Figure 2.12 shows the throughputs of the SMCB×2 and 2SMCB switches when they have the same amount of memory, $k_s = 2$. The amount of memory used is the minimum as SMCB×2 can only work with $k_s \geq 2$, according to Table 2.1. This figure shows that these two switches have the same throughput performance for small and long RTTs, $RTT = 1$ and $RTT = 3$, respectively under unbalanced traffic. Because of this, the following simulations consider only the 2SMCB switch, unless otherwise stated. The effect of long RTTs on the proposed switches is observed by measuring the throughputs under the unbalanced traffic model, as in Section 2.1.

Figure 2.13 shows the throughput performance of the 2SMCB switch, with $k_s \geq 1$. This switch has a symmetric throughput when $w = 0.0$ and $w = 1.0$ or $r_{f(i,j)} = r^{max}_{f(i,j)} = r^{min}_{f(i,j)}$, and achieves 100% throughput for $k_s - RTT \geq 0$ when $w = 1.0$, as the figure shows in all curves, except for d) and f), which have $k_s - RTT < 0$ at this value of w. For $w = 1.0$, the throughput can be 100% using half of the total amount of memory used by the CICB switch.

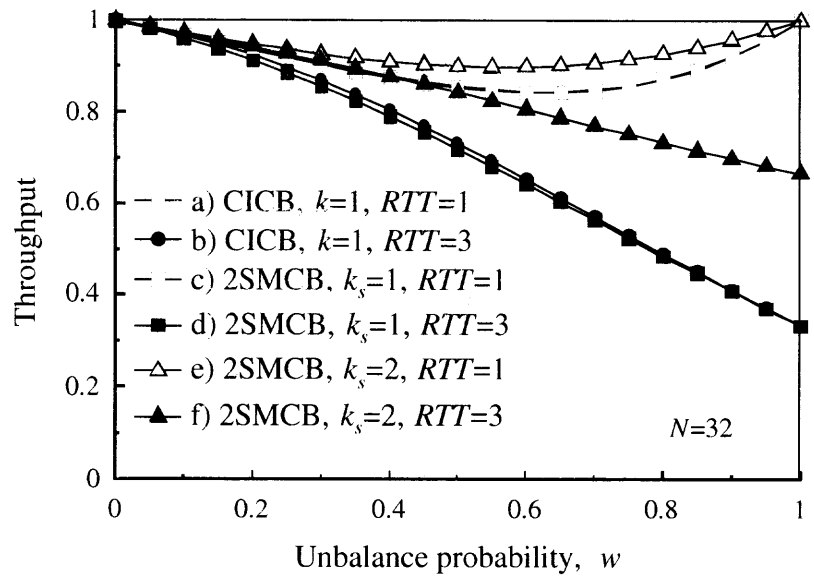

Figure 2.13 Throughput of a 2SMCB switch with half ($k_s = 1$) and equal ($k_s = 2$) amount of memory to the CICB switch.

For the other values of w, it is shown that for $k_s = k$ the throughput of 2SMCB is similar to that of the CICB switch. This is because the buffered crossbar switches have small sensitivity to the crosspoint buffer size under this traffic model. The decreased throughput around $w = 0.6$ in curves a), c), and e) where k_s, $k \geq RTT$, is the result of having a limited and small buffer size, mixed traffic (the high data-rate flow is mixed with a large number of low data-rate flows), as described in Section 2.1, and round-robin arbitration. In these cases, a more elaborated weightless [6] or a weight-based arbitration schemes can be used to improve the throughput for small $k_s - RTT$ values.

As seen in curves e) and f) of Figure 2.13, when the 2SMCB switch has the same amount of memory of the CICB switch (i.e., $k_s = 2k$) in the buffered crossbar, the throughput of the 2SMCB switch is higher than that of the CICB switch under the same RTT values.

Figure 2.14 shows the throughput of the 32×32 mSMCB switch with m inputs sharing the buffered crosspoint, where $1 < m \leq N$. With the same k_s value, the total

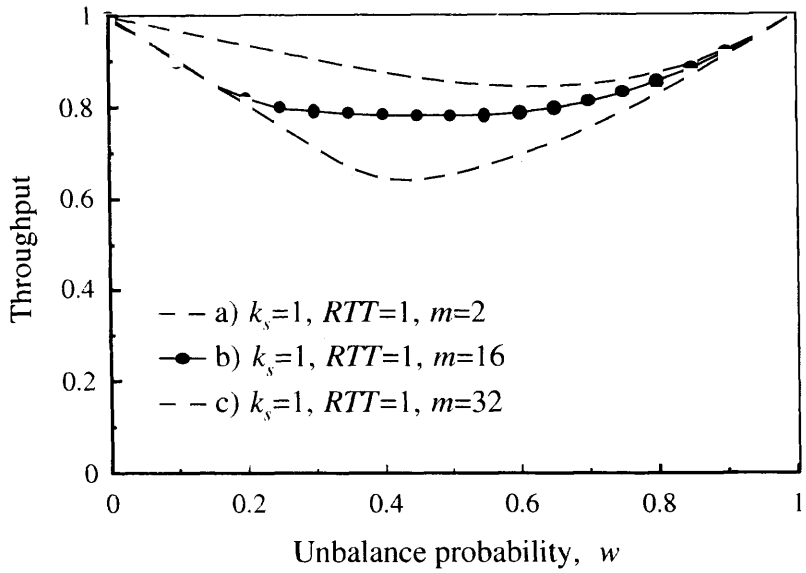

Figure 2.14 Throughput of the 32×32 mSMCB with m inputs sharing the buffered crosspoint.

amount of memory is $\frac{1}{m}$ of the CICB switch. The throughput of the mSMCB switch still achieves 100% when $w = 0.0$ and $w = 1.0$ for $m = \{2, 4, 8, 16, 32\}$. Furthermore, this figure also shows that for $m = 2$, the mSMCB switch achieves the highest throughput. The reason for that is that a 2-to-N matching process is more efficient than an N-to-N matching process for $N > 2$.

2.4.2 Nonuniform traffic: Diagonal and PO2 Distributions

To observe the performance of these switches under traffic models with more complex nonuniform distributions, the diagonal and power-of-two (PO2) traffic models are considered. The diagonal traffic model can be represented as $d\rho(i, j) = d\rho_i$ for $i = j$, $(1 - d)\rho_i$ for $j = (i + 1) \bmod N$, where ρ_i is the load at input i.

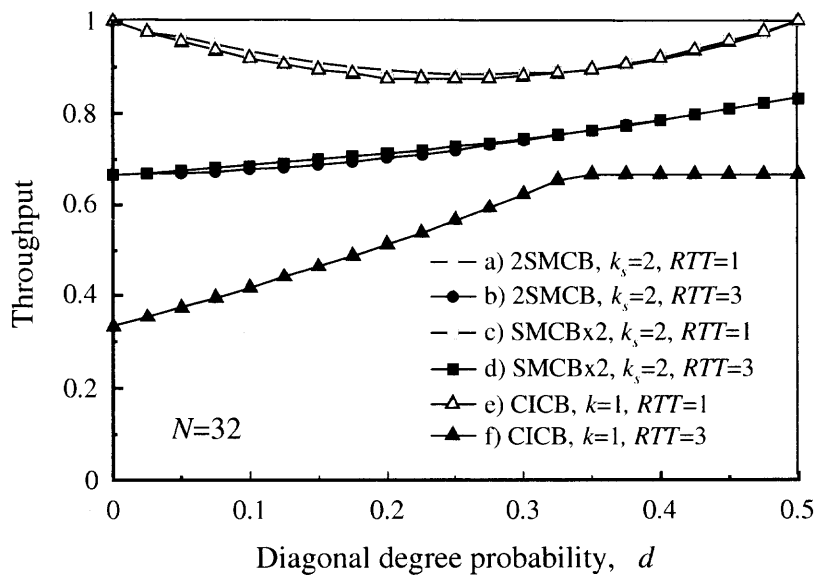

Figure 2.15 Throughput of 32×32 2SMCB, SMCB$\times$2, and CICB switches with round-robin selection, under diagonal traffic.

or by the matrix $\bar{\rho}$ as:

$$\bar{\rho} = \rho \begin{pmatrix} d & (1-d) & \dots & 0 \\ \vdots & \ddots & & \vdots \\ (1-d) & \dots & 0 & d \end{pmatrix}$$

This traffic model distributes the load of an input among two outputs only. The distribution is given by the diagonal degree probability, d.

Figure 2.15 shows the performance of CICB, 2SMCB, and SMCB$\times$2 under this traffic type, for $0 \leq d \leq 0.5$. This figure shows that all three switches, with $k_s = 2k$, or the same amount of memory, have a throughput close to 90% when RTT is small (e.g., $RTT = 1$), such that RTT does not contributes to the performance degradation. In this case, the round-robin selection used in all switches limit the throughput to this levels. For long RTTs (e.g., $RTT = 3$), the throughput of SMCB$\times$2 and 2SMCB is twice as higher as that of the CICB switch. Therefore, under diagonal traffic and with round-robin selection, the SMCB switches perform better than the CICB switch.

To examine the performance of these switches with a traffic model that distributes the input load in different proportions to each output, PO2 traffic is used.

This traffic model can be represented by matrix $\bar{\rho}$ as:

$$\bar{\rho} = \rho \begin{pmatrix} \frac{1}{2^1} & \cdots & \frac{1}{2^N} \\ \vdots & \ddots & \vdots \\ \frac{1}{2^N} & \cdots & \frac{1}{2^{N-1}} \end{pmatrix}$$

For example, the PO2 traffic load of a 4×4 switch is represented as:

$$\bar{\rho} = \rho \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{4} & \frac{1}{8} & \frac{1}{16} & \frac{1}{2} \\ \frac{1}{8} & \frac{1}{16} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{16} & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} \end{pmatrix}$$

Different from the diagonal traffic model, the PO2 traffic model presents a large degree of nonuniformity in the traffic distribution among N possible destinations. The maximum throughput of CICB, 2SMCB, and SMCB$\times$2 switches under PO2 traffic is measured. CICB achieved 95% throughput, while 2SMCB and SMCB$\times$2 achieved 80% under small RTTs. One of the reasons of this low throughput is that round-robin selection limits the performance under this nonuniform traffic model.

2.4.3 Performance of SMCB Switches with Longest-Queue-Occupancy First (LQF) Selection

To improve the performance of the proposed SMCB switches, LQF is used for the input selection instead of round-robin in all three switches, and tested their performance under uniform, unbalanced. diagonal, and PO2 traffic. LQF is implemented in the input arbitration of CICB and SMCB$\times$2, and in the input access scheduler of 2SMCB. In the 2SMCB switch, it is considered that the maximum number of iterations performed by the matching

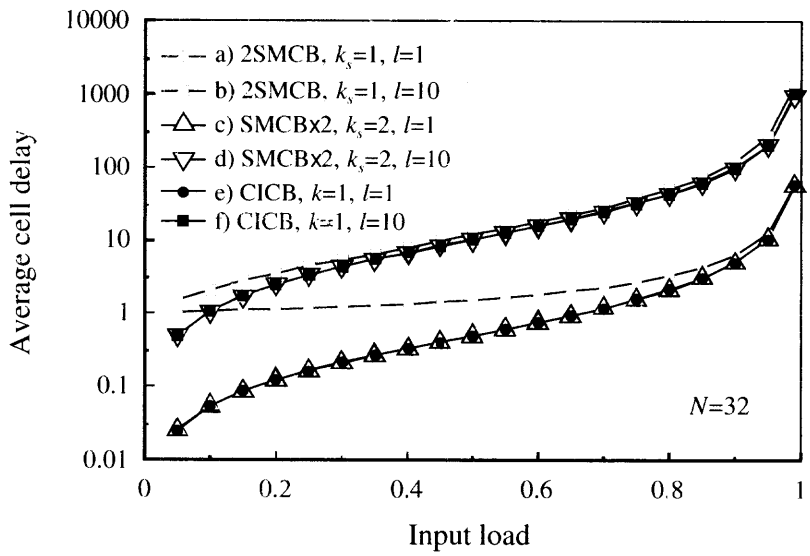

Figure 2.16 Average cell delay of 32×32 2SMCB, 2SMCB and CICB switches with LQF selection, under uniform traffic.

process is $m = 2$ to observe the maximum performance. The output arbiters of these three switches use round-robin selection as otherwise, it would be needed to transmit the VOQ occupancy to the output arbiters, and this may be impractical for implementation. LQF selects the VOQ with longest occupancy at the input port to forward cells to crosspoint buffers.

Uniform Traffic

Figure 2.16 shows the average cell delay of the 2SMCB, SMCB$\times$2, and CICB switches under uniform traffic. The result is similar to those using round-robin selection, shown in Figure 2.11, with however, smaller delay when the input load approaches to 1.0.

Nonuniform Traffic

The three switches with LQF were also tested under unbalanced, diagonal, and PO2 traffic models. Figure 2.17 shows the throughput of these three switches under unbalanced traffic. When $k_s = 2$ and $k = 1$ and small RTTs (e.g., $RTT = 1$), as in curves c), e),

and g), the throughput of CICB and SMCB$\times$2 is 100%, while the throughput of 2SMCB is 97%. The throughput of 2SMCB is lower than the other two because the matching process is limited by the small crosspoint buffer size as those SMBs that are full become ineligible for matching. In addition, the speedup used in SMCB$\times$2 improves the throughput for this small RTT. When RTT is long, (e.g., $RTT = 3$), the throughputs of SMCB$\times$2 and 2SMCB are higher than that of the CICB switch when the switches have the same memory amount. In other words, when $RTT = 3$, the SMCB switches achieve the same performance of that of the CICB switch with half the memory amount in the crossbar and twice the throughput of that of the CICB switch with the same amount of memory. The SMCB switches with LQF selection also achieve 100% throughput with half the memory amount of the CICB switch when input ports handle a single flow with a rate equal to the port capacity, as seen when $w = 1.0$ in this figure, while the CICB switch reaches a throughput of $\frac{k}{RTT}$, or $\frac{1}{3}$ in this case.

Figure 2.18 shows the throughput of these three switches under diagonal traffic. When $k_s = 2$, $k = 1$ and $RTT = 1$, the throughput of CICB and SMCB$\times$2 is higher than that of 2SMCB. Under this scenario, it shows that the throughput of CICB and SMCB$\times$2 is 100%, while that of 2SMCB is 88%. When RTT becomes larger than the crosspoint buffer size, the throughput of CICB becomes 33% (for $d = 0.0$), and the throughput of SMCB$\times$2 and 2SMCB becomes 66% each. Under this value of d, each flow has a port-speed rate, and hence the poorest performance of CICB, and the SMCB switches have twice the throughput of the CICB switch using the same amount of memory. When $d = 0.5$, the flows going through the switch have half the port speed, and therefore, the throughput is twice as higher as that when $d = 0.0$. This scenarios are consistent with those observed under the unbalanced traffic. In addition, 2SMCB and SMCB$\times$2 have the same throughput for long RTTs (e.g., $RTT = 3$), and therefore, the contribution of the speedup becomes negligible under this nonuniform traffic pattern and RTT value.

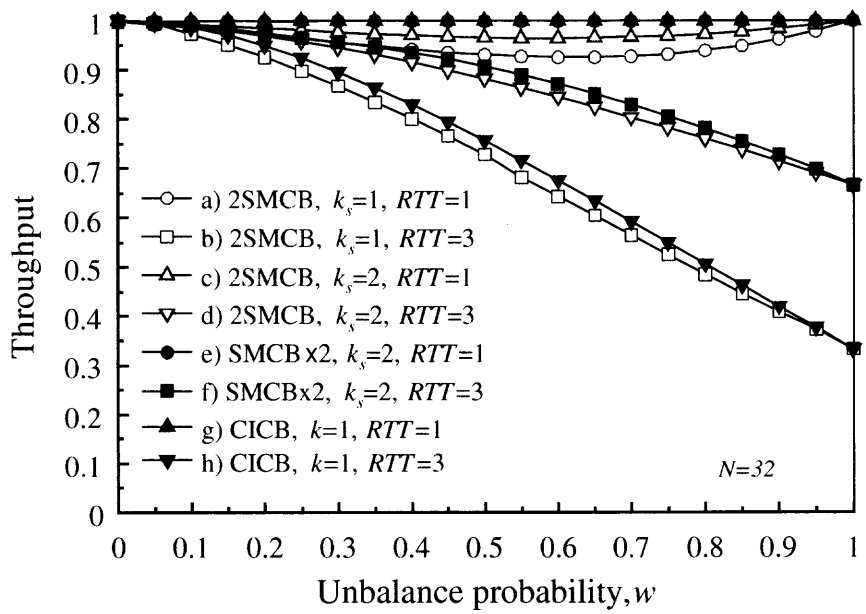

Figure 2.17 Throughput of 32 × 32 2SMCB, SMCB×2, and CICB switches with LQF selection, under unbalanced traffic.

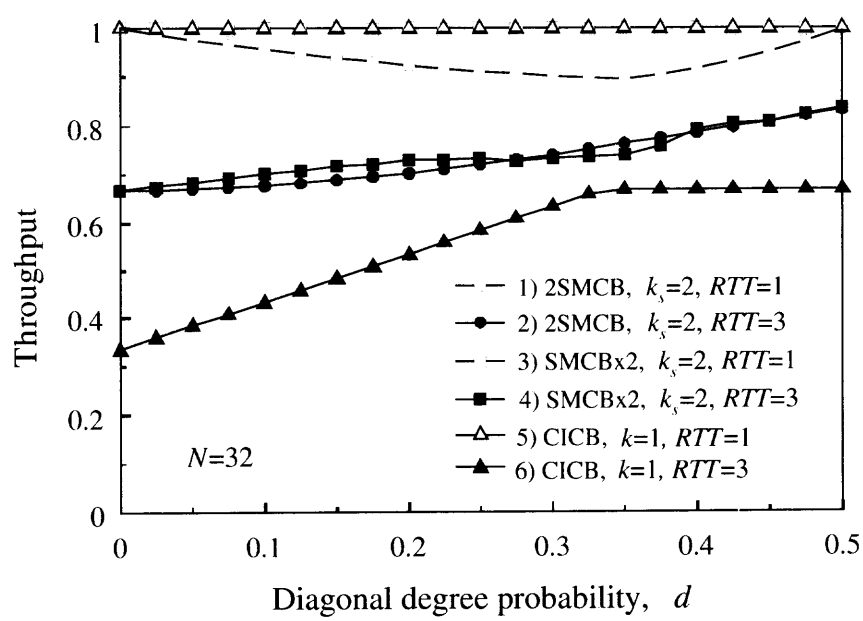

Figure 2.18 Throughput of 32 × 32 2SMCB, SMCB×2, and CICB switches with LQF selection, under diagonal traffic.

2.4.4 Throughput under Long Round-Trip Times and Nonuniform Traffic

This section shows a practical example of the advantages of the SMCB switches. Considering a small k for the CICB switch and small k_s for the SMCB switches, such that $k_s = 2k$, the delivered throughput for $RTT = 7$ and $k_s = \{2, 4, 6, 8, 10, 12\}$ under diagonal and PO2 traffic is measured . Figures 2.19 and 2.20 show the throughput obtained vs. the amount of memory in the 2SMCB, SMCB$\times$2 and CICB switches with an input load of 1.0. Figure 2.18 shows the throughput of 32×32 2SMCB, SMCB$\times$2, and CICB switches under diagonal traffic when the diagonal probability is $d = 0.25$. Each set of three columns indicates the throughput of the switches when they have the same amount of memory. The figure shows that as the crosspoint buffer size increases, the throughput also increases, and that the 2SMCB and SMCB$\times$2 switches deliver higher throughput than the CICB switch for $k_s \leq 8$. The higher throughput of the CICB switch is higher than the others when $k_s = 2k \geq 9$. This figure also shows that 2SMCB has better performance than SMCB$\times$2 and therefore, speedup is not needed for $k_s \leq RTT$. The higher throughput of SMCB$\times$2 for $k_s > 8$ becomes unattractive as in that case a CICB switch with $k = 4$ would be more effective than having speedup in SMCB$\times$2.

Another example is shown in Figure 2.20, which depicts the throughput of these three switches under PO2 traffic when $RTT = 7$. Here, the 2SMCB and SMCB$\times$2 switches achieve better throughput than the CICB switch for any k_s size. The throughput of the SMCB$\times$2 switch delivers slightly higher performance than the 2SMCB switch because of the memory speedup.

2.5 Implementation of an mSMCB Switch

The concern of using shared memory buffers has been memory speedup and implementation complexity. In the previous section, it has been shown that mSMCB switch does not need memory speedup to achieve high performance. This section presents the implementation of an mSMCB switch when $m = 2$ and shows that memory access time is not affected

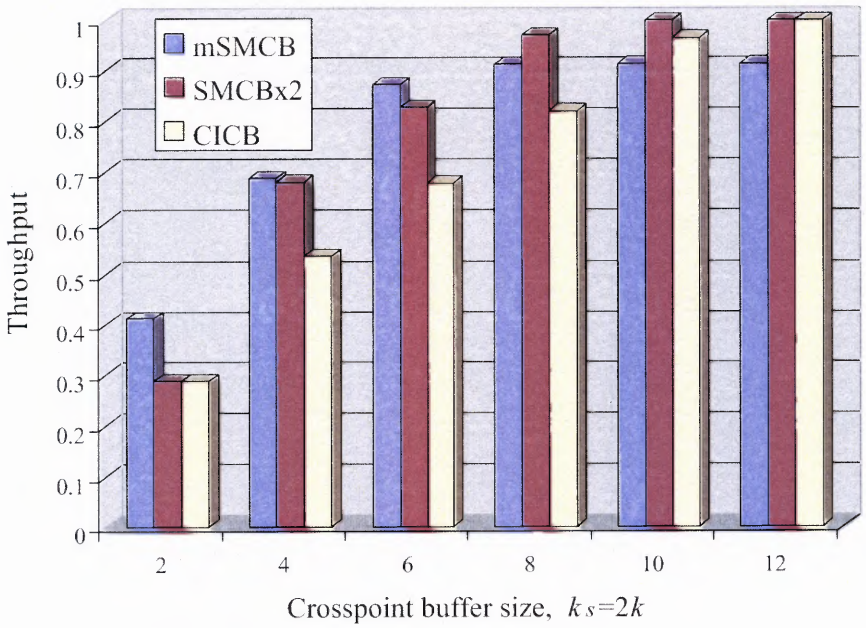

Figure 2.19 Throughput of 32×32 2SMCB, SMCB$\times$2, and CICB switches with LQF selection, under diagonal traffic when $d = 0.25$ and $RTT = 7$.

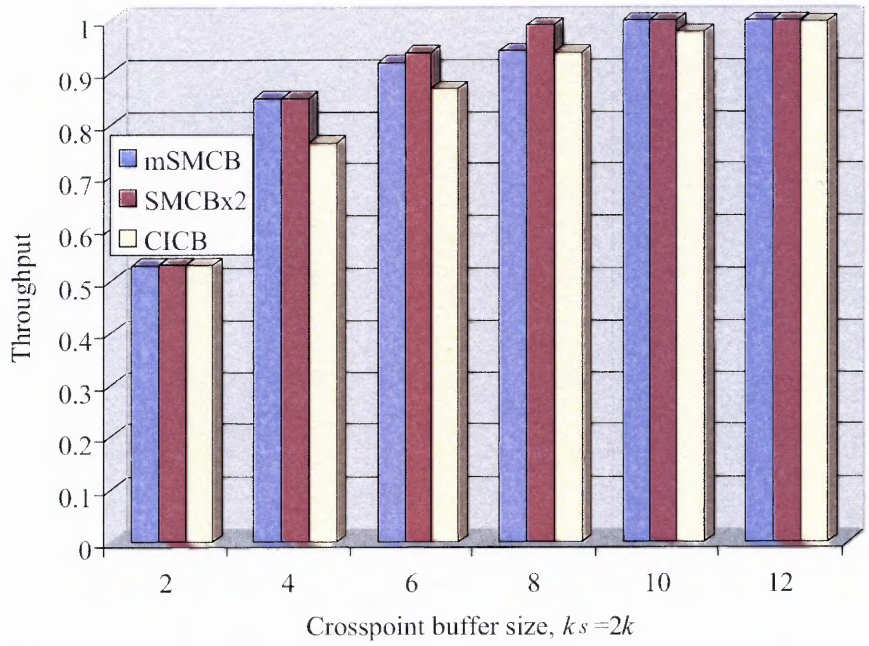

Figure 2.20 Throughput of 30×30 2SMCB, SMCB$\times$2, and CICB switches with LQF selection, under PO2 traffic and $RTT = 7$.

by the implementation. Figure 2.21 shows the block diagram of the design of the control path of the switch fabric of an mSMCB switch. The address decoder decodes the address of the destination address of the cell and the request that comes along with the cell. The request is sent to the input access scheduler to be considered for matching while the cell is sent to the designated buffer at the crossbar. The shared memory buffered is implemented using two linked list logical-queues as shown in Figure 2.22 [4]. It is shown that no additional memory is needed in the linked list logical-queue implementation of the shared memory.

Figure 2.23 shows the timing diagram of the mSMCB switch's operation. The subscript in the diagram represents the timestamp of a cell. There are five stages in the timing diagram: 1) output arbitration (OA) time, 2) time for matching at the input access scheduler, 3) transmission time of cells from input to crosspoint buffer (I) and cell from crosspoint buffer to output (O), 4) transmission time of cell C_i and request of cell C_{i+1}, 5) transmission time of cell C_i and grant C_{i+1} generated by the input access scheduler. At time slot i, cell C_i is scheduled to be forwarded to the shared crosspoint buffer, and the request from this input for the next time slot $i + 1$ is also sent along this cell to the input access scheduler. Cell C_i and request C_{i+1} go through transmission delay on the transmission line between input line cards and the switch fabric. Cell C_i is written into the memory of SMB and wait to be forwarded to output by the output scheduler. Meanwhile, input access scheduler makes decision to grant access to one of the inputs that share the SMBs. The grant is patched to the cell scheduled to go to that port as an output. Assume no competition from other SMBs for this output, Cell C_i is forwarded to the output along with a grant to that port. Cell C_i and the grant info go through transmission delay from the switch fabric to the line cards. The matching process at input access scheduler and OA are designed to give results within one time slot. Transmission time in time slots between input line cards and switch fabric depends on the switch port speed and the distance. Of the five stages, the longest process time is the memory access time of the SMBs which

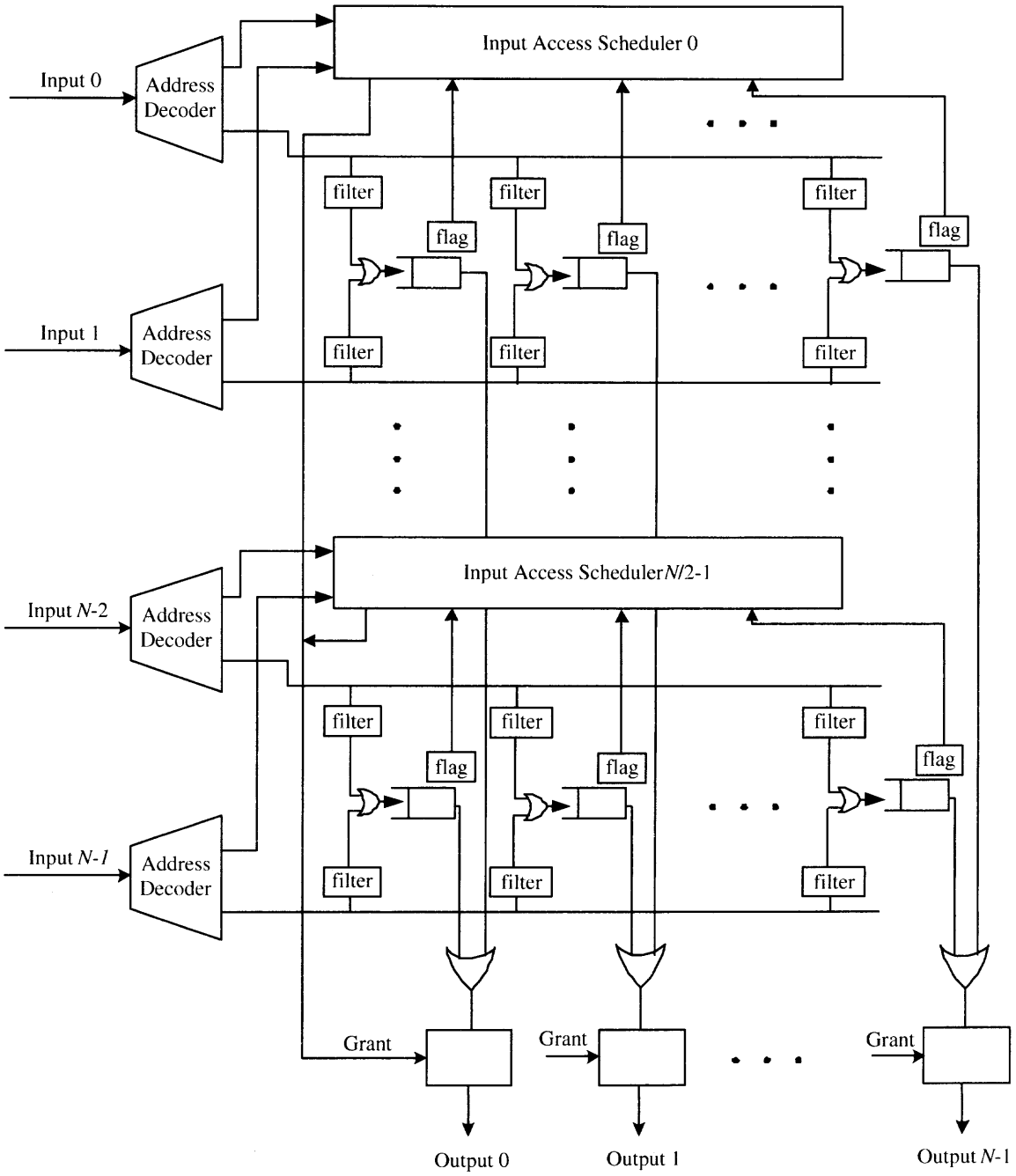

Figure 2.21 Implementation of the mSMCB Switch.

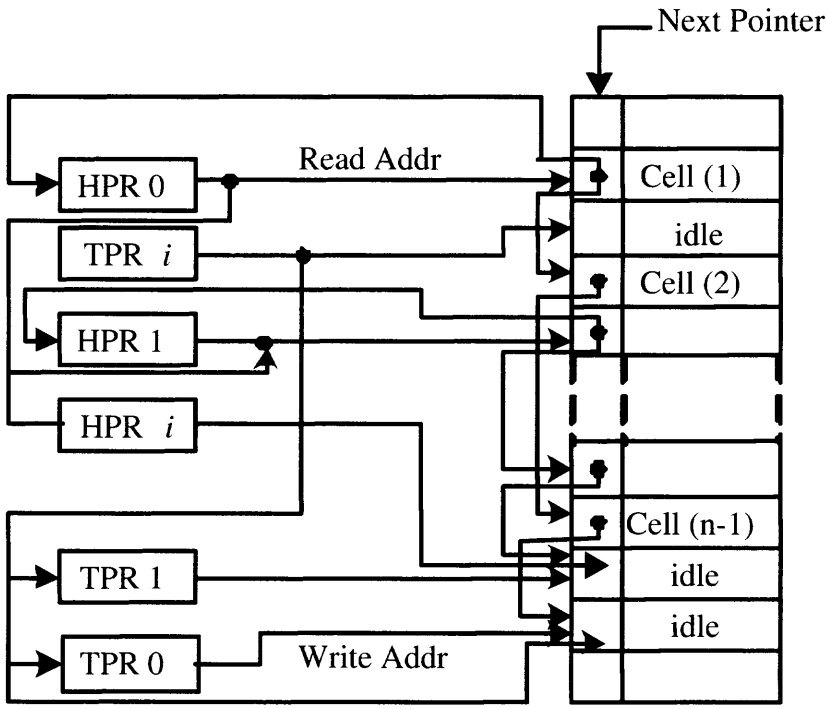

Figure 2.22 Logical-queue implementation of shared memory buffers of the mSMCB switch.

is independent of the design. In figure 2.23, each stage takes one time slot. The switch can be implemented using a five-stage pipelined system with processing time for one stage $\tau = max(\tau_i), i = 1, 2, ..., 5$, where τ_i is the processing time on stage i.

2.6 Conclusions

The effect of long RTTs is addressed, where the crosspoint buffer size k is such that $k < RTT$, in a CICB switch. It is shown that switches based on buffered crossbars with dedicated crosspoint buffers, as in [1], have their maximum throughput defined by the ratio of $\frac{k}{RTT}$ when input ports handle single flows with a data rate equal to the port capacity. To minimize the crosspoint-buffer size, two switches that share the crosspoint buffers among m inputs are proposed such that RTT can be m times longer than that supported by a CICB switch with dedicated buffers without decreasing significantly the switching performance, and providing 100% throughput for port-rate flows and under uniform traffic. The main dif-

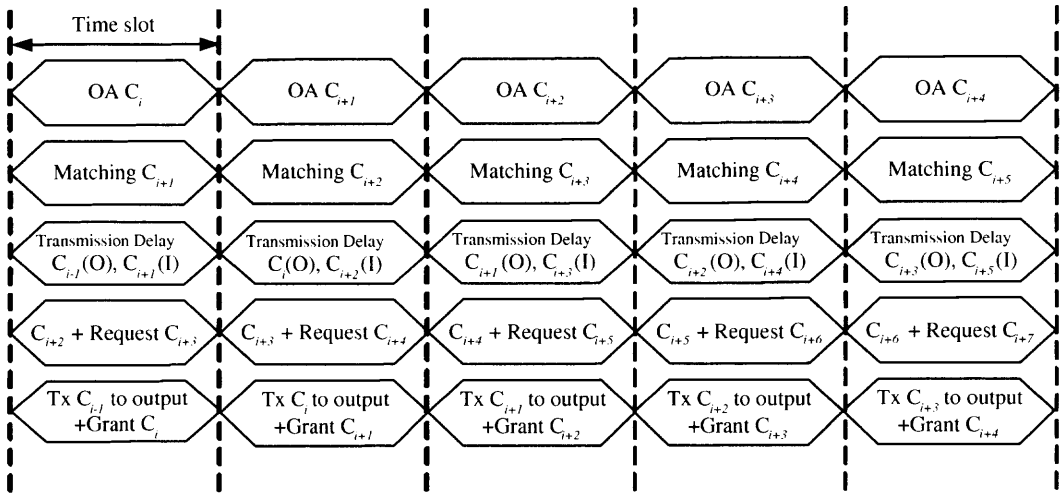

Figure 2.23 Timing diagram of the mSMCB switch's operation.

ference between the proposed switches, mSMCB and SMCB$\times m$, is that the former uses no memory speedup while the second uses a speedup of m. The maximum throughput of these two switches with random-based arbitrations and under independent and identical traffic with uniform distributions is analyzed. It is demonstrated that both switches achieve 100% throughput under this traffic model, such that speedup can be considered not necessary for such performance. The performance of these two switches is studied via computer simulation and the maximum throughput under uniform and nonuniform traffic models and under long RTTs is presented. These switches deliver comparable performance to a CICB switch when RTT is small, and better performance than a CICB switch when RTT is long. The SMCB switches with round-robin selection schemes relax the amount of required memory to $\frac{1}{m}$ of the amount required by a CICB switch under port-speed flows. It is shown that the highest performance is achieved when $m = 2$ for the mSMCB switch. In addition, the performance study shows that the proposed switches provide higher throughput than CICB switch with the same amount memory under traffic with large nonuniformity and long RTTs.

When a weight-based selection scheme is used, such as LQF, for input selection, the SMCB switches, with $k_s \leq RTT$, delivers higher performance than SMCB switch using

round-robin selection, and than a CICB switch with LQF input arbitration under nonuniform traffic and long RTTs. Furthermore, the simulation results show the advantage of using speedup in the SMCB$\times$2 switch for large k_s values (e.g., $k_s \geq RTT$) only. Considering the cost of speedup, the mSMCB switch presents attractive performance at low memory cost. However, the matching process used in mSMCB limits the maximum throughput performance under traffic with nonuniform distributions and small RTT values.

The proposed switches might also be used in cases where the crosspoint buffer size is smaller than RTT and flows with small rates are expected with, however, sudden cell bursts. SMCB switches are less sensitive to a burst as this case is similar to having high rate flows. Therefore, the expected throughput degradation of an SMCB switch is smaller than that of a CICB switch for the duration of the burst.

As future work, it is of interest to find alternative ways to deal with nonuniform traffic in SMCB switches. One way would be to share the crosspoint buffers in a dynamic fashion such that the crosspoint with demand for larger space is granted buffer access at higher rates. Also, as this study also shows that CICB switches are over-provisioned memory-wise, and more efficient switches can be obtained. Therefore, more research can be done in this direction. This issue is particularly important considering that high-speed memory is expensive and therefore, the amount of it needs to be small for practical implementations of high-performance switches.

CHAPTER 3

DIFFERENTIATED SERVICE SUPPORT BY SMCB SWITCHES

3.1 Introduction

As more voice and video traffic emerge along with data traffic in the Internet nowadays, the demand for switches that can provide quality of service (QoS) has increased. Integrated services (IntServ) and differentiated services (DiffServ) are well-known alternatives for QoS provisioning. IntServ focuses on per-flow QoS, which increases processing complexity and decreases scalability of the network. DiffServ focuses on bulk-flow QoS, which reduces complexity and increases scalability of the network. As a solution that combines the advantages of both approaches, service vector networks have been proposed [39]. Service vectors use a nested DiffServ model where several QoS groups are considered to increase class granularity beyond those offered by DiffServ. Here, each node in a network can flexibly select a different service level for a flow as long as the average service of the complete path satisfies the service requirements [40]. In service-vector networks, routers and switches are required to support multiple priority traffic. Considering P service classes $S = (S_0, S_1, \ldots, S_{P-1})$ provided by each link in a network, routers and switches in the network need to provide P classes of service (Class 0, Class 1, ..., Class $P-1$).

Combined input-crosspoint buffered (CICB) switches, which have dedicated crosspoint buffers, are introduced to relax arbitration timing while providing high-performance switching and supporting high-speed ports [21]. For a CICB switch, the required amount of memory in a buffered crossbar to avoid buffer underflow for high-rate flows is $N^2 \cdot k \cdot L$ bytes, where N is the number of input/output ports, k is the crosspoint buffer size in number of cells, and L is the cell size in bytes. The throughput of a CICB switch is dependent on $\frac{k}{RTT}$ ratio and the input load, ρ [28].

In a CICB switch, the crosspoint-buffer size, k, must be at least RTT cells long to avoid underflow [1]. Furthermore, as the buffered crossbar can be physically located far from the input ports, actual RTTs can be long. To support long RTTs in a buffered-crossbar switch, the crosspoint-buffer size needs to be increased, such that at least RTT cells can be buffered. However, the memory amount that can be allocated in a chip is limited. This can make the implementation costly or infeasible when the distance between line cards and the buffered crossbar is long, or else to provide a size of k cells where $k < RTT$ with service degradation. The effect of long RTT is studied in detail in the previous chapter and [29].

In this chapter, the focus is on providing efficient switching for differentiated services, where traffic is assigned different switching priorities while reducing the amount of memory of a switch and supporting long RTTs. Here, It is considered that high priority cells belong to delay sensitive traffic such as voice and video applications, which need to be served prior to other traffic classes to minimize queueing delay within the switch. An interesting scheme using limited memory is presented in [30] for a switch with P traffic classes, where the crosspoint buffer size is larger than RTT for a single class, and smaller than $P \times RTT$.

Because multiple traffic priorities increases the required amount of memory at buffered crossbar, this chapter proposes a CICB switch that keeps the amount of required memory low and that can be used in service-vector networks.

To reduce the memory amount while supporting long RTT values and multiple priority traffic, the study of SMCB$\times m$ switch is extended to support multiple classes of traffic. It is shown that the SMCB switch supports a given RTT with half or less memory than that of a CICB switch, and that of the proposed switch delivers equivalent or better switching performance for different classes of traffic than a CICB switch. It is also shown that no speedup is needed to achieve high throughput when using the shared-memory approach and the equivalent amount of memory in the crossbar.

3.2 CICB Switch with Dedicated Crosspoint Buffers for Differentiated Services

In this section, the architecture of CICB switch with dedicated crosspoint buffers is extended to support multiple traffic classes. The CICB switch architecture follows the one presented in [1]. It is considered that traffic is classified into P different classes. A buffered crossbar has N inputs and outputs. A crosspoint (XP) element in the buffered crossbar that connects input port i to output port j is denoted as $XP(i,j)$. There are $N \times P$ VOQs at each input. A VOQ at input i that stores cells for output j of priority p, where $0 \leq p \leq P-1$, is denoted as $VOQ(i,j,p)$. Here, $P = 3$ is considered for sake of clarity. The XP buffer of $XP(i,j)$ is denoted as $XPB(i,j)$, and this is not prioritized. The size of $XPB(i,j)$ is k cells, where $k \geq 1$. Figure 3.1 shows the switch architecture.

A credit-based flow control mechanism is used to notify at input i whether $XPB(i,j)$ has room available for a cell or not. Each VOQ has a credit counter, where the maximum count is the number of cells that $XPB(i,j)$ can hold. When the number of cells sent by $VOQ(i,j,p)$ reaches the maximum count, the VOQ is considered not eligible for input arbitration and overflow on $XPB(i,j)$ is avoided. The count is increased by one each time a cell is sent to $XPB(i,j)$ and decreased by one each time that $XPB(i,j)$ forwards a cell to output j. If $XPB(i,j)$ can receive at least one cell, then $VOQ(i,j,p)$ is considered eligible by the input arbiter. Prioritized scheduling for input-queued switches is introduced in [41].

An input arbiter at input i selects $VOQ(i,j,p)$ among the eligible VOQs to send a cell to XPB for output j at buffered crossbar. An output arbiter at output port j in the buffered crossbar selects a $XPB(i,j)$, among occupied XPBs from input i, to send a cell to output j. The eligibility of VOQs is determined by the flow control mechanism.

Different from the switch in [1], this CICB switch uses round-robin with strict priority as the input arbitration scheme. In this scheme, round-robin selection is performed among all queues of the highest priority. If there are no cells of the highest priority, round-robin is performed among queues of the second highest priority, and so on. The output

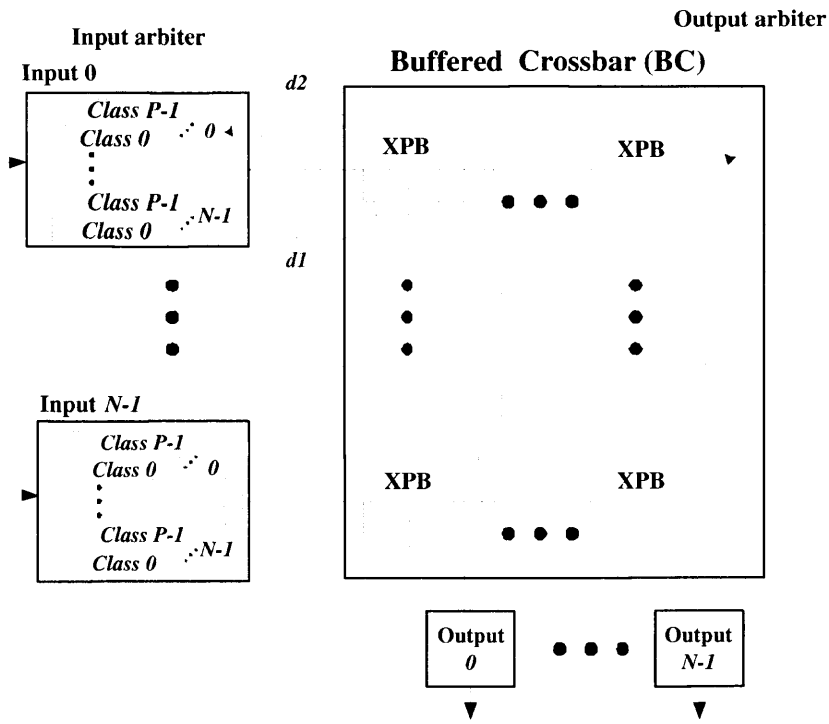

Figure 3.1 CICB switch with dedicated crosspoint buffers and round-robin input and output arbitrations.

arbitration scheme considers only round-robin with no priorities to avoid starving cells of low priority.

This switch has dedicated XPBs, this is, $VOQ(i, j, p)$ can access only the XPB at input row i that is connected to output j in the buffered crossbar, and each XPB can be accessed by P VOQs with the same i, j pair.

3.3 Shared-Memory Crosspoint Buffered Switch (SMCB)

3.3.1 Model Description

The SMCB switch architecture introduced in Chapter 2 is modified to support different classes of traffic. In an SMCB switch, the crosspoint buffer is shared by m inputs, where $2 \leq m \leq N$ [29] as discussed before. The SMCB switch uses input access schedulers to arbitrate the access from the inputs to the XPBs. The input access scheduler matches the sharing inputs and the shared crosspoints to eliminates the speedup of the shared memory.

In this chapter, P classes of traffic are considered, where Class 0 is the class with the highest priority and Class $P-1$ the one with lowest priority. Here, high priority traffic can be considered as delay sensitive traffic that needs to be served prior to the lower priority traffic.

The proposed switch has $N \times P$ VOQs at each input. A crosspoint in the shared-memory buffered crossbar that connects input port i to output j is also denoted as $XP(i,j)$ as in the CICB switch. It is considered that a crosspoint buffer is shared by two inputs (i.e., $m = 2$) in the mSMCB switch to simplify our explanation and because the switch delivers the highest throughput with this value of m [29] (however, the number of inputs sharing the buffer can be set from 2 to N). The buffer for $XP(i,j)$ and $XP(i',j)$, where $0 \leq i, i' \leq N-1$ and $i \neq i'$, that stores cells for output port j and is shared by these two crosspoints (or inputs i and i') is denoted as before or $SMB(q,j)$, where $0 \leq q \leq \frac{N}{2} - 1$. An even N is considered for the sake of clarity. However an odd N can also be considered. Therefore, in a switch where each crosspoint buffer is shared by two inputs, there are $\frac{N^2}{2}$ SMBs in the buffered crossbar.

Figure 3.2 shows the architecture of the SMCB switch for differentiated services with two inputs sharing the buffered crosspoints. To eliminate the speedup at SMBs, only one input is allowed to access an SMB at a time.

An input access scheduler performs a matching process among the N SMBs and the $m = 2$ non-empty inputs that share them. The matching scheme used in this switch is round-robin based [33] with strict priority.

At each output j in the buffered crossbar, there is an output arbiter to select the outgoing cell from non-empty SMBs. The output arbitration scheme is round-robin without considering priorities to avoid SMB blocking. This blocking occurs when a low priority cell in an SMB obstructs the forwarding of a higher priority cell in the input as the output arbiter might neglect service to low priority cells if other SMBs have higher priority cells. Since an SMB holds cells from two different inputs, then an output arbiter considers up to

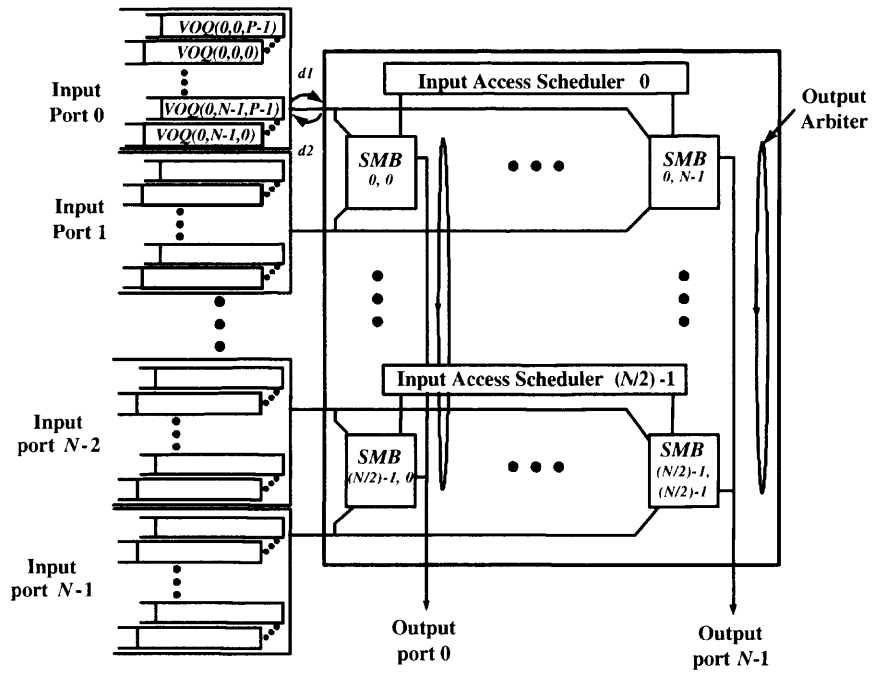

Figure 3.2 $N \times N$ buffered crossbar with shared crosspoints.

two cells from each SMB, where each cell belongs to one of the inputs. Figure 3.2 shows the output arbiters.

The way the 2SMCB switch works is as follows. Cells with priority p, where $0 \leq p \leq P-1$, destined to output j arrive at $VOQ(i,j,p)$ and wait for dispatching. Input i notifies the input access scheduler about the new cell arrival. The input access scheduler selects the next cells to be forwarded to the crossbar by performing a parallel match between the inputs and the SMBs. The selected cell going from input i to output j enters the buffered crossbar and is stored in $SMB(q,j)$. Cells leave output j after being selected by the output arbiters. To reduce the complexity at the buffered crossbar, the output arbiters use round-robin selection without considering the cell priority.

The input access scheduler considers eligible VOQs to those that are not empty and whose corresponding SMBs are not full. After the match, the selection performed by input access scheduler is sent from SMBs to the corresponding VOQs in the next time slot. Therefore, the flow control mechanism between the inputs and the buffered crossbar is

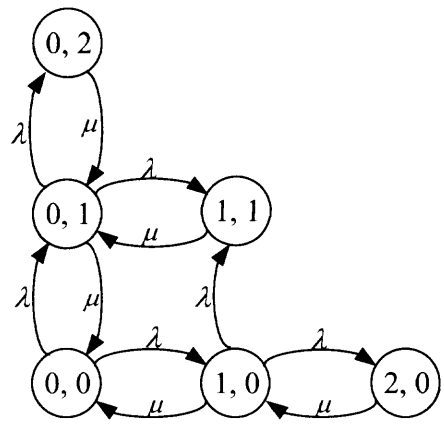

Figure 3.3 State Diagram of a shared memory crosspoint buffer of an $N \times N$ SMCB switch that supports two classes of traffic and $k_s = 2$.

stop-and-go and no arbiter is needed at the line cards. Both cells and flow-control data experience transmission delay between input ports and the buffered crossbar.

3.3.2 Crosspoint-Buffer Blocking Probability with Differentiated Traffic

With the consideration of traffic classes, the blocking probability produced by the crospoint buffer becomes of interest. When a shared crosspoint buffer becomes full, the inputs are inhibited from sending traffic to it. An analog $N \times N$ switch with shared-crosspoint buffers of size two and prioritized random selections is considered that supports two traffic classes. A P-dimensional Markov chain is used to calculate the blocking probability of a shared-crosspoint buffer and independent and identical distributed traffic. Figure 3.3 shows the state diagram of a shared crosspoint buffer with $k_s = 2$. The same arrival (λ) and service (μ) rates are assumed for two classes, high and low. In this figure, state (h, l) represents the number of cells for high and low priority traffic classes, respectively. The incoming cells are blocked when the shared buffer is at states (0, 2), (1, 1), and (2, 0). By following this, it is easy to see that the average blocking probability, P_B, for the combined high and low priorities at the crosspoint buffer, and without considering the effect of the matching process, is $P_B = \frac{\lambda}{\lambda+\mu}$.

3.4 Performance Evaluation of the SMCB Switch under Differentiated Traffic

In this section, the switching performance of the SMCB switch and the CICB switch with dedicated crosspoint buffers (labeled as CICB in the remainder of this chapter) is presented. The throughput of a SMCB switch with different number of inputs and with a single-priority traffic under long RTTs flows with high data rates is studied in [29]. This chapter focuses on handling multi-priority traffic. The switching performance of 32×32 SMCB and CICB switches under traffic with Bernoulli and bursty arrivals with uniform distribution and P classes is compared, where $P = 3$, with Class 0 being the highest priority and Class 2 being the lowest priority. The average cell delay for each priority under uniform traffic and bursty traffic with different burst lengths is studied. Here, a burst is modeled as a two-state Markov modulated process, with an average length l for the active state. The throughput performance under nouniform prioritized traffic, using the unbalanced and diagonal traffic models are obtained.

3.4.1 Uniform Traffic

The average cell delay of the SMCB switch under uniform traffic with average burst lengths of 1 and 100 cells is shown in Figures 3.4 and 3.5 respectively. The RTT under uniform traffic is 1. The minimum average delay of one time slot of the SMCB switch from loads of 0.1 to 0.8 is because the VOQs notify the input access scheduler when a new cell arrives, and this notification takes one time slot, and before forwarding the cell to the buffered crossbar. Note that the magnitude of one time slot is small to be considered significant for high loads. For loads over 0.8, the average cell delay of the highest priority traffic, Class 0, is significantly lower than those of lower priorities, Classes 1 and 2. The average cell delay of SMCB switch under bursty traffic with an average burst length $l = 100$ shows similar magnitude to that of the CICB switch when the load is less than 0.85. When the input load is heavier, the service for the lowest priority class degrades because higher priority classes use most of the service. In this case, Class 2 traffic can no longer achieve 100% throughput

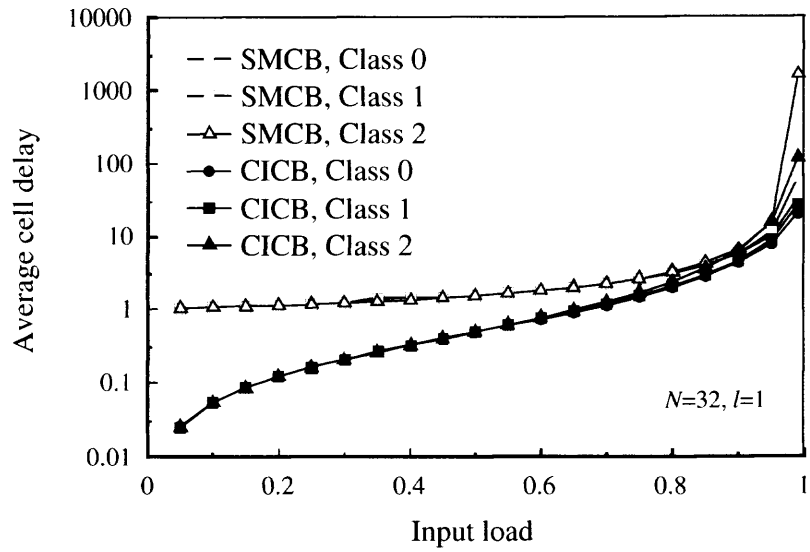

Figure 3.4 Average queuing delay of 32×32 SMCB and CICB switches when $l = 1$, $k_s = k = 1$, and $RTT = 1$.

in neither the SMCB nor the CICB switch. However, the amount of memory used in the SMCB switch is $\frac{N^2}{2}$, which is $\frac{1}{2}$ of that in the CICB switch without speedup.

3.4.2 Unbalanced Traffic

In this section, the throughput performance between SMCB and CICB switch with different crosspoint buffer sizes and RTTs under unbalanced traffic is compared.

Figure 3.6 shows that for the CICB switch, when $k = RTT = 1$, Class 0 and Class 1 traffic achieve 100% throughput under unbalanced traffic. However; the throughput for Class 2 traffic is low. When $k_s = 1$, the total amount of memory in a SMCB switch is $\frac{N^2}{2}$ cells. When $k = 1$, the total amount of memory in the CICB switch is N^2 cells. The SMCB switch achieves similar or better throughput performance than that of the CICB switch with only half the amount of memory in the CICB switch under unbalanced traffic when RTT is not an issue.

When RTT increases from 1 to 3 time slots (or cells), a crosspoint buffer size of 1 is not enough to ensure 100% throughput for Classes 1 and 2 by the CICB switch. Figure 3.7

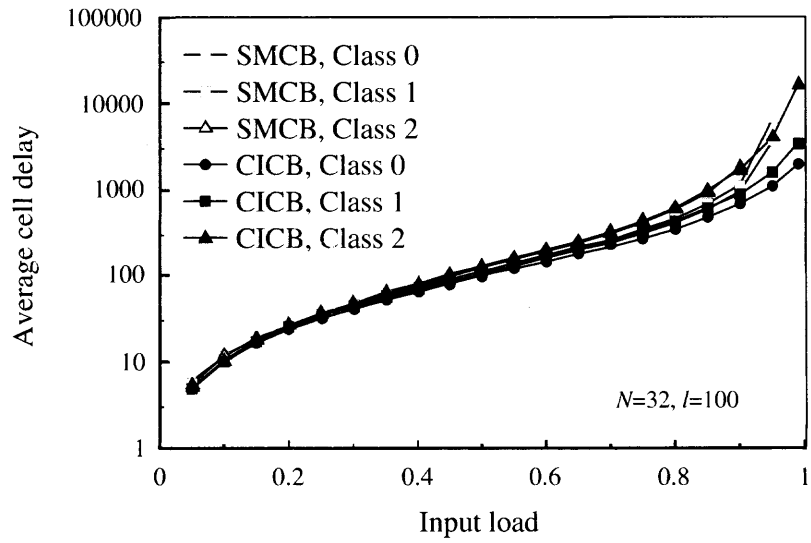

Figure 3.5 Average queuing delay of 32×32 SMCB and CICB switches when $l = 100$, $k_s = k = 1$, and $RTT = 1$.

shows that the throughput for traffic Classes 1 and 2 degrades drastically as the unbalance probability increases, which indicates an increase in the flow rate. For the SMCB switch, the traffic for Class 0 and 1 achieve 100% throughput. However, the throughput of Class 2 traffic degrades as the unbalance probability increases. These results further show the importance of addressing the effect of long RTTs.

Figure 3.8 shows the throughput of the SMCB and CICB switches when the SMCB switch has half and the same amount of memory as that of the CICB switch in the crossbars, and $RTT = 3$. By increasing the crosspoint buffer size by one cell ($k_s = k = 2$), the CICB and SMCB switch achieve higher throughput performance for Class-1 traffic than when $k_s = k = 1$. In this figure, when the SMCB switch has the same amount of memory ($k_s = 4$ and $k = 2$) as that of the CICB switch, all traffic classes have a guaranteed throughput of 100% for $w = 1.0$. Class-2 traffic shows similar throughput performance as in Figure 3.6 for $0 < w < 1.0$.

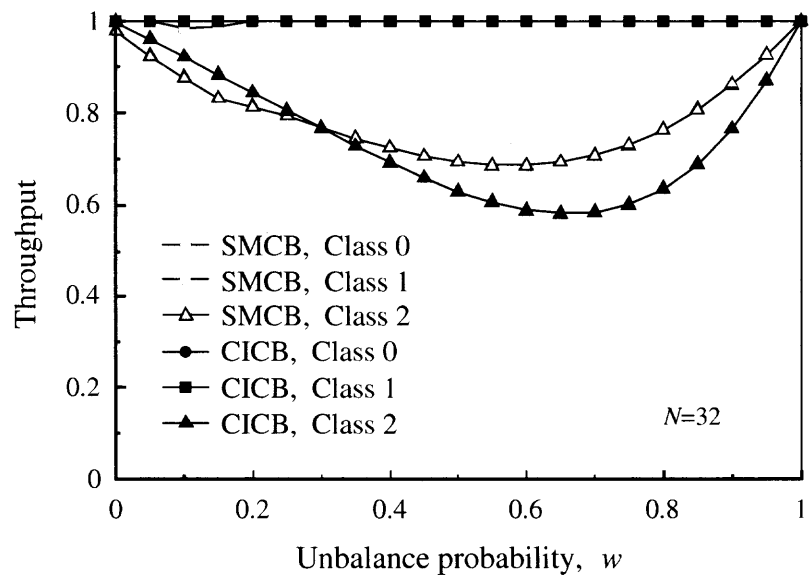

Figure 3.6 Throughput of 32 × 32 SMCB and CICB switches when k_s=k=RTT=1.

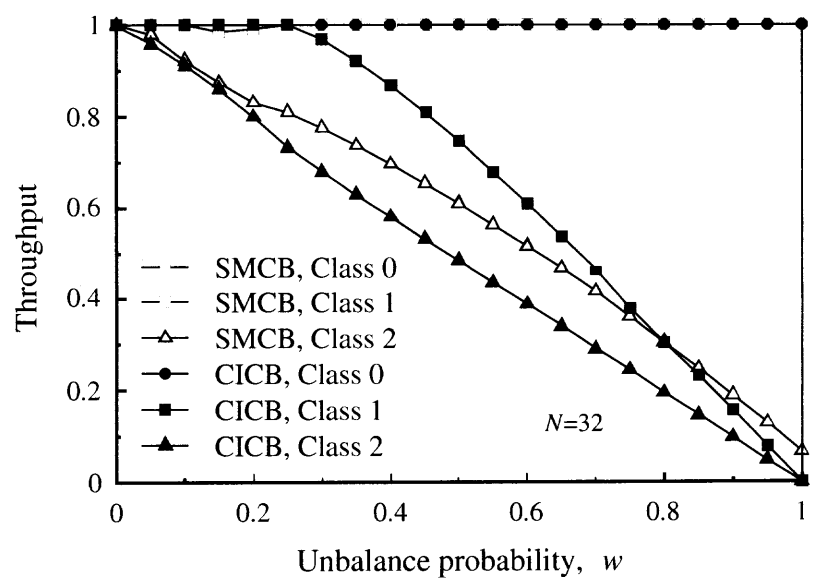

Figure 3.7 Throughput of 32 × 32 SMCB and CICB switches when $k_s = k = 1$, and RTT=3.

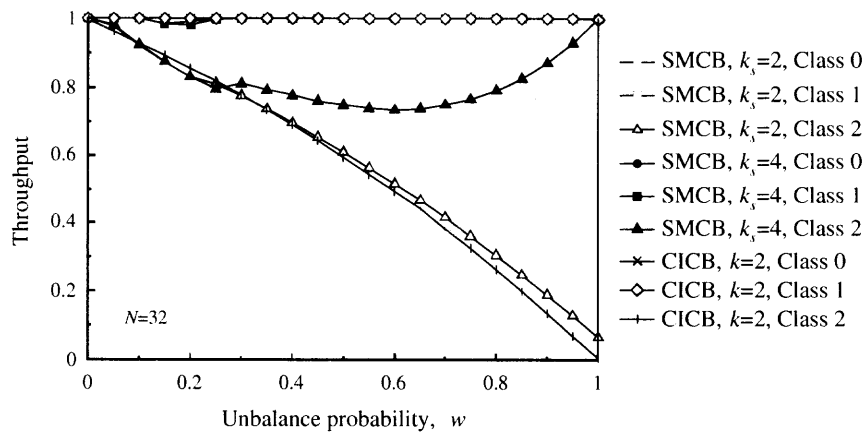

Figure 3.8 Throughput of 32×32 SMCB and CICB switches when SMCB switch has half ($k_s = k = 2$), the same amount of memory as CICB switch ($k_s = \frac{1}{2}$ and $k = 4$), and $RTT = 3$.

3.4.3 Diagonal Traffic

In this section, the throughput performance of the SMCB and CICB switches with different crosspoint buffer sizes and RTTs under diagonal traffic with maximum allowable input load, $\rho_i = 1$ is compared. Since the results under diagonal traffic are symmetric at $d = 0.5$, the following graphs show throughput results for $0 \leq d \leq 0.5$.

Figure 3.9 shows the throughput of the SMCB and CICB switches when the SMCB switch has half the amount of memory of that in the CICB switch. Class-0 and Class-1 traffic achieve 100% throughput each. The throughput of Class-2 traffic for both switches degrade when $d \neq 0, 1$. The throughput of Class-2 traffic of CICB switch is a slightly higher than that of the SMCB switch. Considering the fact that the SMCB switch has half the amount of memory as that of the CICB switch, the throughput advantage is not significant.

Figure 3.10 shows the throughput of the SMCB and CICB switches when both have the same amount of memory, $k_s = 4, k = 2$, and $RTT = 3$. Both switches achieve 100% throughput for Class-0 and Class-1 traffic. Here, the crosspoint buffer size is set to one cell smaller than RTT to show the throughput difference of Class-2 traffic in both

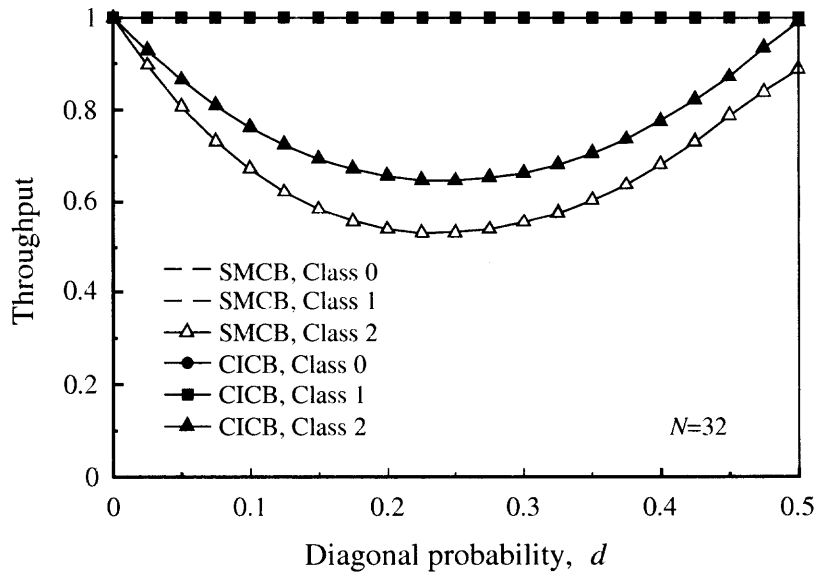

Figure 3.9 Throughput of 32×32 switches when the SMCB switch has half the amount of memory of that in the CICB switch ($k_s = k = 1$), and $RTT = 1$.

switches. With the same amount of memory, the SMCB switch shows higher throughput performance than the CICB switch for Class 2 traffic when $0 \leq d \leq 0.25$.

3.5 Conclusions

This chapter presented the support for differentiated services by a buffered crossbar switch with dedicated buffers, or CICB switch, and for different RTTs. In a CICB switch, it is required that the crosspoint buffer size k be at least RTT cells long to achieve high throughput for all priority traffic classes, and specially for flows with high data rates. To minimize the crosspoint-buffer size, SMCB switch was studied, such that RTT can be twice as long as that supported by a CICB switch with dedicated buffers, without decreasing switching performance. In this way, the SMCB switch provides 100% throughput for high data-rate flows under long round-trip times and when traffic has uniform distributions. This switch relaxes the amount of memory to $\frac{1}{2}$ of the amount required by a CICB switch with dedicated buffers to support P classes of traffic and flows with high data rates. In general,

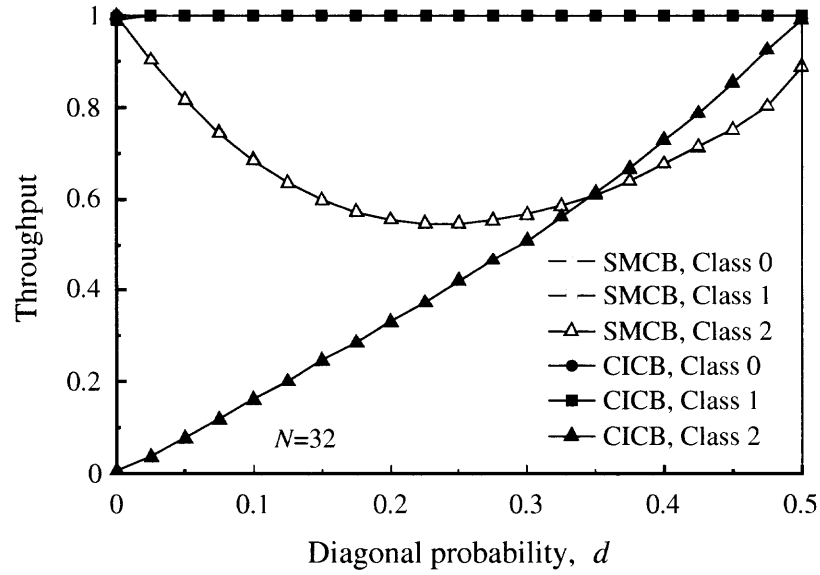

Figure 3.10 Throughput of 32×32 SMCB and CICB switches when they have the same amount of memory ($k_s = 4$ and $k = 2$) and $RTT = 3$.

this memory requirement relaxation is $\frac{1}{m}$, where m can be up to N, when data flows are expected with high data rates (e.g., rates close to the port capacity), with however; a decrease of switching performance for traffic flows with smaller data rates and nonuniform distributions.

In addition, it is shown that the shared memory used in the crosspoint buffers needs no speedup to achieve high throughput. The timing relaxation that a CICB switch has for cell selection is partially lost in the SMCB switch as the input access schedulers perform parallel matching. However, the arbiter in the input access scheduler are placed in the same chip, the buffered crossbar, such that the matching time is rather small. The advantage of the SMCB switch is that memory is more efficiently handled than in a CICB switch such that higher switching performance is achieved without recurring to speedup and large amounts of memory. This trade-off well serves the cases when the distance between the line cards and the buffered crossbar is long. As a future work, the performance of the

SMCB switch can be studied with selection schemes more sensitive to the traffic load [6] as the ones studied here are based on simple round-robin selections.

CHAPTER 4

PACKET SWITCHING AND REPLICATION OF MULTICAST TRAFFIC BY CROSSPOINT BUFFERED PACKET SWITCHES

4.1 Introduction

The migration of broadcasting services, such as cable TV and multimedia-on-demand to packet-oriented networks and the embracing of emerging applications, such as teleconferencing and storage networks, by the Internet are expected to take place in the near future. These dominant applications have the potential of loading up the next generation Internet. To keep up with such bandwidth demand, packet switches and routers need to provide efficient multicast switching and packet replication.

Recently, a lot of research has focused on unicast traffic, where each packet has a single destination. It has been shown that unicast switches achieve 100% throughput under admissible conditions, $\sum_i \lambda_{i,j} < 1$ and $\sum_j \lambda_{i,j} < 1$, where i is the index of inputs ($0 \leq i \leq N-1$), j is the index of outputs ($0 \leq i \leq N-1$) for an $N \times N$ port switch, and $\lambda_{i,j}$ is the average arrival rate from input i to output j while using a variety of switch architectures.

Although it is difficult to describe actual multicast traffic models, in general, it is considered that switches also need to provide 100% throughput under admissible multicast traffic. In multicast switches, the admissibility conditions are similar to those for unicast traffic, however, with the consideration of the fan out of multicast packets. This implies that the fan out of a multicast packet increases the output load.

Multicast packet switching and packet replication has been broadly studied on input buffered (IB) packet switches [42, 43]. IB switches have the limitation of being able to forward (multicast) packets from input to outputs such that: a) up to one multicast packet can depart from each input and b) up to one (unicast or replicated) packet per output. IB

switches with a single first-in first-out (FIFO) buffers in each input suffer from head-of-line (HOL) blocking under unicast traffic [12]. HOL blocking is a phenomenon that occurs when a packet behind the HOL packet cannot be forwarded to its destined output despite the output being idle because the HOL packet has lost contention to a packet from another input. To remove HOL blocking under unicast traffic, $N \times N$ switches use N queues where each one stores packets destined to each different output. This queuing structure is called virtual output queueing (VOQ).

In IB switches, the replication of multicast packets can be handled in two main different ways: 1) at the input unicast buffers (in switches with VOQs), and 2) by performing the replication at the switch fabric.

To resolve which multicast packets are transmitted per output (and therefore, per input) in IB switches, matching between input and output ports is performed. In the matching process, the selection of input-output pairs is based on assigned weights or heuristically. Switches that replicate multicast packets at the inputs require large input buffers for heavy traffic loads with large fan outs. Matching schemes for IB switches with VOQs have shown to provide 100% throughput under unicast admissible traffic using a maximum weight matching (MWM) scheme with, however, prohibitively high processing complexity that may suffer from large response delay under high data rates or large capacity switches. The matching of multicast packets is more complex than that of unicast packets as the contention degree is higher. Alternatively, 100% throughput under unicast traffic can be achieved with lower complexity schemes with, however, speedup (memory and switch fabric running faster than the line speed).

In switches with cell replication at the switch fabric, multicast cells can be stored in a single queue at the input, separated from unicast cells. Here, cell replication is performed by using the replication capabilities of the switch fabric [42]. Although with reduced memory amount at the input ports, this approach might suffer from severe HOL blocking as a number of ports need to be available for receiving a cell at a time slot.

Following the VOQ strategy for unicast traffic, virtual multicast queues (VMQs) can be used to avoid HOL blocking, where there is a queue for each different fan out combination, or up to $2^N - 1$ VMQs in an input. This large number of VMQs makes it an impractical approach, even for small N. As a practical alternative, it has been of interest to find the smallest number of multicast queues k (where each queue can store multicast cells with a set of fan out combinations), where $2 \leq k \leq 2^N - 1$, and optimal queuing policies to reduce HOL blocking [43]. This approach, although it reduces the number of queues, has the drawback of being difficult to find a suitable value of k for different traffic patterns. It also depends on the queuing policies and on the multicast traffic model used.

The switching performance is also a function of the call splitting policy used. No call splitting, or one-shot policy, is the forwarding of multicast cells to their destination ports in a single time slot. If one of the multicast copies cannot be forwarded because of output contention, then the cell remains at the buffer and no cell replication is performed. No call splitting requires VMQs to avoid HOL blocking.

On the other hand, call splitting allows forwarding of some or all of the multicast copies to different output ports in one time slot. Therefore, call splitting mitigates HOL blocking (although it doesn't completely remove it), and at the same time, it relaxes the requirement of VMQs. Call splitting is then widely considered in multicast switches.

In addition to the high performance of combined input-crosspoint buffered (CICB) packet switches under unicast traffic, the crosspoint buffers in these switches help to provide call splitting intrinsically. Different from IB switches, CICB switches are not required to have cell transmission with matched inputs and outputs [21]-[1]. In CICB switches, one input can send up to one (multicast) cell to the crossbar, and one or more cells destined to a single output port can be forwarded from multiple inputs to the crossbar at the same time slot [44]- [45]. Therefore, CICB switches have natural properties favorable for multicast switching as contending copies for a single output can be sent to the crosspoint buffers from several inputs at the same time slot without blocking each other.

In general, CICB switches have dedicated crosspoint buffers for each input-output pair that is available for forwarding cells between those two ports, independent of the actual memory utilization. Since memory used in the crosspoint buffers has to be fast, it is desirable to minimize the amount of it. To decrease the number of needed crosspoint buffers, N^2, in a CICB switch, that has been previously proposed to use of crosspoint buffers that are shared by several inputs. The switch is called shared-memory crosspoint buffer (SMCB) switch [46], [47]. The SMCB switch has shown to provide 100% throughput under unicast uniform traffic, and high performance under traffic with non uniform distributions. Therefore, the SMCB switch showed that with memory sharing the switch can achieve high performance while decreasing the required amount of fast memory.

In this chapter, the following question is addressed: what is the maximum throughput that a CICB switch can achieve under multicast traffic? Furthermore, the throughput of the SMCB switch under multicast traffic is studied through the adoption of a simple scheduling scheme for inputs to access the shared buffers and a set of output arbitration schemes tailored for multicast traffic. The proposed schemes cover a variety of simple weightless and weighted arbitration schemes. With the use of a very simple input-access scheduling scheme, it is shown that the SMCB switch is a feasible and an efficient architecture for the implementation of CICB packet switches that provide unicast and multicast services.

4.2 Combined Input-Crosspoint Buffer Switch

In an $N \times N$ CICB switch, the buffered crossbar has N inputs and N outputs. There is one multicast FIFO at each input. A multicast cell uses a bitmap of size N to represent the destinations of multicast-cell copies. If the bitmap has a value of '1' at the jth position, output j is one of the destinations of the multicast cell. A crosspoint (CP) element in the buffered crossbar that connects input port i to output port j is denoted as $CP(i,j)$. The crosspoint buffer of $CP(i,j)$ is denoted as $CPB(i,j)$. The size of $CPB(i,j)$ is k cells, where $k \geq 1$. Figure 4.1 shows a CICB switch with dedicated crosspoint buffers.

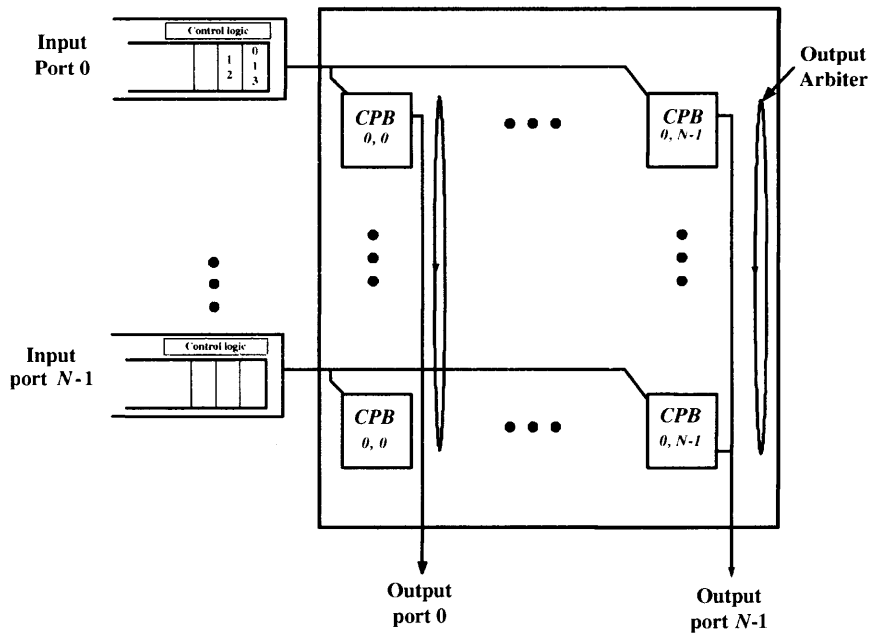

Figure 4.1 Multicast CICB switch with dedicated CPBs.

To avoid cell loss within the switch, a credit-based flow control mechanism is used. This mechanism indicates input i whether $CPB(i, j)$ has room available for a cell or not. To implement this mechanism, each input has a credit counter for each $CPB(i, j)$ where the maximum count is the number of cells that a CPB can hold. When the number of cells sent by the FIFO to $CPB(i, j)$ reaches the maximum count k, the FIFO is inhibited of sending cells to that CPB to avoid buffer overflow. The count for $CPB(i, j)$ is increased by one each time a cell is sent by input i to this CPB and decreased by one each time that $CPB(i, j)$ forwards a cell to output j. A CPB that has available room for at least one cell is considered eligible to receive a cell.

This switch uses call splitting to forward a multicast cell to the buffered crossbar. This is, as long as a CPB is eligible to receive a multicast cell and there is a multicast cell for this SMB from the HOL multicast cells at the input FIFO, the multicast cell is sent to the buffered crossbar. The crossbar performs multicast cell replication. Each time the HOL multicast cell is dispatched to destination j, the jth bit of the multicast bitmap is reset. When the bitmap of the HOL cell is zero, the multicast cell is considered served.

4.3 Shared-Memory Crosspoint Buffered (SMCB) Switch

The SMCB switch also has one multicast FIFO at each input, N^2 crosspoints and $\frac{N^2}{m}$ crosspoint buffers in the buffered crossbar. Each crosspoint buffer is shared by m inputs. Here the crosspoint buffers are shared by two inputs in the remainder of this chapter. In this switch model, each shared crosspoint buffer is denoted as SMB to differentiate from the notation in the CICB switch. The flow control mechanism used in the SMCB is similar as that used in the CICB switch.

A crosspoint in the buffered crossbar that connects input port i to output j is also denoted as $CP(i, j)$. The buffer for $CP(i, j)$ and $CP(i', j)$, where $0 \leq i' \leq N - 1$ and $i \neq i'$, that stores cells for output port j and is shared by these two crosspoints (or inputs i and i') is denoted as $SMB(q, j)$, where $0 \leq q \leq \frac{N}{2} - 1$. An even N is assumed for the sake of clarity. However, an odd N can be used with one input port using dedicated buffers of size 0.5 to 1.0 the capacity of an SMB.

To eliminate the speedup at SMBs, only one input is allowed to access an SMB at a time. To schedule the access to the N SMBs from the two inputs, an input-access scheduler is used. Figure 4.2 shows the architecture of the mSMCB switch when $m = 2$. The size of an SMB, in the number of cells that can be stored, is k_s. There are $\frac{N}{m}$ input-access schedulers in the buffered crossbar. An input-access scheduler selects which non-empty inputs access the SMBs that have room for storing a new cell.

Multicast cells at the inputs are handled in similar way as in the CICB switch, also using a multicast bitmap to indicate the destination of the multicast cells.

4.4 Arbitration Schemes for the CICB Switch

Here, as it is considered that only multicast traffic traverses the switch (unicast can be thought of as a subset of the multicast service) and that a single FIFO at the inputs is used, an input dispatches a copy of the multicast cell to the buffered crossbar as long as a destination CPB is available or not full.

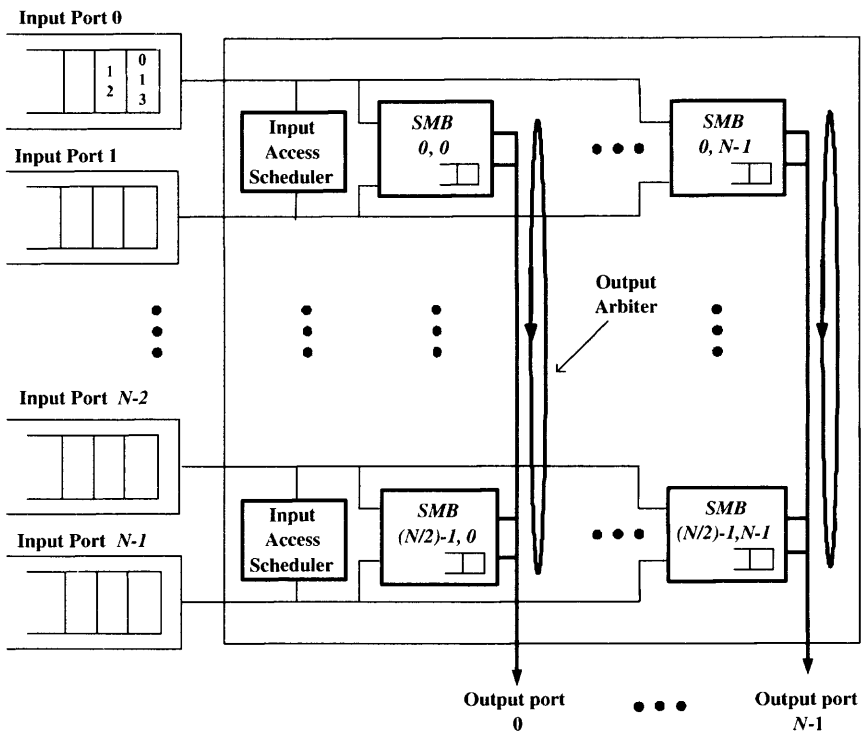

Figure 4.2 $N \times N$ mSMCB switch, $m = 2$.

Output Arbitration Schemes

- Round-robin (RR): This arbitration scheme considers a weightless selection from all occupied CPBs and a pointer to indicate the most preferred input of the priority list per output at each time slot. The output arbiter selects the CPB that is next to the pointer in a round-robin fashion. After a CPB is selected, the pointer moves one position beyond the selected one.

- Longest row first (LRO): In this scheme, an output arbiter selects a non-empty CPB with the largest occupancy in the (row of) CPBs from input i. Ties are broken arbitrarily. This scheme has the advantage of using state information from the crossbar to make a weight-based selection, and therefore, no transmission overhead is added in the information sent from inputs to the buffered crossbar.

- Largest multicast cell occupancy first (LMO): In this scheme, the output arbiter selects a CPB from the input that has the largest number of multicast cells in the FIFO.

This information is sent from each input to the crossbar, therefore, this scheme adds transmission overhead. However, a discussion of methods to decrease this overhead is out of the scope of this dissertation and it is left for future research.

- Largest number of cell copies first (LCO): In this scheme, the output arbiter selects a CPB whose input has the largest number of multicast copies, this is, the sum of the fan out of each multicast cell.

4.5 Arbitration Schemes for the SMCB Switch

In the SMCB switch, the two inputs sharing the SMBs contend for access to each SMB. Under unicast traffic, the input access scheduler performs a match between inputs and SMBs to resolve this. Under multicast traffic, the matching process can be more complex to resolve than under unicast traffic. The following are the selection schemes used to arbitrate input access.

4.5.1 Input Arbitration Scheme

Time alternating access (TAA)

Here, each input is granted access to the SMBs in a predefined alternating time slots. For example, for the two sharing inputs i and i', where $i \neq i'$, even time slots let i have prioritized access to SMBs and odd time slots let i' have prioritized access. In this way, different inputs are allowed to complete the multicast service fairly. This scheme also has the advantage of providing low complexity (i.e., $O(1)$) for the implementation of the input-access scheduler and for observing the natural performance of SMCB switches without elaborated schemes.

4.5.2 Output Arbitration Schemes

All the output arbitration schemes for the CICB switch are also considered for the SMCB, except for the LRO scheme, which is modified to use SMBs instead of CPBs. In this case, LRO considers the occupancy of all SMBs among the cells belonging to input i.

The combination of input and output arbitration schemes are denoted as input arbitration scheme (IAS)-output arbitration scheme (OAS), or IAS-OAS, in the remainder of this chapter.

4.6 Performance Evaluation

Models of the CICB and SMCB switches are implemented in discrete-event simulation programs for computer simulation. The arbitration schemes considered are RR, LRO, LCO, and LMO as output arbitration schemes for the CICB switch, and TAA-RR, TAA-LRO, TAA-LMO, and TAA-LCO as IAS-OAS schemes for the SMCB switch. Switches with sizes of $N = \{8, 16, 32\}$ are considered. Simulation results are obtained with simulation time of 300,000 time slots.

The traffic model used in this study is uniform multicast traffic with Bernoulli arrivals. Each multicast destination fan out, from 1 to N outputs, is represented by multicast vector that indicates the destinations of the multicast copies. For an $N \times N$ switch, the multicast vectors are uniformly distributed over all possible combinations $(2^N - 1)$ [44]. Each multicast fan out is created with the same probability. The average fan out for an $N \times N$ switch is $\frac{N+1}{2}$. Therefore, the maximum admissible input load for such switch size is $\frac{2}{N+1}$. The variable of interest is the maximum achievable throughput under admissible traffic.

Figure 4.3 shows the maximum throughput of 8×8 CICB and SMCB switches with different crosspoint-buffer sizes. The simulation shows that the CICB switch has intrinsic capabilities to perform call splitting, and therefore, it provides high throughput under this traffic pattern, except for the case of the LRO scheme. One of the reasons for the low

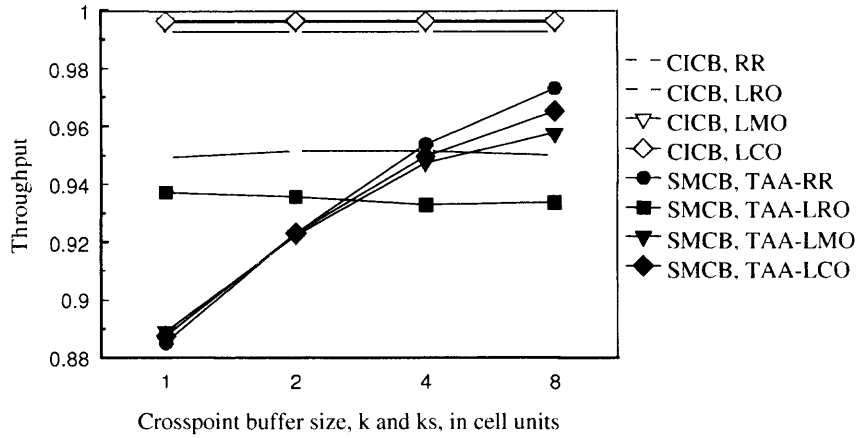

Figure 4.3 Throughput of 8×8 switches using different crospoint-buffer sizes.

throughput of LRO is that the uniformity of the traffic and the small size of the crosspoint buffers, k, makes the selection of the CPBs rather arbitrary, and the selection scheme turns out to be inefficient.

In case of the SMCB switch, this delivers low throughput under small sizes of the SMB, k_s. However, as k_s increases the throughput also increases. Here, the maximum throughput for this small SMCB switch is about 98% when using TAA-RR. This throughput, although lower than that obtained by the CICB switch, is achieved using half the memory amount in the buffered crossbar of that used by the CICB switch with similar performance. In addition, the results show that the SMCB switch using LRO has similar performance to its CICB counterpart. Therefore, the effect of this selection scheme in the SMCB switch produces similar results.

Figure 4.4 shows the throughput performance of 16×16 switches. These results show again the high performance of the CICB switches, except for the one using LRO. Here, however, the measurable differences among the switches is indistinguishable as the switch size increases (see curves for CICB in Figure 4.3). The throughputs obtained by the CICB switches are similar to those obtained in the 8×8 switch. However, for the case of the SMCB switches, the throughputs increase for all arbitration schemes, although the throughputs follow similar trends as in those of the smaller switches for the increasing

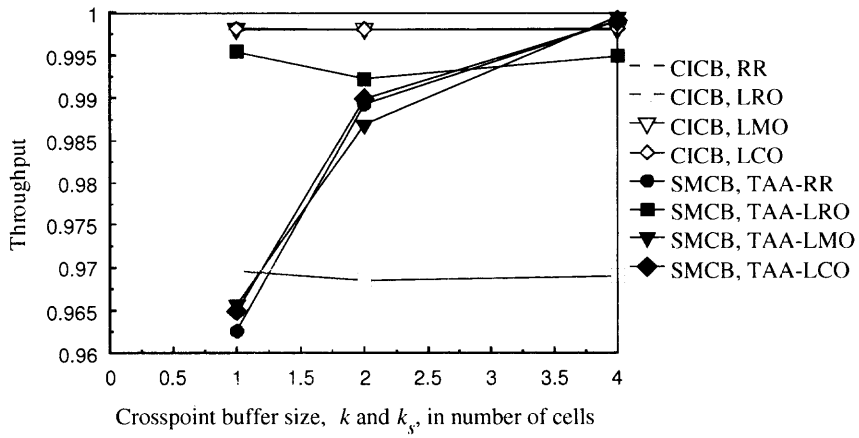

Figure 4.4 Throughput of 16×16 switches using different crospoint-buffer sizes.

values of k_s. Here, the minimum throughput is observed with the smallest k_s (i.e., 1 cell) with a value above 96%. The throughput increases as k_s increases to values beyond 99% with $k_s \geq 2$, which is the memory amount in the buffered crossbar equivalent to having $k = 1$ in the CICB switch. When $k_s = 4$, the throughput delivered by the SMCB switch is higher than that of the CICB switches, except for the cases of the LRO arbitration. As for the SMCB switch with LRO arbitration, the throughput trend is similar to the smaller switch, however, the throughput is higher than that delivered by its CICB counterpart. These results show that the SMCB switch for a 16×16 size has equivalent or better performance than the CICB switch with $k_s = 4$.

As the CICB switches in the previous two sizes show a rather stable throughput, the study is focused on 32×32 SMCB switches to follow up performance trends and scalability. Figure 4.5 shows the throughput of a 32×32 SMCB switch with the different IAS-OAS schemes considered. The throughput of this switch size sharply increases with the increase of k_s. The lowest throughputs are obtained with $k_s = 2$ for all arbitration schemes. Here, the lowest throughput is delivered by TAA-RR (weightless arbitration) with a value of 98.5%. The other schemes deliver 99% throughput or higher. The TAA-LRO scheme, which has the lowest throughput for smaller switch sizes, is rather stable at nearly 100% for any k_s. For $k_s \geq 4$ the throughput of all arbitration schemes is nearly 100%. Therefore,

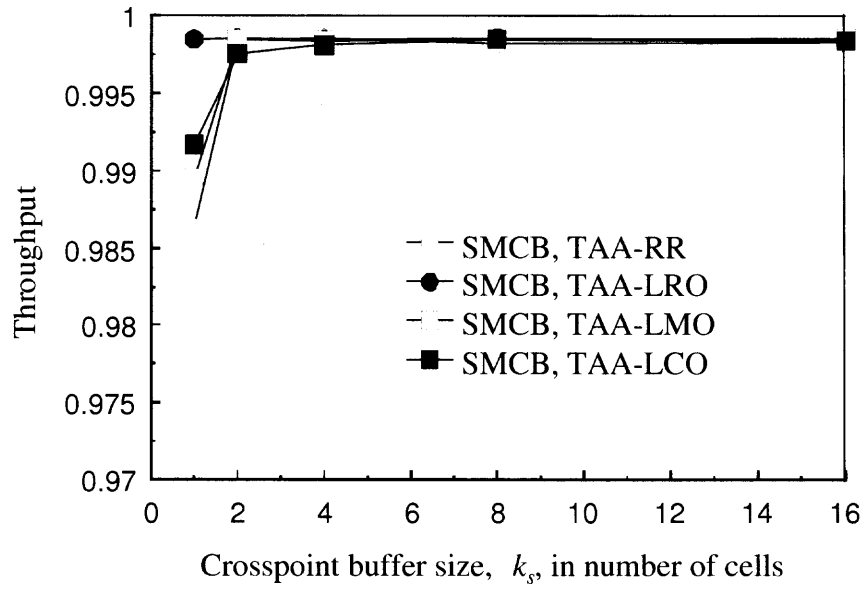

Figure 4.5 Throughput of 32×32 switches using different crospoint-buffer sizes.

it's shown that as the switch size increases the throughput approaches 100%, independently of k_s.

4.7 Output-based Shared-Memory Crosspoint Buffered (O-SMCB) Switch

To improve throughput performance of the SMCB switch under multicast traffic, output-based SMCB (O-SMCB) switch is proposed. To observe the response of the proposed switch under multicast traffic only, the O-SMCB switch is also provisioned with one multicast first-in first-out (FIFO) queue at each input. This switch has N^2 crosspoints and $\frac{N^2}{2}$ crosspoint buffers in the crossbar. Figure 4.6 shows the architecture of the O-SMCB switch. A crosspoint in the buffered crossbar that connects input port i to output j is denoted as $CP(i,j)$. The buffer shared by $CP(i,j)$ and $CP(i,j')$ that stores cells for output ports j or j', where $j \neq j'$, is denoted as $SMB(i,q)$, where $0 \leq q \leq \frac{N}{2} - 1$. An even N is assumed for the sake of clarity. However, an odd N can be used with one input port using dedicated buffers of size 0.5 to 1.0 the size of an SMB. The size of an SMB, in number of cells that can be stored, is k_s. In this chapter, the case of minimum amount of memory, or when

$k_s = 1$ (equivalent to having 50% of the memory in the crossbar of a CICB switch) is studied. Therefore, $SMB(i, q)$ with $k_s = 1$ can store a cell that can be directed to either j or j'. The SMB has two egress lines, one per output. To differentiate the SMCB with buffers shared by inputs with the O-SMCB switch, the SMCB in the above section is denoted as I-SMCB in the rest of this chapter.

To avoid the need for speedup at SMBs, only one output is allowed to access an SMB at a time. The access to one of the N SMBs by each output is decided by an output-access scheduler. A scheduler performs a match between SMBs and the outputs that share them by using round-robin selection. There are $\frac{N}{2}$ output-access schedulers in the buffered crossbar, one for each pair of outputs. Multicast cells at the inputs have an N-bit multicast bitmap to indicate the destination of the multicast cells. Each bit of the bitmap is denoted as D_j, where $D_j = 1$ if output j is one of the cell destinations, otherwise $D_j = 0$. Each time a multicast copy is forwarded to the SMB for the cell's destination, the corresponding bit in the bitmap is reset. When all bits of a multicast bitmap are zero, the multicast cell is considered completely served. Call splitting is used by this switch to allow effective replication and to alleviate a possible head-of-line blocking.

A flow control mechanism is used to notify the inputs about which output replicates a multicast copy and to avoid buffer overflow. The flow control allows the inputs to send a copy of the multicast cell to the crossbar if there is at least one outstanding copy and an available SMB for the destined output. After all copies of the head-of-line multicast cell have been replicated, the input considers that cell served and starts the process with the cell behind.

4.8 O-SMCB Performance Evaluation

The performance of the proposed O-SMCB switch is compared to those of a CICB and I-SMCB switches. Models of 16×16 O-SMCB, I-SMCB, and CICB switches were implemented in discrete-event simulation programs. Similarly to the O-SMCB, the SMBs are

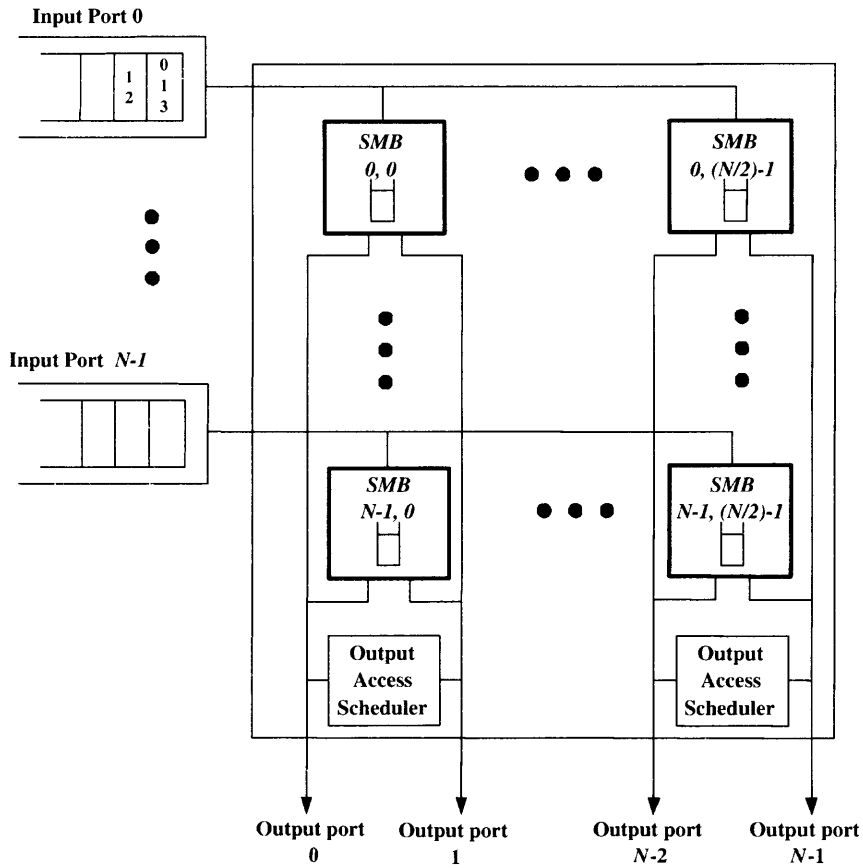

Figure 4.6 $N \times N$ O-SMCB switch with shared-memory crosspoints by outputs.

shared in the I-SMCB switch, however, by (two) inputs. For a fair comparison, the I-SMCB also uses round-robin selections for SMB-access by inputs and for output arbitration. The CICB switch uses round-robin for input and output arbitrations. The maximum achievable throughput for each switch is studied. The switches were simulated for 500,000 time slots.

The simulation considers multicast traffic models with uniform and nonuniform distributions and Bernoulli arrivals: multicast uniform, multicast diagonal with fanouts of 2 and 4 (called diagonal2 and diagonal4, respectively), and broadcast. In the uniform multicast traffic model, multicast cells are generated with a uniformly distributed fanout among N outputs. For this traffic model, the average fanout is $\frac{1+N}{2} = \frac{17}{2}$ and a maximum admissible input load of $\frac{1}{fanout}$=$\frac{1}{8.5}$. This traffic model includes a fanout=1 or unicast traffic. The multicast diagonal2 traffic model has a destination distribution to $j = i$ and $j = (i+1)\%N$ for each multicast cell, and a maximum admissible input load of 0.5. The multicast diagonal4 traffic model has the copies of multicast cells destined to $j = \{i, (i+1)\%N$, $(i+2)\%N$, and $(i+3)\%N\}$ for each multicast cell, and its admissible input load is 0.25 (i.e., output load=1.0). In general, multicast diagonal$m$ traffic has copies of multicast cells destined to $j = (i+\tau)\%N$, where $\tau = 0, 1...m$. Here m is the fanout value for the multicast diagonal traffic. The admissible load is $\frac{1}{m}$. A broadcast multicast cell generates copies for all N different outputs and has a maximum admissible load is $1/16 = 0.0625$.

Under admissible multicast uniform traffic, all switches deliver 100% throughput. These results are observed under both Bernoulli and Bursty arrival. Under admissible multicast diagonal2 traffic, the throughputs observed are 100% for the O-SMCB and CICB switches, and 96% for the I-SMCB switch. Under admissible multicast diagonal4 traffic, the performance of the I-SMCB switch decreases to 67%, while the throughputs of the O-SMCB and CICB switches remain close to 100%. Under broadcast traffic (fanout equal to N), the throughput of the O-SMCB switch is 99% and the throughput of the CICB switch is close to 100%, while the throughput of the I-SMCB switch is 95%. The simulation results under diagonal multicast traffic are shown in Figure 4.7.

Figure 4.8 and Figure 4.9 show the throughput of 16× 16 and 32 × 32 CICB, I-SMCB and O-SMCB switches under diagonal multicast traffic with different fanouts with maximum admissible input load. Here I-SMCB and O-SMCB switches have half the amount of memory as CICB switch. It is shown that the throughput of O-SMCB and CICB switches remains close to 100% with different fanout values. The reason that I-SMCB switch gives worst performance when fanout=2 is because diagonal multicast traffic has copies from the sharing inputs to the same outputs. In another word, the sharing inputs are always competing for the same SMBs. Therefore, the throughput is only 50%. O-SMCB switch achieves high throughput performance with half the amount of memory as CICB switch.

Throughput degradation under overload conditions

Multicast is a traffic type difficult to police for admissibility. Furthermore, the performance of switches under inadmissible traffic (produced by larger fanouts than the expected average) may change. In cases of unicast traffic, the maximum throughput of a switch can remain high with a fair scheduler. However, this might not be the case under multicast traffic. In this experiment, the input load is increased beyond the maximum admissible values in the considered traffic models to observe throughput changes of the O-SMCB, I-SMCB, and CICB switches under these overload conditions. Here, the throughput of the switches is measured as a ratio between the maximum measured throughput and the maximum throughput that a switch is able to provide when all outputs are able to forward a cell.

Under uniform multicast traffic, the throughputs of O-SMCB and I-SMCB switches degrade to 93% when the input load is larger than 0.117 (i.e., output load is larger than 1.0), while the throughput of the CICB switch is 100%. This throughput degradation in the SMCB switches occurs because of the increased number of contentions for SMB access as the traffic load increases. Under multicast diagonal2 traffic, the throughputs of the I-

SMCB and CICB switch drop to 96% and 93%, respectively, while the throughput of the O-SMCB switch remains close to 100%. Under multicast diagonal4 traffic, the throughput of the I-SMCB switch drops to 68%, while the throughputs of the O-SMCB and CICB switches remain close to 100%. Under broadcast traffic, the throughput of the I-SMCB switch decreases to 79%. However, the throughputs of the O-SMCB and CICB switches remain close to 100%.

Table 4.1 Throughput under Multicast Traffic

Traffic type	$Ta(I)$	$Ta(O)$	$Ta(C)$	$Ti(I)$	$Ti(O)$	$Ti(C)$
Uniform	100%	100%	100%	93%	93%	100%
Diagonal2	96%	100%	100%	96%	100%	93%
Diagonal4	67%	100%	100%	68%	100%	100%
Broadcast	95%	99%	100%	79%	100%	100%

Table 4.1 summarizes the obtained throughput for all tested traffic models. In this table, Ta stands for the measured throughput under admissible traffic and Ti for the measured throughput under inadmissible traffic. The letters I, O, and C in parenthesis indicate that a result is related to the I-SMCB, O-SMCB, and CICB switches, respectively. As seen in this table, the performance of the O-SMCB switch is comparable to that of the CICB switch and higher than that of an I-SMCB switch. Therefore, the O-SMCB switch provides comparable performance but with 50% the memory amount of a CICB switch.

4.9 Conclusions

In this chapter, the performance of CICB, I-SMCB, and O-SMCB switch is studied. It is shown that several simple arbitration schemes used in CICB switches can provide high throughput performance when using the minimum crosspoint-buffer size of one cell. In addition, these simple arbitration schemes were used in I-SMCB switch, which uses shared

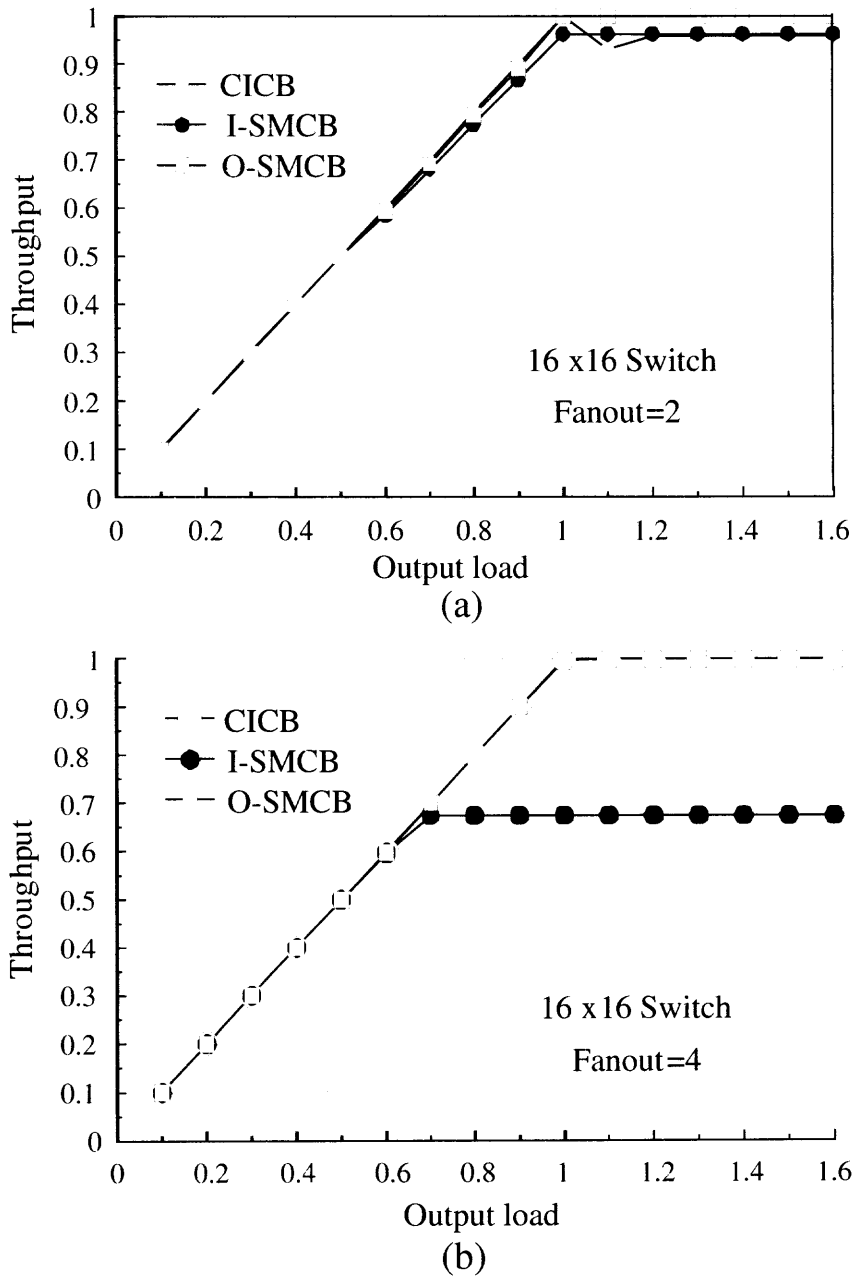

Figure 4.7 Throughput performance of 16 × 16 I-SMCB and O-SMCB switches under a) diagonal2 traffic and b) diagonal4 traffic.

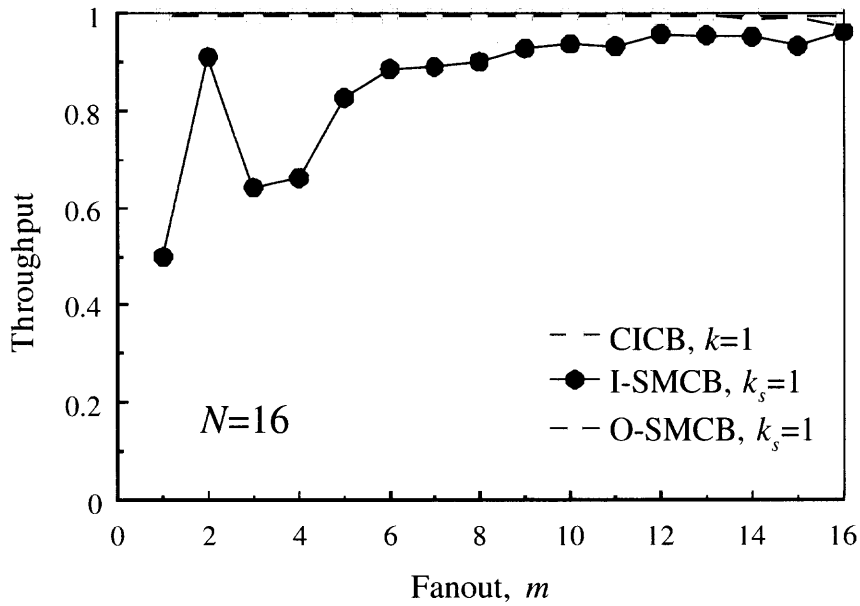

Figure 4.8 Throughput performance of 16 × 16 I-SMCB and O-SMCB switches under diagonal traffic with different Fanout values.

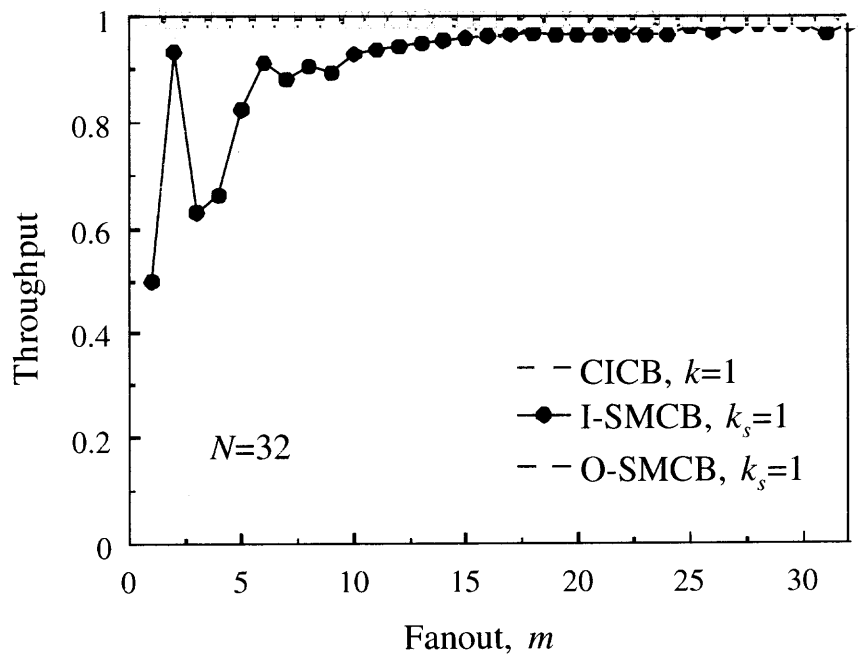

Figure 4.9 Throughput performance of 32 × 32 I-SMCB and O-SMCB switches under diagonal traffic with different Fanout values.

buffers in the crosspoint to reduce the memory amount in the buffered crossbar. The shared memory requires no speedup to provide high performance. It is shown that small switches can provide high performance under uniform multicast traffic. However, the small CICB switches might be more appealing as the memory savings are rather low. As the switch sizes increases, the amount of shared crosspoint buffer needed to provide high switching performance decreases, making the SMCB scalable for multicast traffic.

Furthermore, this chapter studied how the throughput changes in the CICB and SMCB switches under inadmissible (overload) conditions. In general, small SMCB switches are sensitive to overloading conditions as the throughput decreases if the switch load is increased beyond admissibility conditions. These results are expected if the throughput of such switches is not 100% throughput under admissible conditions. As the crosspoint-buffer size increases, the throughput differences decrease in these small switches. As for larger switches, their performance follows similar trends in throughput differences. However, as the crosspoint-buffer size slightly increases, the throughput degradation sharply decreases, making these switches insensitive to overloading.

An exception to the observations above were noted with LRO-based schemes. These schemes showed insensitivity to crosspoint buffer size values and to overloading conditions. In general, the throughput of LRO-based arbitration schemes increase as the switch size increases.

Considering the throughput performance under unicast and multicast traffic with smaller memory amount in the buffered crossbar than in the CICB switch, the SMCB switch is an improved and economical version of a CICB switch that uses memory efficiently.

To improve the throughput performance of a shared memory switch under multicast traffic, a novel switch architecture is proposed to support multicast traffic using a shared-memory switch that shares crosspoint buffers among outputs to use 50% of the memory amount in the crossbar fabric that CICB switches require. The proposed O-SMCB switch,

delivers high performance under multicast traffic while using no speedup. Furthermore, the proposed switch shows an improved performance over I-SMCB switch. The improved switch is based on having the buffers shared by the outputs instead of the inputs. This has the effect of facilitating call splitting by allowing inputs directly access the crosspoint buffers. This simple improvement has a significant impact on switching performance. As a result, the O-SMCB provides 100% throughput under both admissible uniform and diagonal multicast traffic with fanouts of 2 and 4. Furthermore, our proposed switch keeps the throughput high under nonuniform traffic with overloading conditions. The disadvantage of SMCB switches is that time relaxation of CICB switches is minimized because of the matching process used in buffer access. However, the matching is performed in chip and among a moderate number of outputs. Furthermore, the matching process is simpler in the SMCB switches than those used in IB switches for multicast traffic. The O-SMCB switch, with buffer space for $\frac{N^2}{2}$ cells, provides comparable performance to that of a CICB switch, with buffer space for N^2 cells, therefore, saving 50% of the amount of memory.

CHAPTER 5

LOAD-BALANCED CICB SWITCH

5.1 Introduction

SMCB switches reduce the memory amount in the buffered crossbar by half or more comparing to CICB switches. However, when switch size is large and RTT is long, the memory amount in the buffered crossbar needs to be further reduced. By considering the Birkhoff-Von-Neumann switch [48], this chapter proposes a class of load-balanced combined-input crosspoint buffered switches which use a load-balancing stage with a buffered crossbar to relax the crosspoint buffer size such that flows with high data rates can be handled when $k < RTT$ and first-come first-serve (FCFS) scheduling scheme at the output to keep cells from the same flow in sequence [28]. It is shown that these new architectures support high data-rate flows and $RTT = kN$, which results in a crosspoint-buffer of size $\frac{1}{N}$ of that in a switch without load-balancing stage.

5.2 Throughput Degradation in CICB Switches with Rigid Access

The unbalanced traffic load, as defined in Section 2.1, can be represented by matrix $\bar{\rho}$ as:

$$\bar{\rho} = \rho \begin{pmatrix} w + \frac{1-w}{N} & \dots & \frac{1-w}{N} \\ \vdots & \ddots & \vdots \\ \frac{1-w}{N} & \dots & w + \frac{1-w}{N} \end{pmatrix}$$

The performance evaluation of a CICB switch is presented in [49]. In this dissertation, a new term, the refill-ratio, is defined. The refill-ratio matrix M is the ratio of crosspoint buffer size and the round trip time. The element $m_{i,j}$ in matrix M is defined by k/RTT. The maximum allowable output rate P^{max} is given by $P^{max} = \bar{R} \cdot M$. The output of rate P is defined as the minimum of maximum allowable output rate and the

corresponding input load, $P_{i,j} = min(\bar{R}_{i,j}, P_{i,j}^{max})$. The maximum throughput of a CICB switch is given by:

$$Throughput = \frac{\sum\sum P_{i,j}}{\sum\sum \bar{R}_{i,j}}. \tag{5.1}$$

Using (5.1), the throughput that a switch can deliver is calculated without considering the effect of the arbitration schemes. If the unbalanced traffic model is used as input load, it shows that the simulated result has the same behavior as obtained by (5.1). This indicates that degradation can be present in a CICB switch with rigid crosspoint buffer access, independent of the arbitration scheme for high data rates ($w = 1$).

5.3 Load-Balanced Combined-Input Crosspoint Buffered Switches

5.3.1 Load-Balanced CICB Full Access (CICB-FA) to Crosspoint Buffers

The $N \times N$ Load-balanced CICB switch has VOQs at the input ports, a fully interconnected stage that provides connectivity for input i to any of the N crosspoints of output j, and a buffered crossbar. The switch architecture is shown in Figure 5.1. The fully interconnecting stage is combined with the buffered crossbar. A crosspoint in the buffered crossbar is denoted as $XP(h, l)$, where $0 \leq h \leq N - 1$,[4] and the corresponding crosspoint buffer is denoted $XPB(h, j)$. Input i can access $XPB(h, j)$. A crosspoint has a $N - to - 1$ multiplexer, denoted as $MUX(h, j)$, per crosspoint buffer. An input of $MUX(h, j)$ transfers cells from input i. In this way, input i can access any $XPB(h, j)$. The buffered crossbar holds an output arbiter per output and an access scheduler per input. A crosspoint buffer can receive up to one cell from one out of N inputs. Inputs can send up to one cell per time slot. At the buffered crossbar, there are N virtual VOQ counters, each denoted as $VC(i, j)$, which counts the number of cells at $VOQ(i, j)$ that have not been scheduled for forwarding to the buffered crossbar. There is a crosspoint access scheduler per output j

[4]Note that an XP may not be have a one-to-one association with a VOQ as in CICB switches with rigid access.

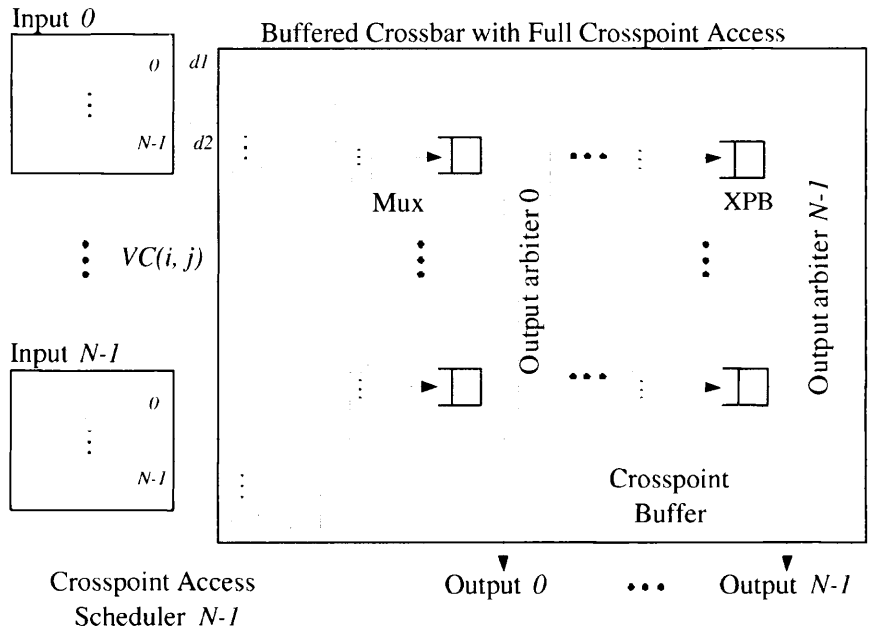

Figure 5.1 $N \times N$ CICB switch with full access (CICB-FA).

that determines which input access a given crosspoint buffer. It is considered the crosspoint buffers have size $k \geq 1$ and use no speedup.

The way this switch works is as follows. After a cell destined to output j arrives in input i, the cell is stored in $VOQ(i, j)$ and a request is sent to the access scheduler at output j. The request is kept in $VC(i, j)$, located at the buffered crossbar. The access scheduler selects up to N cells for crosspoints at output j after considering all those non-empty VCs and available XPBs. The access scheduler notifies the inputs who were selected. Since an input may be granted for XPBs at different outputs. The access scheduler performs matching between inputs and XPBs. After being notified, an input sends the cell to the crosspoint buffer. The output arbiter at output j selects an occupied crosspoint buffer to forward a cell to the output in a first-come first-serve (FCFS) fashion. This switch uses no speedup as inputs and crosspoints process one cell per time slot.

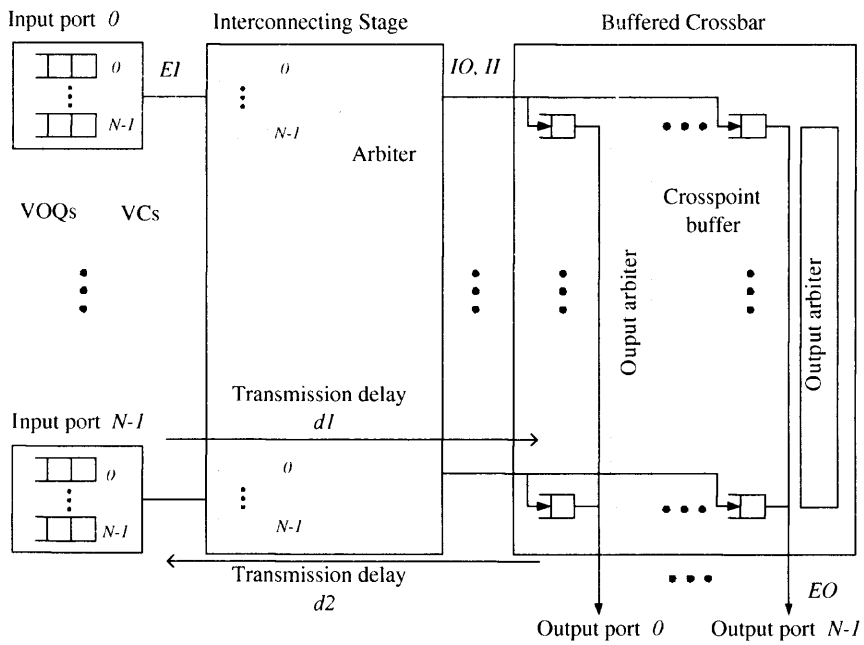

Figure 5.2 $N \times N$ CICB switch with single access (CICB-SA).

5.3.2 Load-Balanced CICB Switch with Single Access (CICB-SA) to Crosspoint Buffers

The switch with full access has N^2 $N - to - 1$ multiplexers. In addition, the crosspoint access scheduler needs to perform matching between inputs and outputs. To minimize the complexity and hardware amount, a simpler CICB switch with flexible access to crosspoint buffers, the CICB switch with single access (CICB-SA) is introduced.

This switch has VOQs in the input ports, an interconnecting stage that uses predetermined and cyclic configurations, as those used in the Birkhoff-Von-Neumann switch [48], connecting its inputs and outputs in a one-to-one fashion, and a buffered crossbar. Figure 5.2 shows this switch. In this switch, the input ports are also called external inputs, each of which is denoted as EI_i. The outputs of the interconnecting stage are called internal outputs, each of which is denoted as IO_l, where $0 \leq l \leq N - 1$. IOs are physically equivalent to the inputs of the buffered crossbar, also called internal inputs, each of which is denoted as II_l. The outputs of the buffered crossbar, or output ports, are also called external outputs, each of which denoted as EO_j.

As in the CICB-FA switch, there also are N VOQ counters, $VC(i,j)$, in each input. In each II_l, there are N crosspoint access schedulers, each denoted as (CAS_l), which schedules access of a cell from input i to XPB_l (via II_l). A CAS is comprised by an arbiter at EI_i that selects a crosspoint buffer by using round-robin selection and the predetermined configuration of the interconnecting stage. The way this switch works is as follows. At EI_i, cells with destination to output j arrive in $VOQ(i,j)$ and send a request indicating the arrival to $VC(i,j)$. The CAS uses a round-robin schedule to select a non-empty and non-inhibited (by the flow-control mechanism) VC. At a scheduling time t, the configuration of the interconnecting stage pairs EI_i to II_l by $l = (i+t) \bmod N$. A CAS schedules a cell for $XPB(l,j)$ at time t if this XPB has room available for one cell and if $VC(i,j) > 0$. Once a cell is dispatched by the input, the packet traverses the interconnecting stage and is held at the XPB. A cell going from EI_i to EO_j may enter the buffered crossbar through II_l and be stored in $XPB(l,j)$. Cells leave EO_j after being selected by the output arbiter. The output arbiter also uses FCFS selection to keep cells of $f(i,j)$ in order. The output arbiter considers the time when a cell arrives at the crosspoint buffer to perform FCFS among dedicated crosspoint buffers. It is studied in [50] of keeping cells in order in two-stage switches. Section 5.4 presents the proof of keeping cells in order by the FCFS arbitration. Cells and flow control data experience the transmission delay from input ports to the buffered crossbar.

5.4 In-Sequence Cell Service with FCFS Output Arbitration

An advantage of using a CICB switch is that all crosspoints buffers are located on chip. This makes it easy to keep the arrival time of incoming cells and to use a simple output arbitration scheme to keep cells in sequence [51]. This is discussed by the following theorem.

Theorem 5.1. *Cells are served in-sequence when First-Come First-Serve (FCFS) is used as selection policy in the output arbiter of a load-balanced CICB switch for any XPB size.*

Proof. Let's identify a cell C by appending the following labels to the cell: EIi, the destined EOj, and the sequence serving order as identification label, or $C(i, j, t)$. The sequence serving order is assigned to a cell at the time the cell arrives at the buffered crossbar.

Consider the following facts:

Fact 1 Each internal input II assigns a timestamp t to a cell at arrival in the XPB;

Fact 2 Each $C(i, j, t)$ from EIi goes to queues at EOj;

Fact 3 An output arbiter uses FCFS selection policy from all $XPB(i, j)$, by selecting the cell that entered first (e.g., smallest t).

Therefore, the output arbiter at output j considers t of the head-of-line (HOL) cells to perform selection (for the set of cells with the same t, a cell is arbitrarily selected).

The order (or sequence) in which cells depart output j depends on the order they arrive at the queues, and in the order they are selected, as HOL cells, for departure by the output arbiter. Therefore, the proof of Theorem 1 is partitioned into the following two lemmas.

Lemma 5.1. *A cell $C_1(g, j, t_1)$ always arrives at the HOL cell in $XPB(k, j)$ for II_k before $C_2(h, j, t_2)$ if $t_1 < t_2$.*

Proof. The order of cell arrivals to an XPB depends on the departure order from the inputs and on the insertion order at the XPB.

Departure from inputs: since there are no buffers between the inputs and the XPBs, the cells arrive at their corresponding XPB_j in the order they departed from the input ports.

Insertion at XPB: because the insertion policy at XPBs is a first-in first-out (FIFO), if cell C_2 arrives at t_1 and C_3 at t_2, C_2 will always be placed ahead (i.e., closer to the HOL) of cell C_3, when $t_1 < t_2$. □

Consider now two cells from two different flows, one from Flow 1 and the other from Flow 2, destined to the same output, such that the cell from Flow 1 arrives a time t_x and

cell from Flow 2 arrives at t_y, where $t_y > t_x$, to different crosspoints (i.e., queues). If the cell from Flow 2 is served first, then it is possible that there is a cell from Flow 2 blocked by cell from Flow 1, therefore, causing out-of-sequence delivery. The departure time of a cell should occur when there is at least one cell in the system with the same t or when all remaining cells have t', such that $t' > t$ and the sequence service number of the cell departed last had $T \leq t'$. It then remains to prove that any two cells arriving at different times are served according to the arrival order, as discussed by the following lemma.

Lemma 5.2. *A cell $C_3(l, j, t_3)$ in the $XPB(k, j)$ will always be served before cell $C_4(m, j, t_5)$ at $XPB(q, j)$, for $t_3 < t_5$ by a FCFS arbiter, independent of the cell position in the queues for output j.*

If this is true, there exist a cell $C_5(l, j, t_4)$, where $t_4 < t_5$, such that C_5 is never be blocked by C_3, and C_5 leaves the switch before cell C_4, where C_5 and C_4 belong to the same flow (i.e., $VOQ(i, j)$).

Proof. Assume that crosspoint queue k has Cell $C_4(m, j, t_5)$ at the HOL, and crosspoint queue q has cell $C_6(n, j, t_2)$ at the HOL, which is followed by $C_3(l, j, t_3)$, and C_3 is followed by $C_5(m, j, t_4)$. Figure 5.3 shows this cell ordering at output j. This placement of cells seems to be favorable for C_4 to depart before C_3, and therefore, before C_5. However, as the arbiter selects a cell with the smaller sequence service number, C_4 is held in the queue until C_3 and C_5 are served as a queue never has cells with unsorted sequence service number, as defined by Lemma 1. □

Since Lemmas 1 and 2 are true. Theorem 1 is also true. □

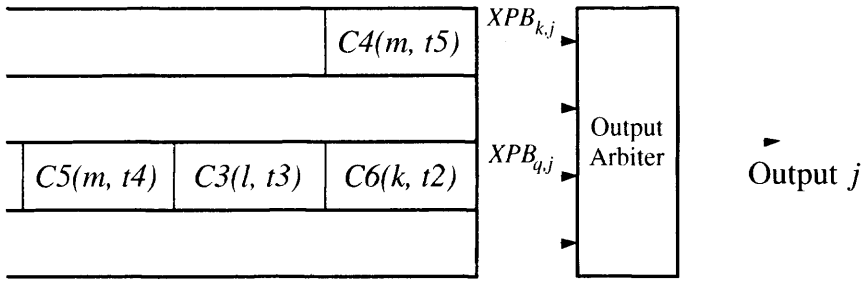

Figure 5.3 Example of cell placement described in Lemma 2.

5.5 Throughput Analysis under Admissible Independent Identical Distributed Traffic and Random Selection Scheme

In this section, the throughput analysis of CICB switch and load balanced CICB switch under admissible independent identical distributed (i.i.d.) traffic is presented. The following assumptions are made in the analysis:

- The arriving traffic at each input port is Poisson process.
- The arrival process to the crosspoint buffer is also Poisson process.
- Input and output scheduling follows random selection.
- An EI connects with II on a random base with probability $\frac{1}{N}$.

The following notation is used for an $N \times N$ load-balanced CICB switch.

- ρ - input load of the switch, $0 \leq \rho \leq 1$.
- $\lambda_{i,j}$ - arrival rate of flow (i, j).
- $\lambda_{i,j}^{X}$ - arrival rate at $XPB(i, j)$.
- $\vec{\Lambda_i}$ - arrival vector for input i, $0 \leq i \leq N-1$. $\vec{\Lambda_i} = [\lambda_{i,j}], 0 \leq j \leq N-1$.
- $\mu_{i,j}$ - service rate at each output port.
- P_{si} - the state probability that there are i cells in the queue.
- $P_{i,j}$ - transition probability from state i to state j, in another word, the transition probability from the state that there are i cells in the queue to the state that there are j cells in the queue.

- n - size of a VOQ in number of cells.
- N - switch size.
- k - XPB size.

Subsection 5.5.1 and Subsection 5.5.2 show the throughput analysis of CICB switch and load balanced CICB switch under admissible i.i.d. traffic, respectively.

5.5.1 CICB Switch

This section presents the analysis of the stability of the CICB switch under admissible i.i.d. traffic. Each VOQ is modeled as an $M/M/1$ queue. The state probability of n cells in the VOQ is given

$$P_{sn} = (\frac{\lambda_{i,j}}{\mu_{i,j}})^n P_0, \quad n = 1, 2 \cdots \tag{5.2}$$

$$P_{s0} = \frac{1 - \frac{\lambda_{i,j}}{\mu_{i,j}}}{1 - (\frac{\lambda_{i,j}}{\mu_{i,j}})^{n+1}} \tag{5.3}$$

The service rate $\mu_{i,j}$ depends on the probability that the corresponding crosspoint buffer is available. Since it is assumed that the arrival to the crosspoints is also Poisson process, each crosspoint buffer is modeled as $M/M/1/k$ queue, where k is the crosspoint buffer size. In this chapter, $k = 1$ is used in the analysis. The probability that a crosspoint is available is $1 - P_{sk}$, where P_{sk} is the state probability that there are k cells at the crosspoint. Therefore, $\mu_{i,j} = \frac{1}{N}(1 - P_{sk})$.

$$P_{sk} = \frac{1 - \frac{\lambda^X_{i,j}}{\mu^X_{i,j}}}{1 - (\frac{\lambda^X_{i,j}}{\mu^X_{i,j}})^{k+1}} (\frac{\lambda^X_{i,j}}{\mu^X_{i,j}})^k. \tag{5.4}$$

Here $\lambda^X_{i,j} = \frac{1}{N}\lambda_{i,j}$, $\mu^X_{i,j} = \frac{1}{N}$

From equation 5.3 and 5.4,

$$P_{sn} = \frac{[N\lambda(1+\lambda)]^n - [N\lambda(1+\lambda)]^{n+1}}{1 - [N\lambda(1+\lambda)]^{n+1}} \tag{5.5}$$

when $k = 1$.

5.5.2 Load Balanced CICB Switch

This section presents the proof of the 100% throughput of the load-balanced CICB switch under admissible independent identical distributed (i.i.d.) traffic. Each VOQ is modeled as an $M/M/1$ queue. Since arrivals are independent and identical distributed. All VOQs that are destined to the same output can be viewed as a superposed Markov process with arrival rate $\lambda_j = \sum_{i=0}^{N-1} \lambda_{i,j}$. The integrated process VOQ_j is presented. VOQ_j can access all N XPBs for output j N times at each time slot. This can be modeled using $M/M/N$ queue as shown in figure 5.4.

Here $\rho = \frac{\lambda_j}{N\mu^I}$ is defined.

The steady state probability of n cells in VOQ_j is represented as P_{sn}. It's calculated using the following equations.

$$P_{sn} = \begin{cases} P_{s0}\frac{(N\rho)^n}{n!}, & n \leq N \\ P_{s0}\frac{\rho^n N^N}{N!}, & n > N \end{cases} \tag{5.6}$$

$$P_{s0} = [\sum_{n=0}^{N-1} \frac{(N\rho)^n}{n!} + \frac{(N\rho)^N}{N!(1-\rho)}]^{-1} \tag{5.7}$$

$$P_{sn} = \begin{cases} [\sum_{n=0}^{N-1} \frac{(N\rho)^n}{n!} + \frac{(N\rho)^N}{N!(1-\rho)}]^{-1} \frac{(N\rho)^n}{n!}, & n \leq N \\ [\sum_{n=0}^{N-1} \frac{(N\rho)^n}{n!} + \frac{(N\rho)^N}{N!(1-\rho)}]^{-1} \frac{\rho^n N^N}{N!}, & n > N \end{cases} \tag{5.8}$$

The service rate of the $M/M/N$ queue depends on the availability of the XPBs for output j.

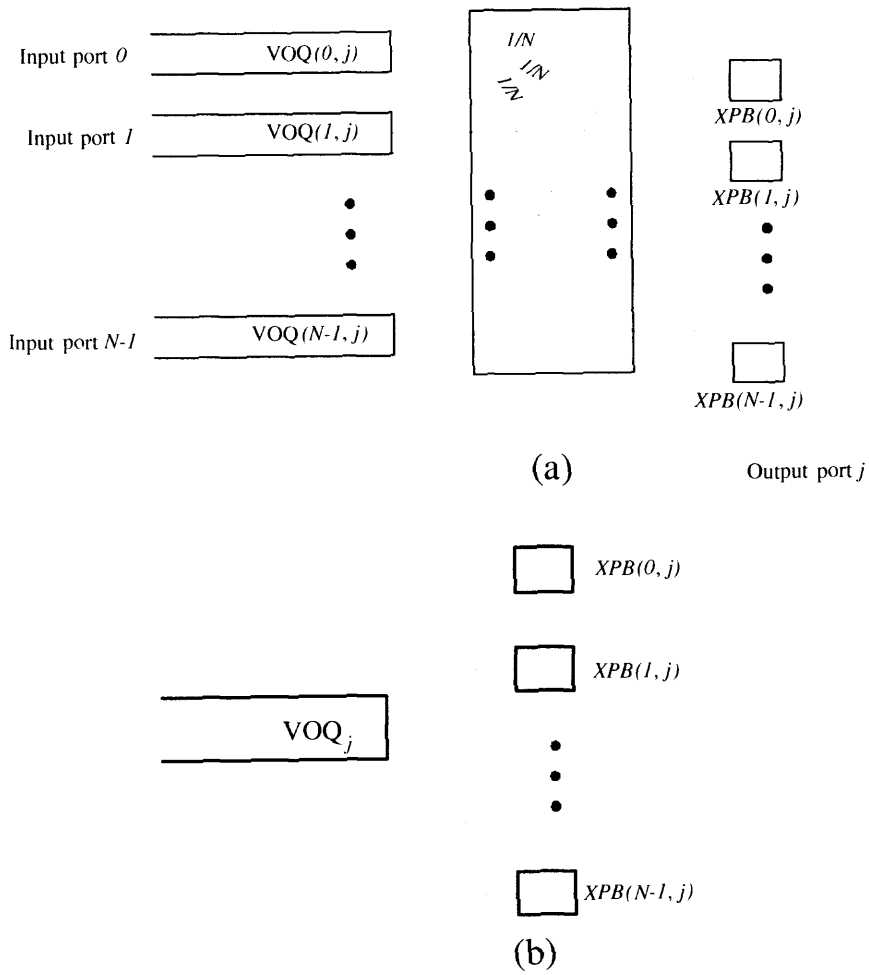

Figure 5.4 M/M/N queuing model of load balanced CICB switch.

$XPB(i,j)$ can be modeled as $M/M/1/k$ queue where k is the XPB size. Here k is set to one. The average arrival rate to each XPB queue after the load balancing stage is calculated as $\lambda_{i,j}^X = \sum_{i=0}^{N-1} \lambda_{i,j} \frac{1}{N}$

The state probability that n cells reside in a VOQ is calculated by

$$P_{sn} = \begin{cases} [\prod_{t=1}^{n} \frac{\rho \cdot \lambda_{i,j}}{t \cdot \mu}] \cdot P_{s0}, & n \leq N \\ [\frac{1}{N!} \cdot (\frac{\lambda}{\mu})^N][\prod_{t=N+1}^{n} \frac{\lambda}{N \cdot \mu}] \cdot P_{s0}, & n > N \end{cases} \tag{5.9}$$

$$P_{s0} = [1 + \sum_{t=1}^{N-1} \frac{1}{t!} \cdot (\frac{\lambda}{\mu})^t + \sum_{t=N}^{n} \frac{1}{N! N^{t-N}} (\frac{\lambda}{\mu})^t]^{-1} \tag{5.10}$$

The crosspoint buffer size is set to one in the analysis. The probability that $XPB(i,j)$ is available is calculated using $M/M/1$ queueing model. To differentiate the notation with VOQs at the inputs, the superscript X is used in the notations to represent the XP queues.

P_{ij}^X - transition probability from state i to state j.

P_{si}^X - state probability that there are i cells in the XP queue.

$P_{01}^X = \sum_{i=0}^{N-1} \frac{1}{N} \cdot \frac{\lambda_{i,j}}{\sum_{j=0}^{N-1} \lambda_{i,j}} \cdot \rho \cdot \lambda_{i,j}$

$P_{10}^X = \frac{1}{N}$

$$\begin{cases} P_{01}^X P_{S0}^X = P_{10}^X P_{S1}^X; \\ P_{S0}^X + P_{S1}^X = 1; \end{cases} \tag{5.11}$$

$$P_{s0}^X = \frac{P_{10}^X}{P_{10}^X + P_{01}^X} \tag{5.12}$$

The arrival to each XPB queue after the load balancing stage is estimated $\lambda_{i,j}^X = \sum_{i=0}^{N-1} \lambda_{i,j} \cdot \frac{1}{N}$.

It is assumed that the output scheduling is random in the calculation $\mu^I = P_{s0}^X$.

Under admissible i.i.d traffic, $\sum_{i=0}^{N-1} \lambda_{i,j} \leq 1$. Therefore, $\lambda_{i,j}^{X} \leq \frac{1}{N}$, $\frac{\lambda}{\rho_{max}} = \frac{2}{N}$.

$$P_{sn} = \frac{N^{N-n}}{N![\sum_{n=0}^{N-1} \frac{1}{n!} + \frac{N}{N!(N-2)}]} \tag{5.13}$$

As $n \rightarrow \infty$, $\lim_{n \rightarrow \infty} P_{sn} = 0$.

Therefore, the length of VOQ, n, converges to ε, where $\varepsilon < \infty$, $\lim_{n \rightarrow \infty} P_{sn} > B < \varepsilon, B > 0$.

The switch is said to be weakly stable if the above condition is met [52].

5.6 Performance Analysis

Several traffic patterns and flow data rates are considered in the performance study of the proposed two switches. First, it is shown that the performance under traffic with uniform distributions remains high as that delivered by CICB switches with rigid crosspoint access. Second, it is shown that the proposed switches deliver higher throughput than CICB switches with rigid access under nonuniform traffic patterns, such as the unbalanced, diagonal, and power-of-two traffic models. The throughput is 100% for $RTT \leq k$. Furthermore, these switches can deliver close to 100% throughput under admissible traffic patterns for $RTT > k$ using a weight-based arbitration. This is a unique feature of these switches as CICB switches with fixed access cannot support such long RTT values.

5.6.1 Uniform Traffic

Figure 5.5 shows the average cell delay of a CICB switch using longest queue first (LQF) as input arbitration and FCFS as output arbitration, a CICB-SA switch, and a CICB-FA switch, all under uniform traffic. The average cell delay only considers the queuing delay. For low loads, the CICB has smaller average cell delay. However, in the CICB-SA and CICB-FA switches, cells spend an extra time slot at the input queues as their requests are sent to the crosspoint access scheduler and that have to be granted ($RTT = 1$). The total

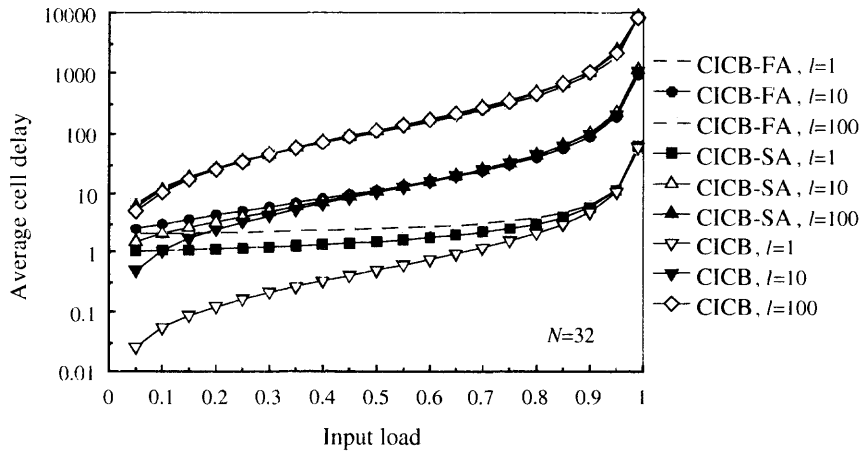

Figure 5.5 Average queuing delay of a 32×32 CICB switches under uniform traffic.

delay is small in any case. In general, all switches have similar average delays, which indicates that the flexible access scheduling has not measurable effect in the switching performance. The figure also shows the average delay of all switches under bursty traffic with average burst lengths $l = \{10, 100\}$. The simulation results show that the delay increases in proportion to the burst length.

5.6.2 Nonuniform Traffic: Unbalanced

The effect of long RTTs in the proposed switch model can be studied by measuring the switch throughput under the unbalanced traffic model, as in Section 2.1. Figure 5.6 shows the throughput performance of the CICB, CICB-SA, and CICB-FA switches when $k = 1$ for different RTTs. The CICB switch has the throughput degraded under $RTT > k$ as w increases. The worst case is reached at $w = 1.0$ as discussed in Section 2.1. CICB-SA and CICB-FA switches have their throughput high despite the increase of RTT. However, both switches have their throughput below 99% when $RTT \geq 31$. Furthermore, the CICB-FA switch has the highest throughput when $RTT = N = 32$. While the throughputs of all switches with flexible access decreases for different values of w, their throughputs remain high when $w = 1.0$, which is the case for high data rate flows, while a switch with rigid access has the throughput degraded to $\frac{k}{RTT}$.

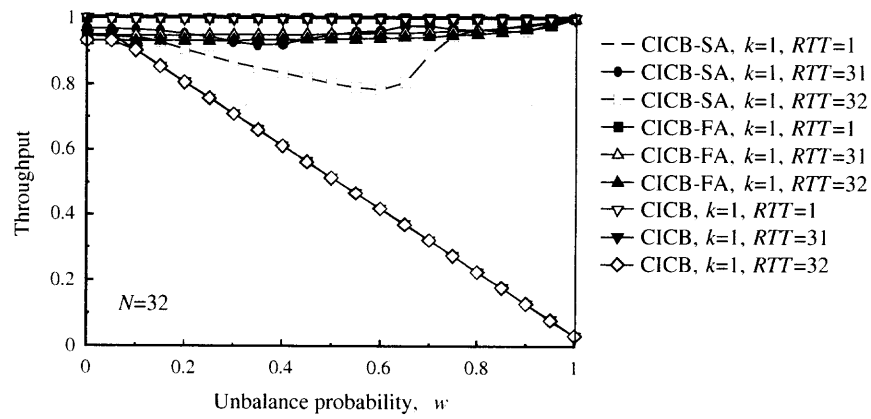

Figure 5.6 Throughput of CICB switches with k=1 and $RTT = \{1, 31, 32\}$ under unbalanced traffic.

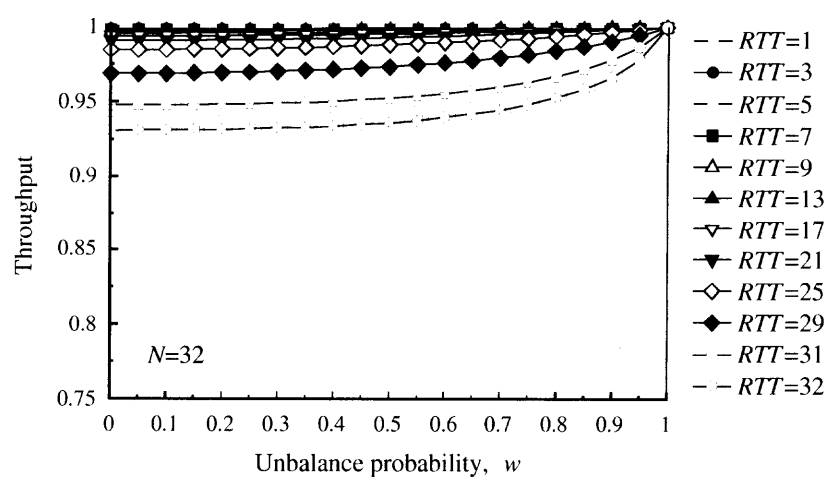

Figure 5.7 Throughput of the 32×32 CICB-FA switch with k=1 under unbalanced traffic.

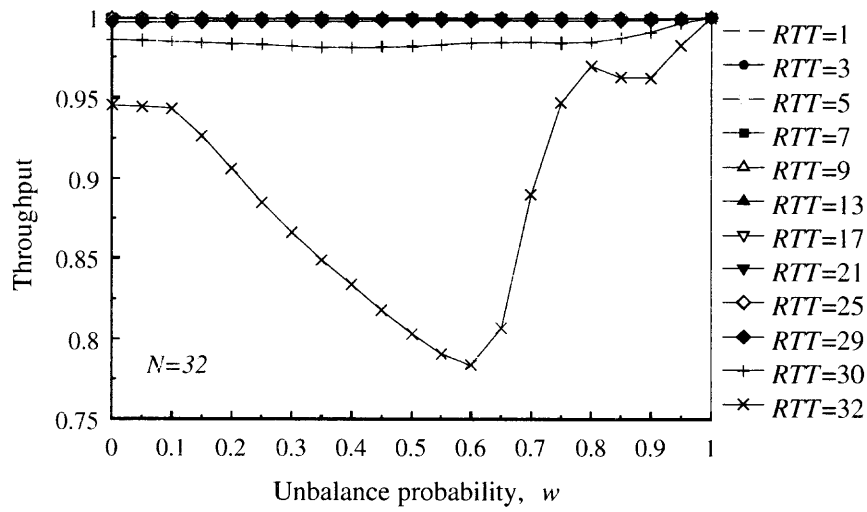

Figure 5.8 Throughput of the 32×32 CICB-SA switch with k=1 under unbalanced traffic.

The throughput of the CICB-SA and CICB-FA switches, under different RTT values and $k = 1$ is shown in Figures 5.7 and 5.8. It is shown in Figure 5.7 that the CICB-FA switch delivers close to 100% throughput for $RTT \leq 21$. For larger RTT values, the throughput falls below 99%. The throughput is the lowest when $w = 0.0$ (i.e., uniform distribution) or for flows with low data rates.

As Figure 5.8 shows, the throughput of CICB-SA is higher than that of CICB-FA. The throughput of CICB-SA is close to 100% when $RTT \leq 29$ and decreases slightly, although below 99%, when $RTT = 31$. For $RTT \geq 32$, the throughput decreases rapidly.

5.6.3 Nonuniform Traffic: Power of Two

In addition, CICB-SA and CICB-FA switches were simulated under power-of-two traffic [13] for 30×30 switches. This traffic model presents a large nonuniformity in the traffic distribution among N possible destinations. It is shown in Figure 5.9 that both switches deliver 100% throughput under this traffic pattern for $RTT = 1$ and $k = 1$.

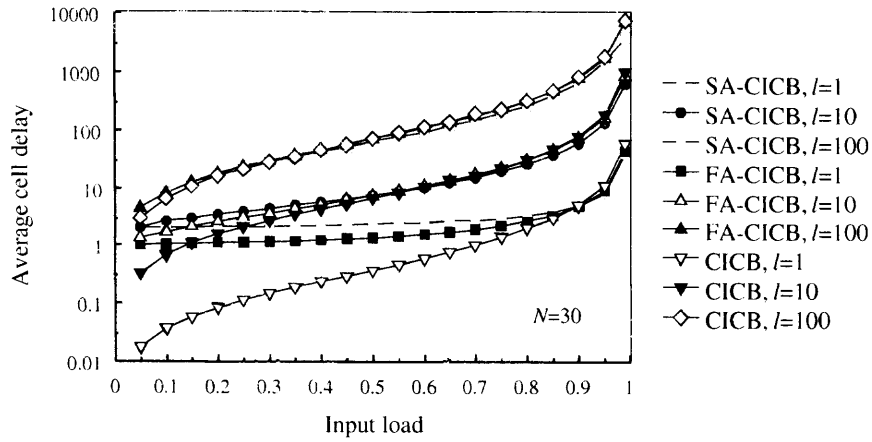

Figure 5.9 Performance of 30×30 switches with $k = 1$ under PO2 traffic.

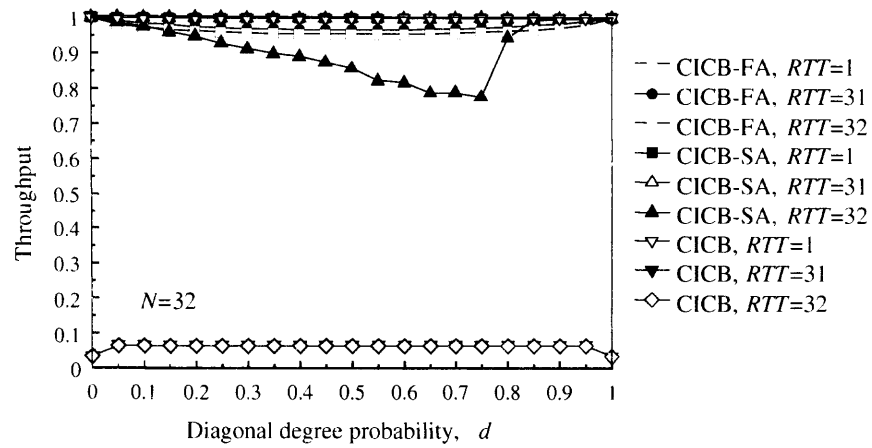

Figure 5.10 Throughput of the 32×32 switches with $k = 1$ under diagonal traffic.

5.6.4 Nonuniform Traffic: Diagonal

This traffic model presents load distributions among two outputs per each input. The distributions are given by the diagonal degree probability, d. Figure 5.10 shows the switching performance of the CICB-FA and CICB-SA switches under diagonal traffic for $0 \leq d \leq 1$. This figure shows that these two switches can support $RTT \leq 31$ and achieve close to 100% throughput.

5.7 Conclusions

In this chapter, it is observed that switches based on buffered crossbars with dedicated crosspoint-buffer access have their maximum throughput as the ratio of $\frac{k}{RTT}$, when input ports handle a single flow with a data rate equal to the port capacity. To minimize the crosspoint-buffer size, a switch model that uses a load-balancing stage in front of the buffered crossbar, such that inputs can flexibly access different crosspoint buffers is proposed. It is proved that the load balanced CICB switch is weakly stable under admissible i.i.d traffic. The proposed switch supports RTTs that can be kN-time-slot long, while providing 100% throughput for such high data-rate flows. As a comparison, for a given RTT size, the load-balanced CICB switch requires a minimum $k = \frac{RTT}{N}$ cells while a simple CICB switch requires a minimum $k = RTT$ cells. Therefore, the proposed switch relaxes the amount of memory to $\frac{1}{N}$ of the amount required by a CICB switch with dedicated access to crosspoint buffers.

CHAPTER 6

MEMORY-MEMORY-MEMORY CLOS-NETWORK SWITCH

6.1 Introduction

In the previous chapters, different single-stage crosspoint buffered switches were proposed and studied to support differentiated services, multicast traffic, and long RTTs. Most high-performance packet switches are currently based on a single-stage switch fabric. The single-stage crosspoint buffered switch has the limitation of not being scalable because of implementation constraints, such as pin count on a die. To support larger number of ports, multi-stage switches have been introduced.

Clos-network switch is a multiple-stage switch [53], constructed using smaller switch modules that are arranged in three stages. Each module is implemented using a crossbar switch fabric. The key challenge for the implementation of a Clos-network switch is the design of routing/scheduling algorithms that efficiently route and dispatch packets to avoid contention [9], [11]. A buffer-less Clos-network switch is also called a space-space-space (S^3) Clos-network switch since switching in all three stages is done based on space. To resolve internal and output contention, buffers are introduced to Clos-network switch. The buffer placement in the first and third stages has been proposed to give place to memory-space-memory (MSM) Clos-network switches [11][54]. Several schemes for dispatching cells from the first-stage modules to the third-stage modules have been proposed [11], [55],[56]. However, MSM Clos-network switches may still require long resolution time for configuration. A Clos-network switch with memory in all stages or memory-memory-memory (MMM) Clos-network switches can not only be considered as the straight forward architecture to scale up single-stage buffered crossbar switches, but also to reduce the configuration time of a Clos-network switch by performing separate selection of cells at each

stage. Some examples of such MMM Clos-network switches have been proposed in [57], [58].

This chapter studies memory-memory-memory (MMM) Clos-network switches with modules at each stage implemented using a crosspoint buffered switch fabric. This switch can provide high throughput performance and eliminate the need for complicated centralized routing and dispatching algorithms. Because a Clos-network switch provides multiple eligible paths to connect an input to an output, cells may be delivered out-of-sequence. This chapter shows that an MMM Clos-network switch can use simple buffered crosspoint switches as switch modules, with however low switching performance in trade-off for simplicity. It is also shown that the effect of having more memory in the central module in an MMM switch can provide throughput improvement. To solve the out-of-sequence problem of the MMM Clos-network switch, a new switch architecture that uses extended memory in all modules and oldest cell first scheduling scheme is proposed.

6.2 Memory-Memory-Memory Clos-Network Switch with Minimum Memory Amount at CM (MM^{m}M)

A buffered Clos-network packet switch consists of three stages of switch modules, input modules (IMs), central modules (CMs), and output modules (OMs). In an $N \times N$ switch, there are p IMs, m CMs and p OMs. IMs/OMs have n input/output ports, respectively. Here $p \times n = N$. Each IM is an $n \times m$ crosspoint buffered switch, each OM is an $m \times n$ crosspoint buffered switch, and each CM is a $p \times p$ crosspoint buffered switch. This section studies this MMM Clos-network switch with minimum memory amount in the central module architecture (MM^{m}M). Figure 6.1 shows an $N \times N$ MM^{m}M Clos-network packet switch.

The following terminology is used in this section:

- i: IM index, where $0 \leq i \leq p-1$.
- j: OM index, where $0 \leq j \leq p-1$.

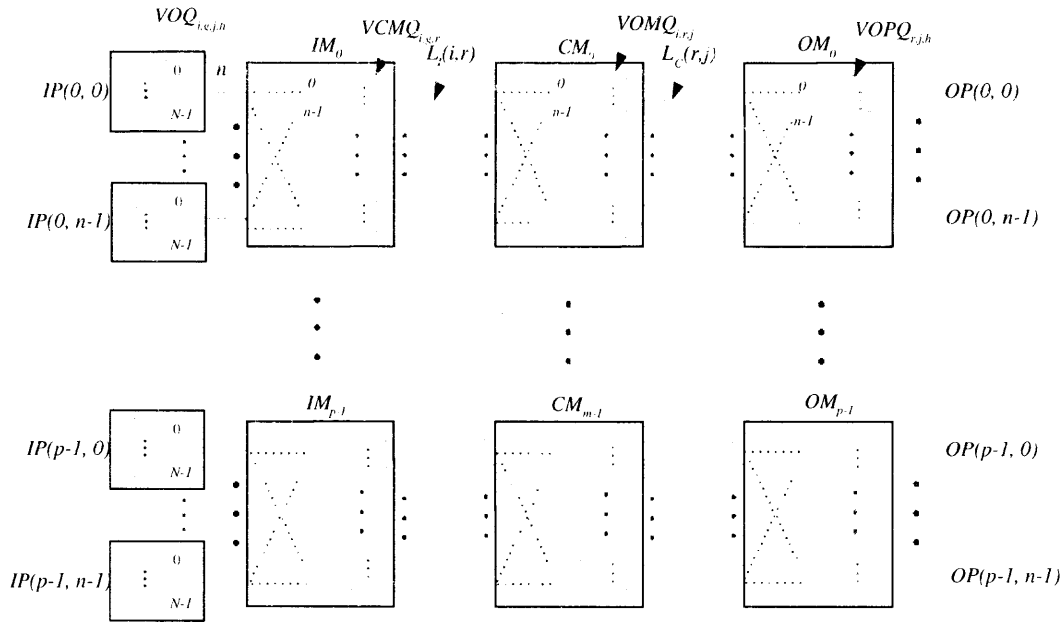

Figure 6.1 $N \times N$ MM^{m}M Clos-network packet switch.

- g: IP number in each IM, where $0 \leq h \leq n-1$.
- h: OP number in each OM, where $0 \leq h \leq n-1$.
- k_1: Size of crosspoint buffers in IMs.
- k_2: Size of crosspoint buffers in CMs.
- k_3: Size of crosspoint buffers in OMs.
- $IM(i)$: The (i+1)th input module, where $0 \leq i \leq p-1$.
- $CM(r)$: The (r+1)th central module, where $0 \leq r \leq m-1$.
- $OM(j)$: The (j+1)th output module, where $0 \leq j \leq p-1$.
- $IP(i,g)$: The (g+1)th input port at $IM(i)$, where $0 \leq g \leq n-1$.
- $OP(j,h)$: The (h+1)th output port at $OM(j)$, where $0 \leq h \leq n-1$.
- $L_I(i,r)$: Output link of $IM(i)$ that is connected to $CM(r)$.
- $L_C(r,j)$: Output link of $CM(r)$ that is connected to $OM(j)$.
- $S_I(i,r)$: Scheduler for $L_I(i,r)$ at $IM(i)$.
- $S_C(r,j)$: Scheduler for $L_I(r,j)$ at $CM(r)$.
- $S_O(j,h)$: Scheduler for $OP(j,h)$ at $OM(j)$.

- $VOQ(i, g, j, h)$: Virtual output queue at $IP(i, g)$ that stores cells destined to $OP(j, h)$.
- $VCMQ(i, g, r)$: Virtual central module queue at $IM(i)$ that stores cells coming from $IP(i, g)$ and scheduled to go through $CM(r)$ to its destination port.
- $VOMQ(i, r, j)$: Virtual output module queue at $CM(r)$ that stores cells coming from $IM(i)$ and destined to $OM(j)$.
- $VOPQ(r, j, h)$: Virtual output port queue at $OM(j)$ that stores cells coming from $CM(r)$ and destined to $OP(j, h)$.

Input ports: There are N VOQs at each input port to avoid HOL blocking. Each VOQ stores cells destined to an output port. There is one arbiter at each input port. The arbiter schedules cells to be forwarded to the queues in the IM of that input. The arbitration scheme used at input ports can be either round-robin (RR) or longest queue first (LQF).

Input modules: An IM has $m \times p$ virtual central module queues (VCMQs). Each queue stores cells that are destined to $OM(j)$ routed through $CM(r)$. There are m arbiters in each IM, one per output link. The arbiter $L_I(i, r)$ at $IM(I)$ selects $CM(r)$ that a cell coming from an input is to be routed. The selection of $CM(r)$ is based on round-robin selection. The cells are then stored in the selected VCMQ at link to $CM(r)$.

Central modules: Each CM has $p \times n$ VOMQs, each of which stores cells coming from $L_I(i, r)$ of $IM(i)$ destined to $OM(j)$. The arbiter at each output link uses round-robin selection to select a cell to be forwarded to its destined OM.

Output modules: At each OM there are $m \times n$ VOPQs, each of which stores cells coming from $L_C(r, j)$ and destined to $OP(j, h)$. The arbiters at OM use round-robin selection to select a cell to be forwarded to the output.

The switch works as follows. Cells arriving at the inputs are stored in VOQs based on their destination ports. The arbiter at the input ports selects an eligible VOQ to forward a cell to the IM. A VOQ is considered eligible when it is non-empty and the destined VCMQ in the IM is available. The selection of VCMQ a cell is to be sent from is based on round-robin. The arbiter at the output link of the IM selects a cell at a VCMQ destined to a VOMQ at an assigned CM on round-robin fashion. The cell is forwarded to the VOMQ and awaits

for arbitration. Arbiters at output links of CMs select a cell at VOMQs to be sent to VOPQ, also in a round-robin fashion. After a cell is forwarded to the destined OM, it is stored in VOPQs and awaits for output arbitration to be forwarded to the output port. Output arbiters also perform round-robin selection among available cells at VOPQs.

6.3 Memory-Memory-Memory Clos-Network Switch with Extended Memory at CMs (MM^{e}M)

As it can be seen, cells from the same flow can be routed through different (or same) CMs in the MMM switch. A cell that has a given time stamp may be scheduled to go to the output port before older cells. Therefore, out-of-sequence arrivals may occur. Either complicated re-sequencing at the output port or an architecture with scheduling schemes that delivers cells in sequence is required. It is intuitive to use more buffers at CMs to store cells coming from different IMs as cells of a flow may get dispersed among CMs. An MMM Clos-network switch with extended memory at CMs, called the (MM^{e}M) switch, is introduced and studied in this section.

The proposed buffered Clos-network packet switch has a similar architecture to the MM^{m}M switch. The MM^{e}M switch also has three stages of switching modules, IMs, CMs, and OMs. In an $N \times N$ switch, there are p IMs, m CMs and p OMs. As in the previous switch, n input ports are connected to one IM, and $p \times n = N$. Each IM is an $n \times m$ crosspoint buffered switch and each OM is an $m \times n$ crosspoint buffered switch. Different from the MM^{m}M switch, each CM in the MM^{e}M switch is constructed using a $p \times p$ crosspoint buffered switch with $p \times N$ buffers. Figure 6.2 shows the $N \times N$ MM^{e}M switch.

The terminology in the previous section is used except for VOMQs at CMs. Different from $VOMQ(i, r, j)$ defined for an MM^{m}M Clos-network switch, in an MM^{e}M switch, $VOMQ(i, r, j, h)$ represents a VOMQ at $CM(r)$ that stores cells from $IM(i)$ destined to $OP(j, h)$.

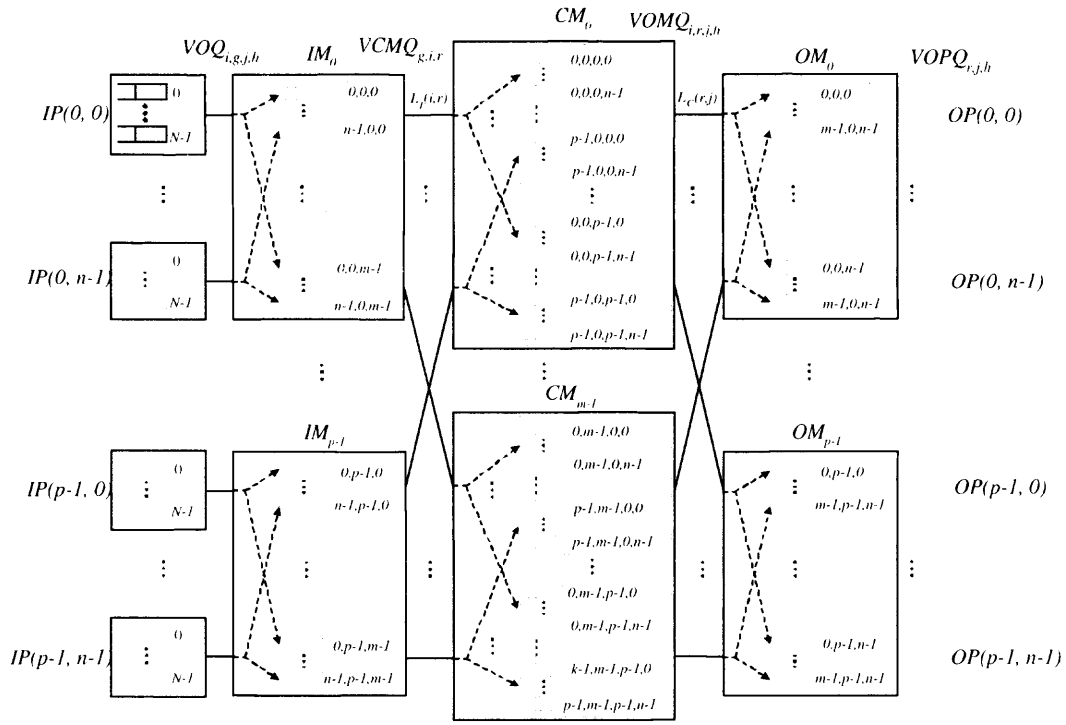

Figure 6.2 $N \times N$ MM^{e}M Clos-network packet switch.

Input modules: These modules are equivalent to those used in the MM^{m}M switch.

Central modules: A CM has $p \times N$ buffers each of which stores cells from $IM(i)$ to $OP(j, h)$. There are p schedulers at each CM, one per input link. Each CM is designed to have $m \times k \times n$ VOMQs, each of which stores cells from $L_I(i, r)$ of $IM(i)$ to $OP(j, h)$. The scheduler at the output link of the CM selects an eligible VOMQ to forward cells in a round-robin fashion. Virtual queues per output port in the CM are used to separate flows from different IMs to a specific output port.

Output modules: These modules are equivalent to those used in the MM^{m}M switch.

The MM^{e}M switch works similarly to an MM^{m}M switch. The difference is that the arbiter at output links of IMs considers the destination port of a cell at IM to forward cells to available VOMQ at CM, and the arbiter at output links of CMs selects a cell out of N possible candidates.

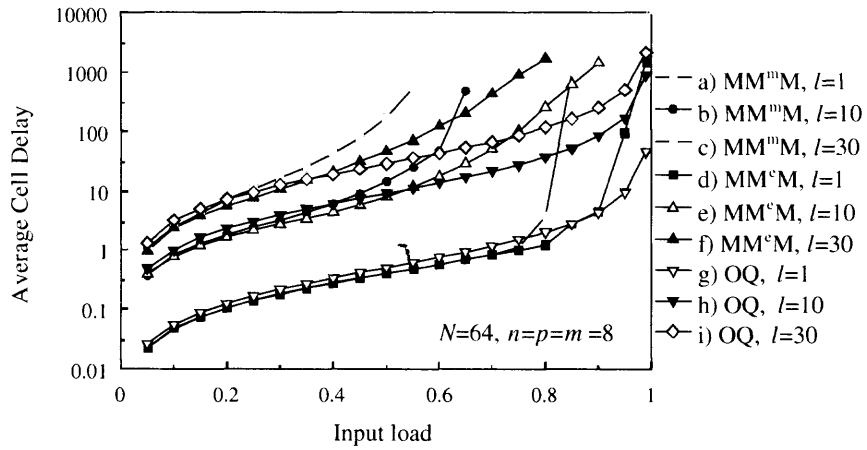

Figure 6.3 Average cell delay of 64×64 MM^{m}M, MM^{e}M, and OQ switches.

6.4 Performance Analysis

This section compares the performance of the proposed MM^{e}M Clos-network switch with MM^{m}M Clos-network switch under uniform and nonuniform traffic models. The traffic models considered are uniform, unbalanced, and diagonal, as defined in previous chapters.

6.4.1 Uniform Traffic

The round-robin selection scheme is used at the input port, IM, CM and OM schedulers for MM^{m}M and MM^{e}M switches under uniform traffic. Figure 6.3 shows the average cell delay of 64×64 MM^{m}M, MM^{e}M, and output-queued (OQ) switches. Curves a), b) and c) show that the MM^{m}M switch delivers limited throughput under round-robin selection. The throughput decreases when the burst length increases. Curves d), e) and f) show that the MM^{e}M switch has higher throughput than that of the MM^{m}M switch. The performance of MM^{e}M switch is comparable to the OQ switch when the load is less than 0.9 and with Bernoulli arrivals.

Figure 6.4 and figure 6.5 show the average cell delay of 64×64 MM^{m}M and MM^{e}M switch under uniform traffic when $l = 1$ with larger number of queue size in all modules respectively. As it is shown in figure 6.5, MM^{e}M switch can achieve 100% throughput when $k_1 = k_2 = k_3 = 8$.

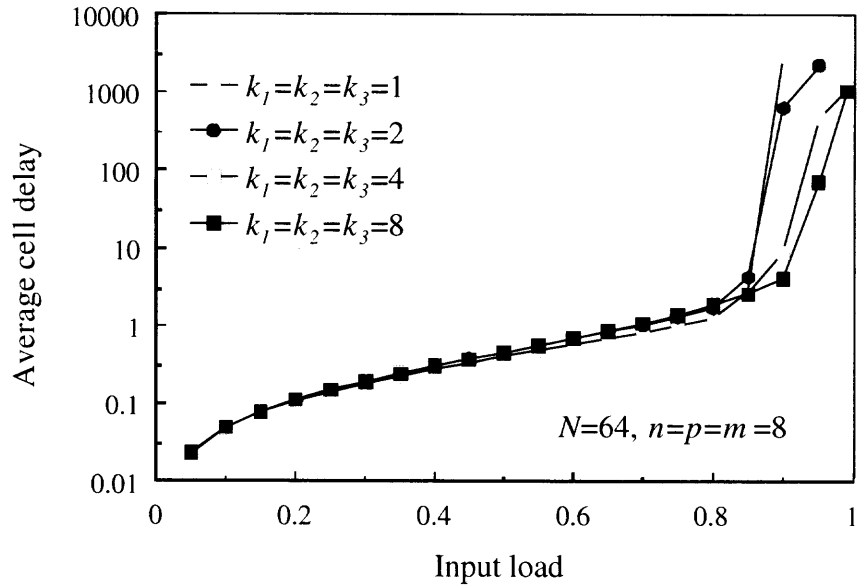

Figure 6.4 Average cell delay of 64 × 64 MM^{m}M switch under uniform traffic with different crosspoint buffer sizes.

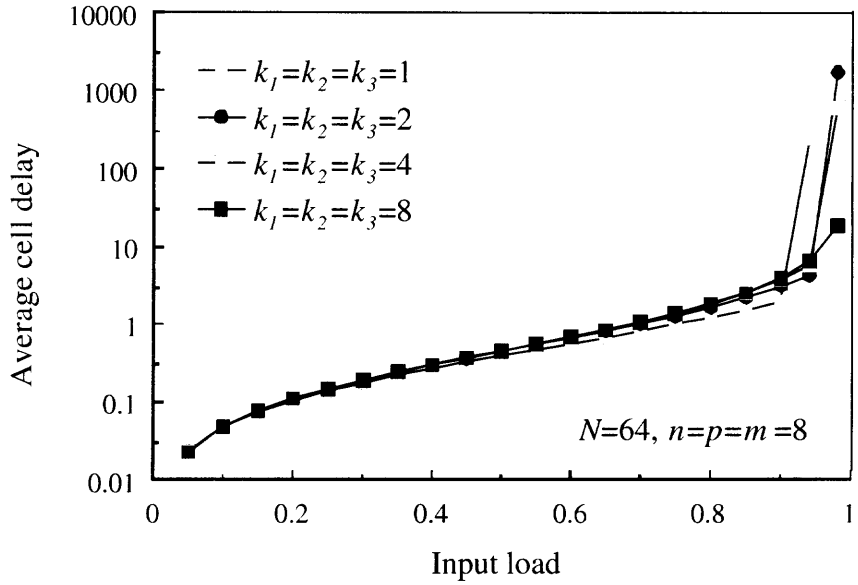

Figure 6.5 Average cell delay of 64 × 64 MM^{e}M switch under uniform traffic with different crosspoint buffer sizes.

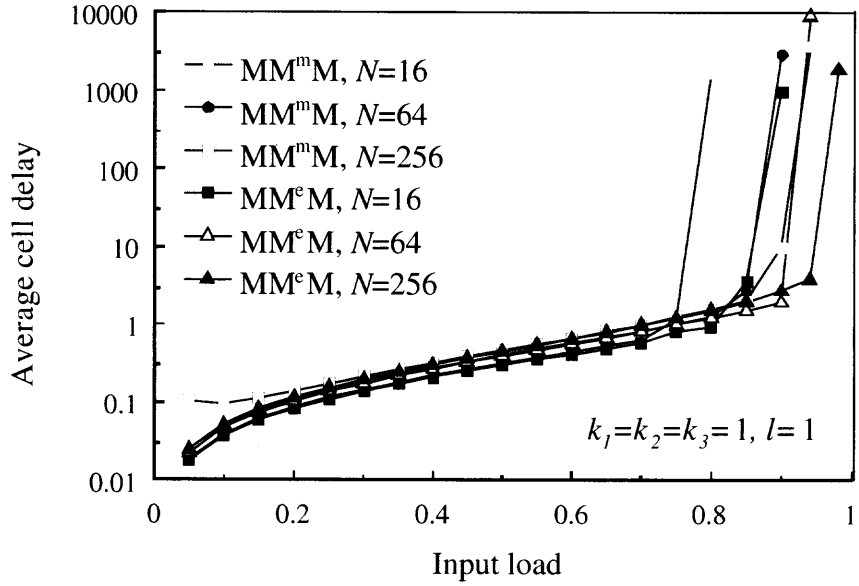

Figure 6.6 Average cell delay of MM^{m}M and MM^{e}M switches under uniform traffic with different switch sizes.

6.4.2 Nonuniform Traffic

The MM^{m}M and MM^{e}M switches were tested under unbalanced and diagonal traffic. The longest queue first (LQF) selection is used at each input port and round-robin selection for the rest of the arbiters in the different modules to study the best performance that can be achieved with both switches under nonuniform traffic and the minimum crosspoint-buffer size, one cell, in all modules.

6.4.2.1 Unbalanced Traffic. Figure 6.7 shows the throughput performance of 64×64 MM^{m}M and MM^{e}M switches under unbalanced traffic. The MM^{e}M switch achieves higher throughput than the MM^{m}M switch. Figure 6.8 shows the throughput performance of MM^{m}M and MM^{e}M switches with different switch sizes under unbalanced traffic. It is shown that as switch size increases, the throughput increases for both switches. In general, the MM^{e}M switch shows higher throughput than the MM^{m}M switch.

6.4.2.2 Diagonal Traffic. Figure 6.9 compares the throughput of 64×64 MM^{m}M and MM^{e}M switches under diagonal traffic. It is clear to see that the throughput of the MM^{e}M

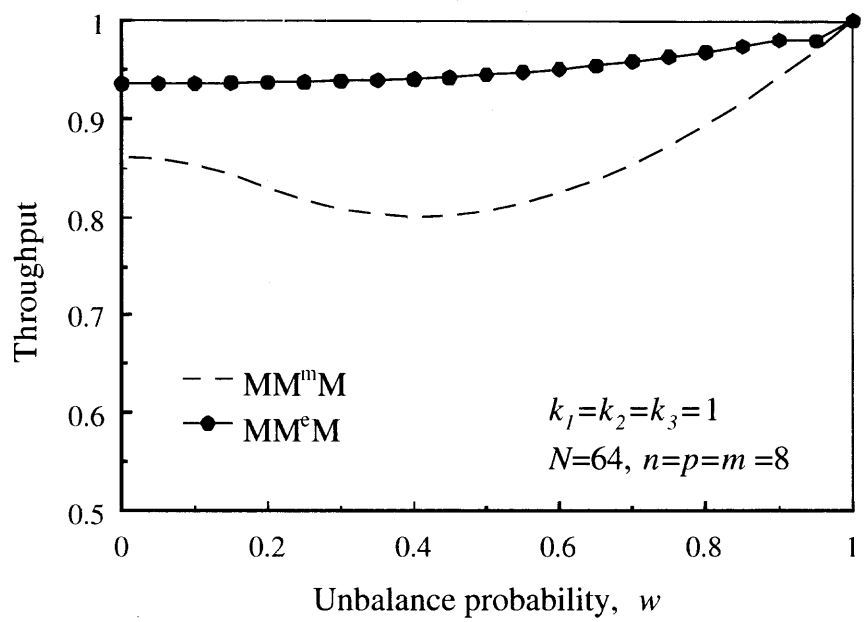

Figure 6.7 Throughput of 64×64 MM^{m}M and MM^{e}M switches under unbalanced traffic.

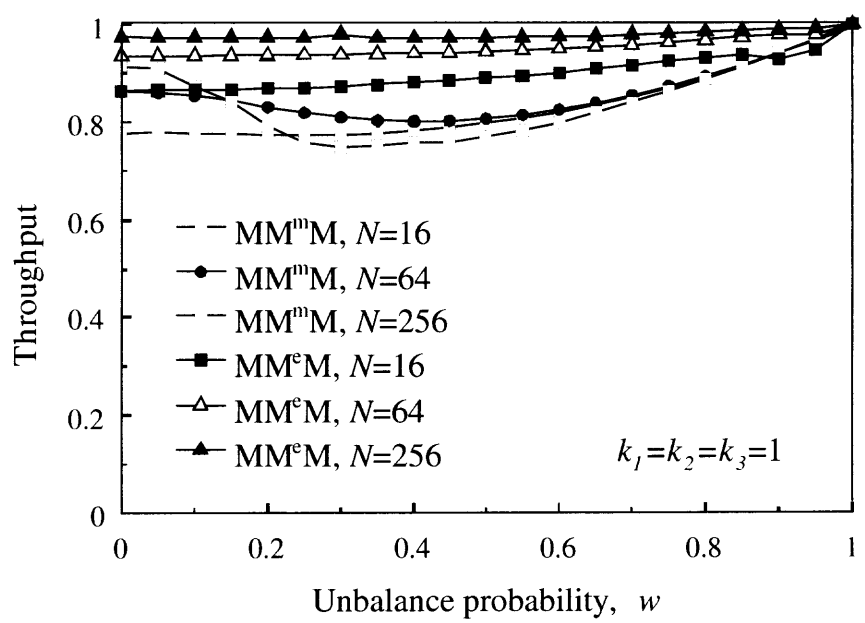

Figure 6.8 Throughput of MM^{m}M and MM^{e}M switches under unbalanced traffic with different switch sizes.

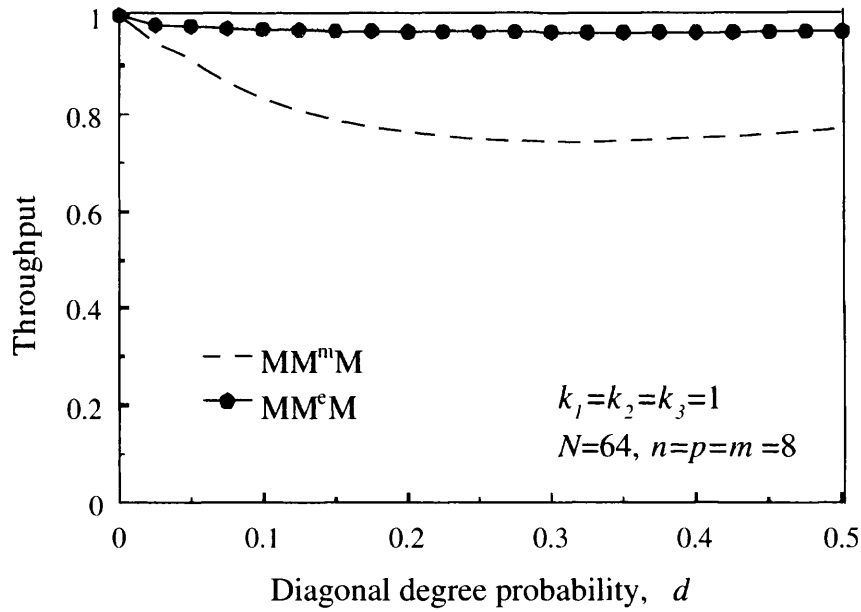

Figure 6.9 Throughput of 64×64 MM^{m}M and MM^{e}M switches under diagonal traffic.

switch is close to 99%, which is much higher than that of the MM^{m}M switch. As the switch size N increases, the throughput of both switches increases, as shown in Figure 6.10.

6.5 MMM Clos-network Switch with Service of Cells in Sequence

It is shown that an MM^{e}M switch has higher throughput performance than an MM^{m}M switch in section 6.4. To keep the transmission of cells in sequence in an MMM switch, several queueing and arbitration mechanisms are proposed in this section. Figure 6.11 shows the architecture of an MMM switch that keeps the service of cells in sequence. This switch is referred to as MCS in the remainder of this chapter. In this switch, there are $n \times p \times m$ queues in each IM, CM and OM. There are $n \times m$ time stamp registers at each IM and $n \times m$ time stamp registers at each OM. The queues in all the modules are kept to a minimum size of one cell.

The following additional notations are used in the description of this switch.

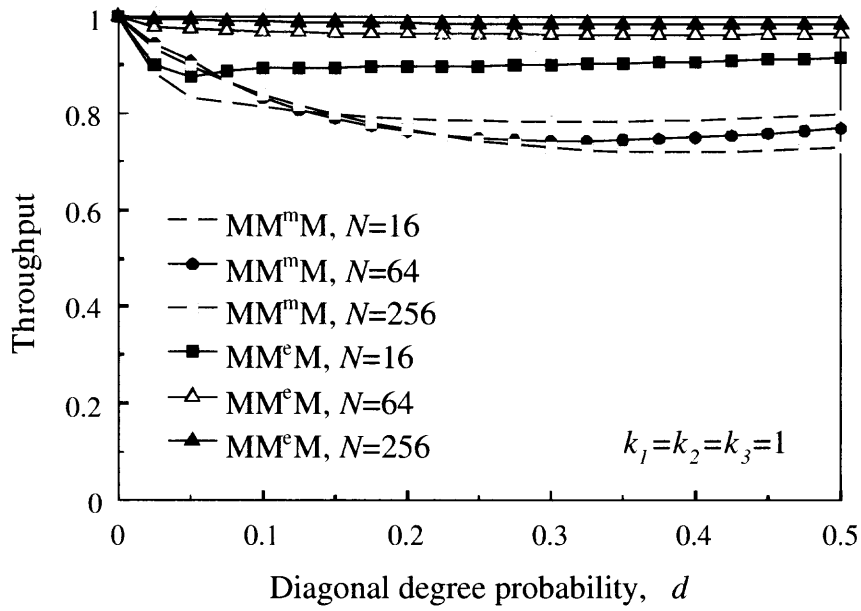

Figure 6.10 Throughput of MM^{m}M and MM^{e}M switches under diagonal traffic with different switch sizes.

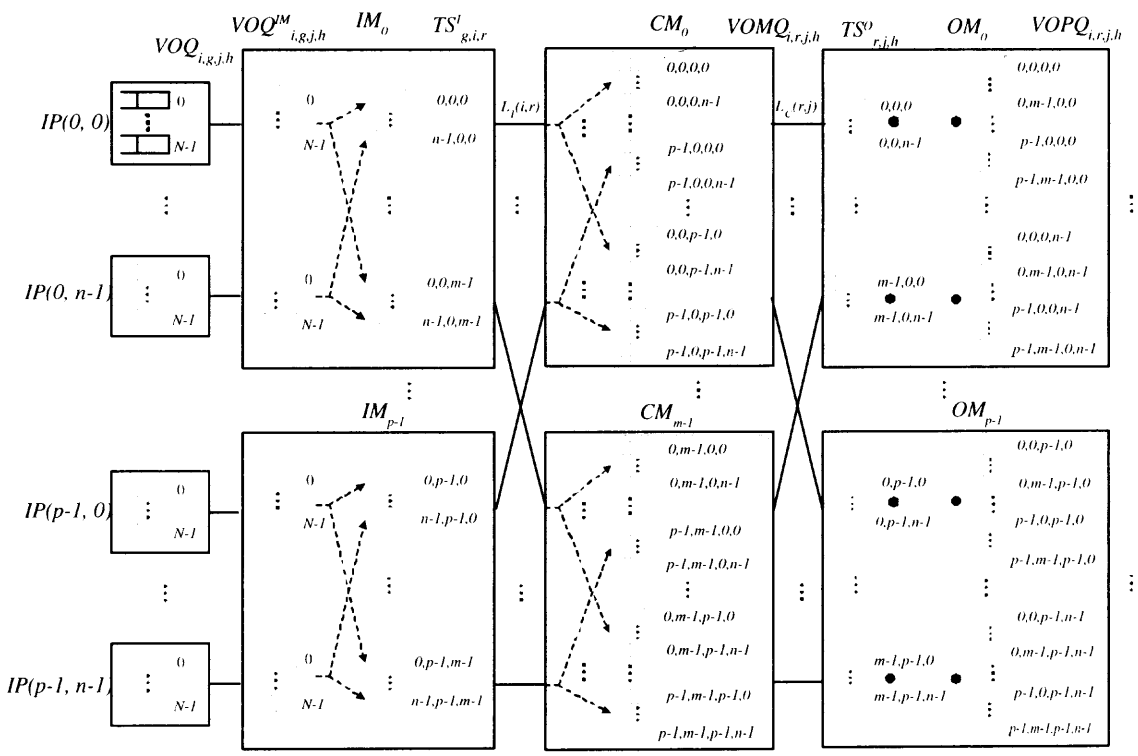

Figure 6.11 An $N \times N$ packet switch that serves cell in sequence, or MCS.

Input Module

- $VOM^{IM}(i, g, j, h)$: queue at $IM(i)$ that holds cells from $IP(i, g)$ destined to $OP(j, h)$. This is equivalent to VOQs at input ports, but with buffers with a size of one cell.
- $TS^{I}(i, g, r)$: register at egress link of $IM(i)$ connected to $CM(r)$ that holds one of the nth oldest time stamps of a HOL cell in $VOM^{IM}(i, g, j, h)$. There are $n \times N$ VOM^{IM}s and $n \times m$ TS^{I}s in each IM.

Central Module

- $VOMQ(i, r, j, h)$: queue at $CM(r)$ that holds cells from $IM(i)$ to $OP(j, h)$. There are $n \times p^2$ queues in each CM.
- $A^{O}_{CM}(r, j)$: arbiter at egress link of $CM(r)$ that connects to $OM(j)$. This arbiter selects the HOL cell of VOMQs from different input modules with the oldest time stamp (or in a first-come first-serve scheduling) for each output port. Since there are n output ports connected to an OM, $A^{O}_{CM}(r, j)$ sends a set of n time stamps to $OM(j)$, one per output port for $OM(j)$. The time stamps are stored in the time stamp registers at $OM(j)$ and participate in the matching process at OMs to select cells for forwarding from CMs to the queues in OMs.

Output Module

- $VOPQ(i, r, j, h)$: queue at $OM(j)$ that stores cells from $IM(i)$, routed through $L_c(r, j)$ and destined to $OP(j, h)$.
- $TS^{O}(r, j, h)$: time stamp register at ingress links of OM. There are $m \times n$ time stamp registers for each OM, each of which stores a time stamp sent from $L_c(r, j)$.
- $A^{I}_{CM-OM}(r, j)$: arbiter at ingress link of $OM(j)$ connected to $CM(r)$ that sends requests with weight of value of $TS^{O}(r, j, h)$ to egress link arbiters at $OM(j)$.
- $A^{E}_{CM-OM}(j, h)$: arbiter at egress link of $OM(j)$ connected to $OP(j, h)$ that grants the requests from $A^{I}_{CM-OM}(r, j)$ with the oldest time stamp.
- $A_{OP}(j, h)$: arbiters at egress link of $OM(j)$ connected to $OP(j, h)$ that selects a cell with the oldest time stamp to be forwarded to $OP(j, h)$.

Matching Process in OMs

Queues at OM are separated by CM and IM to avoid blocking and to allow receiving cells in sequence. The matching process in OMs allows several CMs send cells from the same of different flows to (the queues in) an output port, or an m:1 match per output, where m is the number of CMs (note that a match per output in an IB switch can be denoted as a 1:1 match). This matching process is different from that used in IB switches. Here, a match can be performed from several input arbiters to an output arbiter at the same time slot under conditions to ensure that cells are transmitted in sequence. The following OM match scheme describes intrinsically these conditions.

Step 1: OM ingress-link arbiter $A^I_{CM-OM}(r, j)$ sends a request to output arbiter $A^O_{CM-OM}(j, h)$ where $TS^O(r, j, h)$ has a time stamp stored.

Step 2: $A^O_{CM-OM}(j, h)$ selects the request that has the oldest time stamp and sends the grant to that ingress-link arbiter when $VOPQ(i, r, j, h)$ is available.

Step 3: $A^I_{CM-OM}(r, j)$ selects the request with oldest time stamp and accepts the grant from output arbiter $A^O_{CM-OM}(j, h)$.

Step 4: The following process is to be executed in the following n iterations based on the acceptance of a grant from OM egress link arbiters to find the most possible match while keeping cells from same IM in sequence.

- Arbiters of an OM's egress that received an acceptance in the previous iteration grant another request from:

  - same IM, i, if the time stamp of the request is equal to the time stamp of the cell previously accepted the grant plus one, $TS^O_{t+1}(r', j, h) = TS^O_t(r, j, h) + 1$, $r' \neq r$, where $t + 1$ is the iteration after iteration t;

  - different IM, i, if the time stamp of the request is equal to the time stamp of the cell previously accepted the grant, $TS^O_{t+1}(r', j, h) = TS^O_t(r, j, h)$, $r' \neq r$.

- OM 's egress link arbiters who didn't receive an acceptance in the previous iteration grant another request from different r if the time stamp of the request is equal to that of the request in the first iteration, $TS^O_{t+1} = TS^O_{t=1}$.

If an OM egress-link arbiter $A^E_{CM-OM}(j,h)$ gets its grant accepted, it continues granting requests with the next oldest time stamp from same input module or requests with same time stamp as previous accepted request from another input module until a possible t^{th} request. If $A^E_{CM-OM}(j,h)$ does not get its grant accepted, it grants requests from different CMs with the same time stamp as the time stamp of the request it previously granted to and so on until a possible t^{th} request.

The switch works as follows: When a cell enters an IM, it is stored in VOQ^{IM} of the set corresponding to the ingress link that stores the cells for $OP(j,h)$. A time-based (e.g., $r = (g+T)\ \%\ m$, where T is the time stamp of the current time slot) load balancing stage determines $TS^I(i,g,r)$, where the time stamp of a cell (obtained at entering the IM) is stored. This is equivalent to selecting the CM where the cell will be routed to. One of the m arbiters, $A^E_{IM-CM}(i,r)$, at each egress link of IM selects the time stamp of the oldest cell. The cell at the HOL of the selected queue is sent to $CM(r)$ at the next time slot and stored in the queue connected to $OM(j)$. Since VOM^{IM} is used and the size of queue is one cell, no cell can be forwarded to IM from the same flow until the HOL cell is served. This process ensures cells from the same flow are kept in sequence when going through the IMs and CMs.

$A^I_{CM-OM}(r,j)$ at CMs selects the oldest time stamps of cells that are destined to the same output port and send the time stamps to the registers at $OM(j)$. The cells wait in queues for dispatching and are forwarded once they get a grant signal from OMs.

As described above, $OM(j)$ holds $m \times n$ time stamp registers, $TS^O(j,n,r)$, with m TS^Os per OM ingress link. Each set has an arbiter, $A^E_{CM-OM}(r,j)$. At the egress links of OM, there are $n \times m \times p$ OM queues, denoted as $VOPQ(i,r,j,h)$, with $m \times p$ queues per egress link (or $OP(j,h)$). Each output has an arbiter, $A_{OP}(j,h)$, that selects a non-empty VOPQ that has a cell with oldest time stamp each time slot (e.g., oldest-cell first selection). OM arbiters, A^I_{CM-OM} and A^E_{CM-OM}, perform matching to select the oldest cells for each output, using TS^O values. The matching result is sent to $CM(r)$, which

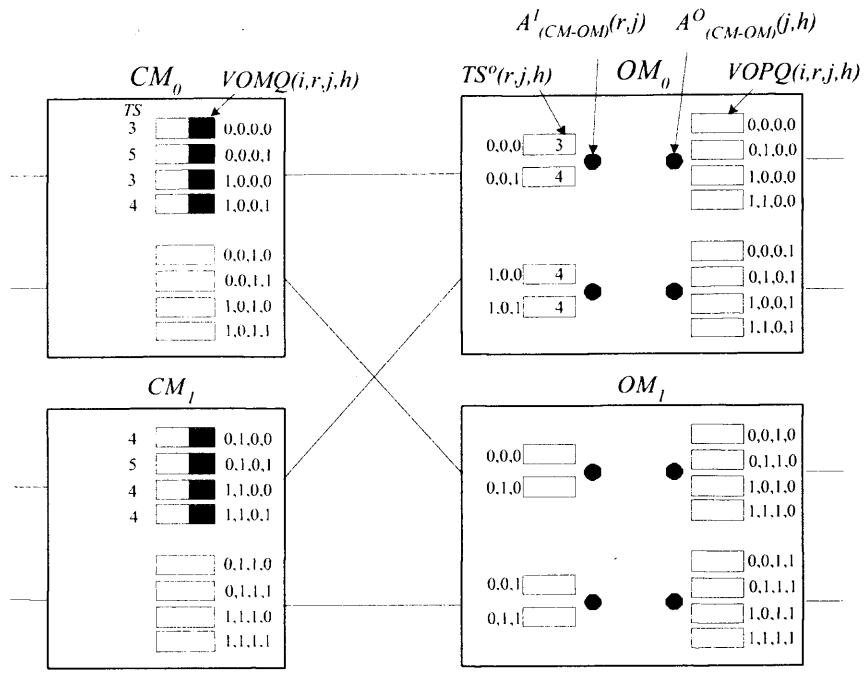

Figure 6.12 Example of matching between CM and OM of a 4×4 MCS.

works as a grant signal. CMs send the selected cells to OMs in the next time slot and cells are stored in VOPQs. A_{OP} selects a cell and forwards it to the output in the next time slot. The matching process that keep cells in sequence in the OMs as explained previously is shown with an example of a 4×4 MCS.

Figure 6.12 shows an example of the matching process at OMs of a 4×4 MCS with $n = m = p = 2$ where the match is used to indicate which cells at CM are forwarded to OM with the purpose of keeping cells in sequence.

In this example, it is considered that only those VOMQs with brown squares have cells waiting for dispatching. The number next to the cell under TS represents the time stamp of the cell. In this case, $VOMQ(0,0,0,0)$, $VOMQ(0,0,0,1)$, $VOMQ(1,0,0,0)$, and $VOMQ(1,0,0,1)$ have cells with time stamps three, five, three, and four, respectively. $A^I_{CM-OM}(0,0)$ selects cells with oldest (i.e., the smallest) time stamps that are destined to output ports zero and one, which are three for cell at $VOMQ(1,0,0,0)$ and four for cell at $VOMQ(1,0,0,1)$. The time stamps are then sent and stored in $TS^O(0,0,0)$ and

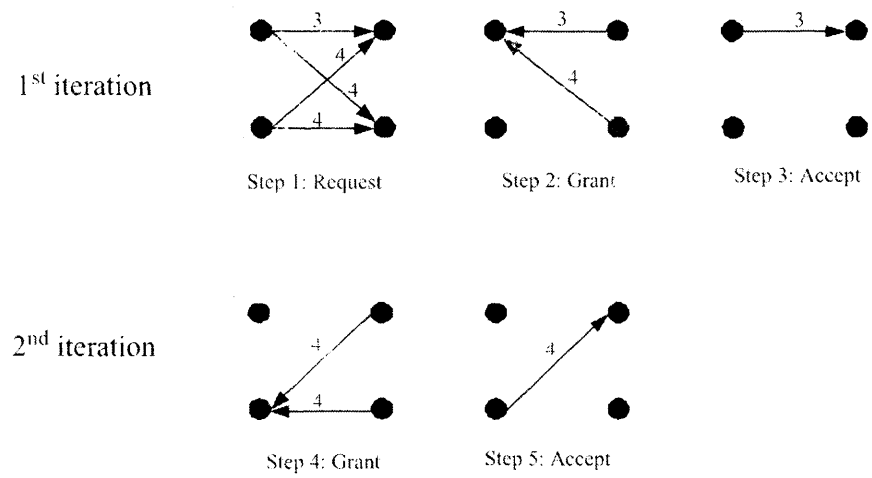

Figure 6.13 Example of matching process at OM.

$TS^O(0,0,1)$, respectively. Similar process is done in $CM(1)$. Figure 6.13 shows the matching process performed in $OM(0)$.

Step 1: $A^E_{CM-OM}(0,0)$ sends request to output zero and one based on the requests from $TS^O(0,0,0)$ with $i = 1$ and $TS^O(0,0,1)$ with $i = 1$, respectively. Similarly, $A^I_{CM-OM}(1,0)$ sends request to outputs 0 and 1 based on the requests from $TS^O(1,0,0)$ with $i = 0$ and $TS^O(1,0,1)$ with $i = 1$.

Step 2: $A^O_{CM-OM}(0,0)$ has two requests with value three and four. The arbiter selects the request with oldest time stamp, which is three, and gives grant to $A^I_{CM-OM}(0,0)$. ($A^O_{CM-OM}(0,1)$ gives grant to $A^I_{CM-OM}(0,0)$.)

Step 3: $A^I_{CM-OM}(0,0)$ accepts the grant from $A^I_{CM-OM}(0,0)$ with a smaller time stamp, three.

Step 4: In this example, $A^O_{CM-OM}(0,0)$ has its grant accepted and the time stamp of the cell is three. Therefore, it can continue giving grant to the request with time stamp four from $i = 1$ or request with time stamp three from $i \neq 1$. Here, $A^I_{CM-OM}(1,0)$ has request with time stamp four from $i = 1$. Therefore, the grant from $A^O_{CM-OM}(0,0)$ goes to $A^I_{CM-OM}(1,0)$. $A^O_{CM-OM}(0,1)$ was not matched in the first iteration. Therefore, it grants a request from a different $A^I_{CM-OM}(r,j)$ with the same time stamp as the previous request, which is four. Here, $A^I_{CM-OM}(1,0)$ has request with time stamp four. $A^O_{CM-OM}(0,1)$

gives grant to $A^I_{CM-OM}(1,0)$. $A^I_{CM-OM}(1,0)$ receives two grants with same time stamps and accepts a grant randomly, in this case, $A^O_{CM-OM}(0,0)$. Two cells are matched to be forwarded to OMs in two iterations. By doing so, cells destined to the same output port are served in sequence regardless of which CM the cells go through. The proof of in-sequence service is presented in Subsection 6.5.1.

6.5.1 Providing Service of Cells in Sequence with MCS

Here, a cell is identified as $C_x(i,g,j,h,t_T)$, where the set (i,g,j,h) indicates that the cell belongs to a flow going from $IP(i,g)$ to $OP(j,h)$, and t_T is the time stamp (or arrival order label) assigned to the cell at the time it arrives at IM. This time stamp determines the serving order for this cell. Here, the discussion is based on the oldest-cell first selection scheme, where for any two time stamps t_a and t_b, the smaller of the time stamps is considered to be the older. For simplicity and without losing generality, it is assumed that all IMs have the same reference clock such that cells arriving at the same time to different CMs get assigned the same time stamp.

Any two cells $C_a(v_1,w_1,x_1,y_1,t_\alpha)$ and $C_b(v_2,w_2,x_2,y_2,t_\beta)$ belong to the same flow if $v_1 = v_2$, $w_1 = w_2$, $x_1 = x_2$, and $y_1 = y_2$.

This section uses the following definition of service of cells in sequence: Consider any two cells $C_1(i,g,j,h,t_1)$ and $C_2(i,g,j,h,t_2)$ such that $t_1 < t_2$, it is said that cells C_1 and C_2 are served in sequence if cell $C_1(i,g,j,h,t_1)$ departs no later than $C_2(i,g,j,h,t_2)$ from any queuing system or output port of the switch.

To show that cells are served in sequence in the proposed switch, the following theorems are used to organize this proof.

Theorem 6.1. *Cells are served in sequence from input ports and IMs when the oldest-cell first selection is used at both input ports and output link arbiters of IMs, and destined CMs are selected arbitrarily.*

Proof. Consider the following conditions:

1. The size of each VOQ^{IM} is one cell.
2. A cell is assigned to an IM egress link in an arbitrary fashion. Here, we consider a time-based load-balancing scheme as an example.
3. The n oldest time stamps at the VOQ^{IM} of $IP(i, g)$ are stored in the time stamp registers TS^I at each output link of IM, distributed arbitrarily.
4. An output link arbiter uses the oldest-cell first selection policy from all time stamps from an input port.

The proof of Theorem 1 is partitioned into the following two lemmas.

Lemma 6.1. *A cell $C_1(i, g, j, h, t_1)$ always arrives at $VOQ^{IM}(i, g, j, h)$ before $C_2(i, g, j, h, t_2)$ for $t_1 < t_2$.*

Proof. Cells belonging to the same flow are stored in the same VOQs, which uses first-in first-out (FIFO) service, therefore these cells are sent to IM in sequence.

On the other hand, because the size of VOQ^{IM}s is one cell, no two cells from the same flow are at an IM at any time. □

Lemma 6.2. *A cell $C_1(i, g, j, h, t_1)$ always departs from $VOQ^{IM}(i, g, j, h)$ before $C_2(i, g, j, h, t$ if $t_1 < t_2$.*

Proof. Because the size of VOQ^{IM}s is one cell and a cell from an input can only go to one VOQ^{IM} (per its destination), only one cell from each flow can be placed in an IM at any time. Therefore, C_2 can be sent to its IM only if C_1 receives service.

□

Since Lemmas 6.1 and 6.2 are true. Theorem 6.1 is also true. □

Theorem 6.2. *Cells from the same flow arrive in sequence to OM.*

Consider the following conditions:

i) Each VOMQ and VOPQ have a buffer size of one cell.

ii) Each $TS^O(r, j, h)$ holds the n oldest time stamps of the VOMQs ($VOMQ(i, r, j, h)$) at CM's egress link.

iii) OM's ingress link arbiters perform the OM matching process with OM's egress link arbiters, using up to n iterations.

Because cells belonging to a flow may be forwarded to the same or different CMs, the following lemmas discuss the order in which they arrive at their destined OM. Theorem 6.2 proves cells arrive at OMs (or depart from CMs) in sequence.

Lemma 6.3. *In MCS, any two cells of a flow that traverse a single CM arrive in sequence to their destined OM.*

Proof. The proof of this lemma is similar to that of Lemma 6.2.

Because VOMQs have a size of one cell and cells arrive in sequence to a CM, there is only one cell C_1 from a flow at CM and that cell is older than any other cell of that flow at IM. Therefore, C_1 has to be sent to an OM to let another cell from the same flow to arrive at the CM.

□

Lemma 6.4. *In MCS, any two cells C_1 and C_2 from a flow that are stored in different CMs arrive at OM in sequence.*

Proof. Assume cells $C_1(i, g, j, h, t_1)$ and $C_2(i, g, j, h, t_2)$ are from the same flow $f(i, g, j, h)$ and cells $C_a(i', g', j, h, t_1)$ $C_b(i', g', j, h, t_2)$ are from another flow $f(i', g', j, h)$, where $t_1 < t_2$. Since the OM matching process allows the transmission of several cells from different CMs to the VOPQs, two flows are considered in the following discussion.

Consider that any two cells from the same flow are at different CMs: C_1 is at $CM(1)$ and C_2 is at $CM(2)$. Because of the presence of another flow, the following two scenarios are considered:

- **Case 1.** Cells C_a is at $CM(1)$ and C_b is at $CM(2)$.

This is, the time stamp of either C_1 or C_a is in TS^O for $CM(1)$ and the time stamp of either C_2 or C_b is in TS^O for $CM(2)$.

In the first iteration of the OM matching process, the oldest cell is selected. This is, either C_1 or C_a is selected.

In the second iteration of the OM matching process, a time stamp t_x where $t_x > t_1$ is selected and the only one is t_2, then C_2 or C_b can be selected. Because the OM matching process in a matched port selects a younger time stamp only from the same flow after the first, then C_2 is selected if C_1 was selected in the first iteration, and C_b is selected only if C_a was selected in the first iteration.

- **Case 2.** Cells C_a is at $CM(2)$ and C_b is at $CM(1)$.

  This case has the same result in the first iteration. Furthermore, C_a has an older time stamp and this is equal to C_1, then the OM matching scheme selects C_a, and C_2 remains in $CM(2)$.

In both cases, cells are not sent out of sequence.

Note that if flows are destined to different output ports, the matching process is performed separately by different arbiters and different output ports. Therefore, the two flows do not interfere with each other in the order cells are transmitted as decided by the OM matching process.

□

Lemma 6.3 and Lemma 6.4 are proven true. Therefore, Theorem 6.2 is proven true.

Theorem 6.3. *Any two cells from a flow stored at the destined OM are served in sequence.*

Proof. Considering the following:

Theorem 6.2 proves that any two cells from the same flow arrive at $OM(j)$ in sequence.

VOPQs have a size of one cell, then any two cells from the same flow will be at the head of line.

Output arbiter $A_{OP}(j, h)$ selects the oldest cell of the head of line of VOPQs to be forwarded to $OP(j, h)$, then the arbiter selects the older cell of a flow to leave the switch.

□

Theorem 6.1, 6.2 and 6.3 are proven true. Therefore, cells are served in sequence with the proposed MCS.

6.6 Conclusions

Single-stage crosspoint buffered switches can provide high throughput performance. However, single-stage crosspoint buffered switches have limited scalability. To support switches with larger number of ports, different types of Clos-network switches have been introduced. This chapter studies the performance of MMM Clos-network switches with minimum number of buffers (MM^{m}M) using round-robin and LQF arbitration schemes at the input port and round-robin arbitration scheme at the arbiters in the switch modules. A new architecture with extended memory in central modules of and MMM Clos-network switch (MM^{e}M) is proposed to study the effect on throughput performance when using more buffers in central modules. The proposed MM^{e}M switch achieves higher throughput than an MM^{m}M switch under both uniform and nonuniform traffic.

To address the out-of-sequence problem with an MMM Clos-network switch, a new switch architecture is proposed. This new switch uses memory in all modules, and uses oldest-cell first (or first-come first-serve) as the selection scheme in all arbiters. MCS services cells in sequence. This switch uses different architectures from buffered crossbars in different modules and a new matching process for x:1 matching at the output ports, where x is a number equal to the number of input ports connected to one IM. As a reference, IB switches use 1:1 matching, where one input can be matched to only one output. It is proven that the MCS services cells in sequence.

CHAPTER 7

CONCLUSIONS

The explosion of Internet traffic requires high speed packet switches to perform fast switching and support multiple service classes. As the switch speed increases, the transmission delay on the connecting cables between line cards and switch fabric is not negligible. In this dissertation, the need of reducing memory amount while supporting long RTT in practical switch implementation is addressed.

CICB packet switches were introduced to relax input-output arbitration timing and to provide high throughput under admissible traffic. However, the amount of memory required in the crossbar of an $N \times N$ switch is $N^2 \times RTT \times k \times L$, where k is crosspoint buffer size, L is the packet size, and RTT is the round-trip time. As the speed of switch increases and large scale switches reside on multiple racks, RTT can be several time slots and is not negligible. The memory amount required makes the implementation costly or infeasible. To reduce the required memory amount, two shared memory combined-input crosspoint-buffered (SMCB) packet switches, SMCB$\times m$ and mSMCB switches, are introduced, where the crosspoint buffers are shared among inputs. SMCB$\times m$ switch uses a memory speedup of m and dynamic memory allocation of the shared memory. mSMCB switch avoids using speedup by arbitrating the access of inputs to the crosspoint buffers. These two switches reduce the required memory of the buffered crossbar by 50% or more and achieve equivalent throughput under i.i.d. traffic with uniform distributions when using random selections. This is proven using queuing analysis. Simulation results are presented to show the performance of the switches under uniform, unbalanced, diagonal and PO2 traffic with both weightless and weighted scheduling schemes, round-robin and LQF selection respectively. It is shown that mSMCB switch has same throughput performance as the SMCB$\times m$ switch, but with the advantage of not using speedup. It is also shown

that when $m = 2$, the switches give optimum performance under admissible traffic and have the simple $2 - N$ matching scheme that is easy to implement. The implementation of mSMCB switch is presented. It is shown that the implementation of mSMCB switch can use a pipelined system approach.

To meet the demand for switches that can support QoS, mSMCB is extended to support differentiated services and long distances between line cards and the buffered crossbar. The architecture of CICB and mSMCB switches are modified to support P different priorities of traffic. The throughput performance of CICB switches and mSMCB switches are compared under uniform, unbalanced and diagonal traffic. It is shown that CICB switches use $N^2 \times k \times L \times P$ amount of memory to avoid blocking of high priority cells. mSMCB switch requires $\frac{1}{mP}$ of the memory amount in a CICB switch and is able to achieve similar throughput performance to a CICB switch with similar priority management while using no speedup in the shared memory.

Multicast traffic is expected to increase in packet-oriented networks by the inclusion of broadcast and multimedia-on-demand services. Crosspoint buffered switch has intrinsic properties that are favorable for switching and replication of multicast packets. Different arbitration schemes is studied for an SMCB switch to support multicast traffic. To improve the throughput performance, an output-based SMCB switch (O-SMCB) is proposed.

As SMCB switches can efficiently support an RTT twice as long as that supported by CICB switches and as the performance of SMCB switches is bounded by a matching between inputs and crosspoint buffers, three load-balanced CICB switches with flexible access to crosspoint buffers, CICB-FA and CICB-SA are proposed to support longer RTTs than SMCB switches and to provide higher throughput under a wide variety of admissible traffic models. The CICB switches with flexible access allow an input to use any available crosspoint buffer at a given output instead of having a dedicated access crosspoint buffer where a crosspoint buffer can be accessed by only one flow. It is shown in the simulation results that the proposed switches achieve similar performance by reducing the required

crosspoint buffer size by a factor of N. The proof of serving cells in sequence using FCFS output arbitration is presented. Queuing analysis is used to prove that the load balanced CICB switches with random selection of the load-balancing stage is weakly stable. It is shown that the proposed switches support RTTs that can be kN-time-slot long, while providing 100% throughput for such high data-rate flows. As a comparison, for a given RTT size, the load-balanced CICB switch requires a minimum $k = \frac{RTT}{N}$ cells while a simple CICB switch requires a minimum $k = RTT$ cells. The proposed switch relaxes the amount of memory to $\frac{1}{N}$ of the amount required by a CICB switch with dedicated access to crosspoint buffers.

Scalability of Crosspoint buffered switches is addressed in this dissertation. Memory-memory-memory (MMM) Clos-network switch is studied with minimum and extended memory amount at central modules. The throughput performance increases under both uniform and non-uniform traffic with more buffers implemented in central modules of an MMM Clos-network switch. To tackle the out-of-sequence problem with the MMM Clos-network switches, a new switch architecture with extended memory in all modules and scheduling schemes is introduced. The proposed switch is proven to serve cells in-sequence.

REFERENCES

[1] R. Rojas-Cessa, E. Oki, Z. Jing, and H. J. Chao, "CIXB-1: Combined input-one-cell-crosspoint buffered switch," *Proc. IEEE Workshop on High Performance Switching and Routing*, May 2001, pp. 324–329.

[2] A. K. Kloth, Ed., *Advanced Router Architectures*, 1st ed. Taylor and Francis, 2006.

[3] P. Newman, T. Lyon, and G. Minshall, "Flow labelled IP: A connectionless approach to ATM," *Proc. IEEE INFOCOM'96*, vol. 3.

[4] H. J. Chao, C. H. Lam, and E. Oki, Eds., *Broadband Packet Switching Technologies: A Practical Guide to ATM Switches and IP Routers*, 1st ed. John Wiley and Sons, Inc., 2006.

[5] "White paper: ATM switching architecture." FORE systems, November 1993.

[6] R. Rojas-Cessa and E. Oki, "Round-robin selection with adaptable-size frame in a combined input-crosspoint buffered switch," *IEEE Commun. Letters*, vol. 7, no. 11, November 2003, pp. 555–557.

[7] Y. S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The knockout switch: a simple, modular architecture for high-performance switching," *IEEE J. Select. Area Commun.*, vol. 5, no. 8, October 1987.

[8] S. Nojima, E. Tsutsui, H. Fukuda, and M. Hashimoto, "Integrated packet network using Bus matrix," *IEEE J. Select. Areas Commun.*, vol. SAC-5, no. 8, 1987, pp. 1284–1291.

[9] F. A. Tobagi, T. K. Kwok, and F. M. Chiussi, "Architecture, performance and implemenation of the tandem banyan fast packet switch," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, October 1991, pp. 1173–1193.

[10] J. N. Giacopelli, J. J. Hickey, W. S. Marcus, W. D. Sincoskie, and M. Littlewood, "Sunshine: a high-performance self-routing broadband packet switch architecture," *IEEE J. Select. Areas Commun.*, vol. 9, no. 8, October 1991, pp. 1289–1298.

[11] F. M. Chiussi, J. G. Kneuer, and V. P. Kumar, "Low-cost scalable switching solutions for broadband networking: the ATLANTA architecture and chipset," *IEEE Commun. Mag.*, December 1997, pp. 44–53.

[12] M. Karol and M. Hluchyj, "Queuing in high-performance packet-switching," *IEEE J. Select. Area Commun.*, vol. 6, December 1988, pp. 1587–1597.

[13] A. Bianco, M. Franceschinis, S. Ghisolfi, A. Hill, E. Leonardi, F. Neri, and R. Webb, "Frame-based matching algorithms for input-queued switches," *Proc. IEEE Workshop on High Performance Switching and Routing*, 2002, pp. 69–76.

[14] L. Mhamdi and M. Hamdi, "Practical scheduling algorithms for high-performance packet switches," *Proc. IEEE ICC'03*, vol. 3, 2003, pp. 1659–1663.

[15] E. Oki, N. Yamanaka, Y. Ohtomo, K. Okazaki, and R. Kawano, "A 10-gb/s (1.25 gb/s x8) 4 x 0.25-μm cmos/simox atm switch based on scalable distributed arbitration," *IEEE J. Solid-State Circuits*, vol. 34, no. 12, December 1999, pp. 1921–1934.

[16] P. Giaccone, B. Prabhakar, and D. Shah, "Randomized scheduling algorithms for high-aggregate bandwidth switches," *IEEE J. Select. Area Commun.*, vol. 21, no. 4, May 2003, pp. 546–559.

[17] S. Motoyama, D. W. Petr, and V. S. Frost, "Input-queued switch based on a scheduling algorithm," *Electronics Letters*, vol. 31, no. 14, July 1995, pp. 1127–1128.

[18] M. Karol and M. Hluchyj, "Queuing in high-performance packet-switching," *IEEE J. Select. Area Commun.*, vol. 6, December 1988, pp. 1587–1597.

[19] D. Shah, "Maximal matching scheduling is good enough," *Proc. IEEE GLOBECOM'03*, vol. 6, October 2003, pp. 3009–3013.

[20] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Trans. Commun.*, vol. 6, no. 8, August 1999, pp. 1260–1267.

[21] Y. Doi and N. Yamanaka, "A high-speed ATM switch with input and cross-point buffers," *IEICE Trans. Commun.*, vol. E76, no. 3, March 1993, pp. 310–314.

[22] K. C. K. Yoshigoe, "A parallel-polled virtual output queue with a buffered crossbar," *Proc. IEEE Workshop on High Performance Switching and Routing*, May 2001, pp. 271–275.

[23] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, and N. Chrysos, "Variable packet size buffered crossbar (CICQ) switches," *Proc. IEEE ICC)*, vol. 2, June 2004, pp. 1090–1096.

[24] T. Javadi, R. Magill, and T. Hrabik, "A high-throughput algorithm for buffered crossbar switch fabric," *Proc. IEEE Workshop on High Performance Switching and Routing*, May 2001, pp. 324–329.

[25] L. Mhamdi and M. Hamdi, "MCBF: a high-performance scheduling algorithm for buffered crossbar switches," *IEEE Commun. Letters*, vol. 7, no. 9, 2003, pp. 451–453.

[26] J. Chao, "Next generation routers," *Proc. the IEEE*, vol. 90, no. 9, September 2002, pp. 1518–1558.

[27] F. Abel, C. Minkenberg, R. P. Luijten, M. Gusat, and I. Iliadis, "A four-terabit packet switch supporting long round-trip times," *IEEE Micro*, vol. 23, no. 1, January-February 2003, pp. 10–24.

[28] R. Rojas-Cessa, Z. Dong, and S. G. Ziavras, "Load-balanced-CICB switch and long round-trip time support," *Proc. IEEE Globecom '05*, vol. 2, 2005, pp. 1002–1006.

[29] Z. Dong and R. Rojas-Cessa, "Long round-trip time support with shared-memory cross-point buffered packet switch," *Proc. IEEE Hot Interconnects, 2005. 13th Symposium on*, 2005, pp. 138–143.

[30] R. Luijten, C. Minkenberg, and M. Gusat, "Reducing memory size in buffered crossbars with large internal flow control latency," *Proc. IEEE Globecom '03*, vol. 7, December 2003, pp. 3683–3687.

[31] G. Shrimali, I. Keslassy, and N. McKeown, "Designing packet buffers with statistical guarantees," *Proc. High Performance Interconnects '04*, August 2004, pp. 54–60.

[32] F. Abel, C. Minkenberg, R. P. Luijten, M. Gusat, and I. Iliadis, "A four-terabit single-stage packet switch with large round-trip time support," *Proc. IEEE Workshop on High Performance Switching and Routing*, August 2002, pp. 5–14, 21–23.

[33] N. McKeown, "The *i*SLIP scheduling algorithm for input-queue switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, 1999, pp. 188–201.

[34] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," *Proc. IEEE INFOCOM 2000*, vol. 2, March 2000, pp. 556–564.

[35] D. Gross and C. M. Harris, Eds., *Fundamentals of Queueing Theory*, 3rd ed. John Wiley and Sons, Inc., 1998.

[36] N. C. Hock, Ed., *Queueing Modelling Fundamentals*, 1st ed. John Wiley and Sons, Inc., 1996.

[37] T. Anderson, S. Owicki, J. Saxe, , and C. Tacker, "High-speed switch scheduling for local area networks," *ACM Trans. on Computer Systems*, vol. 11, no. 4, November 1993, pp. 319–352.

[38] G. Nong, J. K. Muppala, and M. Hamdi, "Analysis of nonblocking ATM switches with multiple input queues," *IEEE/ACM Trans. on Networking*, vol. 7, no. 1, February 1999, pp. 60–74.

[39] J. Yang, J. Ye, and S. Papavassiliou, "A flexible and distributed architecture for adaptive end-to-end QoS provisioning in next generation networks," *IEEE J. Select. Area Commun.*, vol. 23, no. 2, February 2005, pp. 321–333.

[40] Z. Qin, R. Rojas-Cessa, and N. Ansari, "OSPF-based adaptive and flexible security-enhanced QoS provisioning," *Proc. IEEE Sarnoff Symposium, 2006*, March 2006.

[41] A. Minagar and S. Safavi, "The optimized prioritized *i*SLIP scheduling algorithm for input-queued switches with ability to support multiple priority levels," *Proc. 10th International Conference on Telecommunications, 2003*, vol. 2, no. 23, 2003, pp. 1680–1685.

[42] T. Lee, "Non-blocking copy networks for multicast packet switching," *IEEE J. Select. Areas Commun.*, vol. 6, December 1998, pp. 1455–1467.

[43] A. Bianco, P. Giaccone, C. Piglione, and S. Sessa, "Practical algorithms for multicast support in input queued switches," *Proc. IEEE Workshop on High Performance Switching and Routing*, June 2006.

[44] M. Hamdi and M. Lhamdi, "Scheduling multicast traffic in internally buffered crossbar switches," *Proc. IEEE ICC'04*, vol. 2, 2004, pp. 1103–1107.

[45] L. Mhamdi and S. Vassiliadis, "Integrating uni- and multicast scheduling in buffered crossbar switches," *Proc. IEEE Workshop on High Performance Switching and Routing*, June 2006.

[46] Z. Dong and R. Rojas-Cessa, "Long round-trip time support with shared-memory crosspoint buffered packet switch," *Proc. IEEE Hot Interconnects*, August 2005, pp. 138–143.

[47] ——, "Shared-memory combined input-crosspoint buffered switch for differentiated services," *Proc. IEEE Globecom'06*, November 2006.

[48] C. S. Chang, D. S. Lee, and Y. S. Jou, "Load balanced Birkhoff-von Neumann switches," *Proc. IEEE Workshop on High Performance Switching and Routing*, May 2001, pp. 276–280.

[49] M. Nabeshima, "Performance evaluation of a combined input- and crosspoint-queued switch," *IEICE Trans. Commun.*, vol. E83-B, no. 3, March 1993, pp. 737–741.

[50] I. Keslassy and N. Mckeown, "Maintaining pakcet order in two stage switches," *Proc. IEEE INFOCOM'02*, vol. 2, June 2002, pp. 23–27.

[51] R. Rojas-Cessa, Z. Dong, and Z. Guo, "Load-balanced combined input-crosspoint buffered packet switch and long round-trip times," *IEEE Commun. Letters*, vol. 4, no. 7, July 2005, pp. 661 – 663.

[52] I. Elhanany and M. Hamdi, Eds., *High Performance Switch Architectures*, 1st ed. Springer, September 2006.

[53] C. Clos, "A study of non-blocking switching networks," *Bell Systems Technical Journal*, 1953, pp. 406–424.

[54] X. Li, Z. Zhou, and M. Hamdi, "Space-memory-memory architecture for Clos-network packet switches," *Proc. IEEE ICC'05*, vol. 2, May 2005, pp. 1031–1035.

[55] E. Oki, Z. Jing, R. Rojas-Cessa, and H. J. Chao, "Concurrent round-robin-based dispatching schemes for Clos-network switches," *IEEE/ACM Trans. on Networking*, vol. 10, no. 6, December 2002, pp. 830–844.

[56] K. Pun and M. Hamdi, "Static round-robin dispatching schemes for Clos-network switches," *Proc. IEEE Workshop on High Performance Switching and Routing*, May 2002, pp. 239–243.

[57] J. Chao, J. Park, S. Artan, S. Jiang, and G. Zhang, "Trueway: a highly scalable multi-plane multi-stage buffered packet switch," *Proc. IEEE Workshop on High Performance Switching and Routing*, May 2005, pp. 246–253.

[58] N. Chrysos and M. Katevenis, "Scheduling in non-blocking buffered three-stage switching fabric," in *Proc. IEEE INFOCOM'06*, April 2006, pp. 23–29.