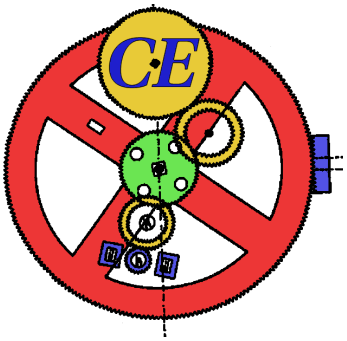

Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2011

MSc THESIS

Scheduling in Partially Buffered Crossbar Switches

Di Cao

CE-MS-2011-06

Abstract

Intensive studies have been conducted to identify the most suitable architecture for high-performance packet switches. These architectures can be classified by queuing schemes, scheduling algorithms and switching fabric structures. The crossbar based switching fabric has been widely agreed to be the most suitable one, for its low cost, scalability and native multicast support. Large numbers of commercial implementations and literature studies have been conducted on the unbuffered crossbar switching architecture. Due to the requirement of the centralized scheduler, scheduling algorithms in the unbuffered crossbar have generally high complexities. This leads to time-consuming scheduling processes that prevent the unbuffered architecture from scaling up with the modern optical link operating at the Gb/S range. The buffered crossbar architecture has been proposed to overcome the scheduling complexity bottleneck faced by the unbuffered crossbar. The introduction of cross point buffers decouples the centralized scheduling process and lowers the scheduling complexity. However, the drawback of the buffered crossbar lies in the fact that it requires N^2 expensive on-chip memories, N being the size of the switch, limiting the scalability of the buffered crossbar architecture. To provide the scheduling simplicity brought by the buffered crossbar while having a cost close to the unbuffered one, the partially buffered crossbar architecture has been proposed. With the combination of advantages of the previous two architectures, the Partially Buffered Crossbar (PBC) is deemed as one of the competitive candidates for next-generation switching architectures. However, the previously proposed algorithms did not fully exploit its potential. In this thesis, we: *i*) propose a unicast scheduling algorithm that further pushes the performance of the PBC switch under various non-uniform traffic settings, while using as few as 2 internal buffers per output. *ii*) study the multicast traffic support by the partially buffered crossbar switch and come up with an effective multicast scheduling algorithm.

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science

Scheduling in Partially Buffered Crossbar Switches

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Di Cao
born in Beijing, China

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Scheduling in Partially Buffered Crossbar Switches

by Di Cao

Abstract

Intensive studies have been conducted to identify the most suitable architecture for high-performance packet switches. These architectures can be classified by queuing schemes, scheduling algorithms and switching fabric structures. The crossbar based switching fabric has been widely agreed to be the most suitable one, for its low cost, scalability and native multicast support. Large numbers of commercial implementations and literature studies have been conducted on the unbuffered crossbar switching architecture. Due to the requirement of the centralized scheduler, scheduling algorithms in the unbuffered crossbar have generally high complexities. This leads to time-consuming scheduling processes that prevent the unbuffered architecture from scaling up with the modern optical link operating at the Gb/S range. The buffered crossbar architecture has been proposed to overcome the scheduling complexity bottleneck faced by the unbuffered crossbar. The introduction of cross point buffers decouples the centralized scheduling process and lowers the scheduling complexity. However, the drawback of the buffered crossbar lies in the fact that it requires N^2 expensive on-chip memories, N being the size of the switch, limiting the scalability of the buffered crossbar architecture. To provide the scheduling simplicity brought by the buffered crossbar while having a cost close to the unbuffered one, the partially buffered crossbar architecture has been proposed. With the combination of advantages of the previous two architectures, the Partially Buffered Crossbar (PBC) is deemed as one of the competitive candidates for next-generation switching architectures. However, the previously proposed algorithms did not fully exploit its potential. In this thesis, we: *i*) propose a unicast scheduling algorithm that further pushes the performance of the PBC switch under various non-uniform traffic settings, while using as few as 2 internal buffers per output. *ii*) study the multicast traffic support by the partially buffered crossbar switch and come up with an effective multicast scheduling algorithm.

Laboratory : Computer Engineering
Codenumber : CE-MS-2011-06

Committee Members :

Advisor:	Dr. Lotfi Mhamdi, CE, TU Delft
Chairperson:	Dr. ir. Koen L. M. Bertels, CE, TU Delft
Member:	Dr. Zaid Al-Ars, CE, TU Delft
Member:	Dr. Fernando A. Kuipers, NAS, TU Delft

Dedicated to my parents, my friends and my love.

Contents

List of Figures

Acknowledgements

Two-year study at Delft University of Technology has turned out to be a very challenging, educating and most importantly happy experience, forming an unique and precious memory that will be permanently stored in my L1 cache. I owe many people many thanks. Firstly, I would thank my parents for their love and encouragement of all time. Secondly, I would thank my supervisor, as well as my friend, Dr. Lotfi Mhamdi, who gave me the chance to study the field of high performance switching and scheduling as my master project. He guided me throughout the whole project with his experience and excellence, encouraged me when confronted by hardship, and gave me valuable opinions when it came to the design of a scheduling algorithm and the composition of my final thesis. At last but never the least, I would dedicate my thank to all my friends, specially to Yingchao Wu, Imran Ashraf, and Ning Xie, for all their encouragement and support that powered me with confidence, along my way towards success.

Di Cao
Delft, The Netherlands
May 24, 2011

1 Introduction

This chapter provides a minimal background and outlines the motivation of this thesis.

1.1 Overview

The Internet has proven to be the most universal, scalable data communication network ever. It supports various applications ranging from pure data transmission to real-time video conferencing. Comparing to telephony networks that use circuit-switching technology [1], the Internet employs packet-switching [2] technology and statistical multiplexing [3] to improve the efficiency of link usage between two nodes. A link is shared among a number of users and the cost of transmission is cheaper than that of the circuit-switched networks. However, the share of links leads to a serious problem known as network contention. The policies that solve contention are known as scheduling algorithms, forming one of the most important research topics in the field of networking. The advances of scheduling algorithms contribute largely to the prosperousness of the Internet today.

The Internet is growing rapidly. From its early days in the 1960s, the number of hosts gets doubled every 15 months, and the network traffic load gets doubled every 12 months [4]. The dense wavelength division multiplexing (DWDM) technology available in mid 90s further powers the Internet transmission link capacity with almost unlimited bandwidth by allowing different colors of light traveling in the same link. As link speed reaches the order of Gigabit per Second (Gb/S), the limiting factor of the Internet performance has shifted to the switching capacity of the routers. Since the Internet is structured hierarchically (shown in Fig. 1.1), the traffic density aggregates from the end users along the path to the core routers, demanding the core routers to have immense

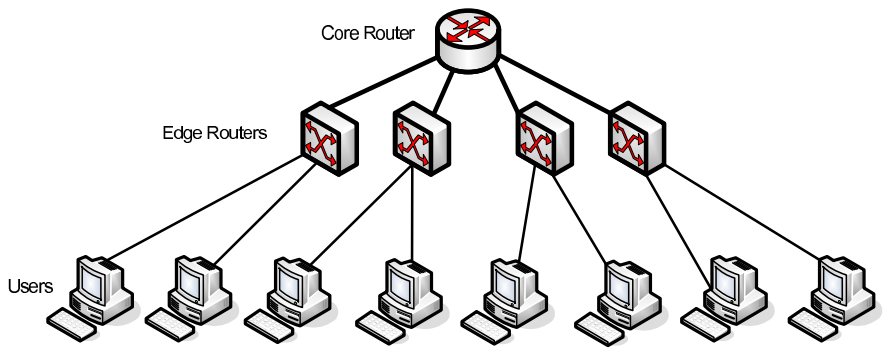

Figure 1.1: The hierarchical structure of the Internet.

switching capacity. As concluded in [4], the maximum capacity of core routers must grow at the same rate as the growth of Internet traffic. Hence, the switching capacity of core routers has to be improved continuously to meet the demand of constantly increasing network traffic.

For the past two decades, different switching architectures have been proposed and studied [5] [6] [7]. Among them, the crossbar based switching architectures have been widely agreed as the most suitable ones, hence forming the major focus of literature studies [8] [9] [10]. Crossbar switches can be characterized by the memory placement and different queuing strategies. The input-queued (IQ) switches have received intensive studies due to their simple hardware implementation and low requirement on memory access rate. The IQ switches with single FIFO per input require almost no scheduling algorithm, however, suffer from the well-known head-of-line (HoL) blocking issue [11]. The HoL blocking raises when a cell destined to a free port gets blocked by another cell that goes to a congested output, limiting the throughput of IQ-FIFO switches to only 58.6% [11]. To address the HoL issue, researchers proposed the queuing scheme known as the virtual-output-queues (VOQs), where N separate FIFOs are kept at each input port, one per output. While introducing an affordable hardware overhead, the VOQ completely eliminates the HoL blocking issue. With the VOQ queuing scheme, the IQ switches need more complex scheduling policies to resolve packet contention. The maximum-weight-matching (MWM) algorithms, have proven to achieve 100% throughput and are stable under any admissible traffic pattern [12]. While delivering the optimum performance, the MWM algorithms are too complex to be implemented into a hardware circuit and hence possess only theoretical values. By comparing to the MWM algorithms, maximal size matching (MSM) algorithms are much simpler and feasible to be translated into a hardware implementation. Maximal size matching algorithms operate on iterations to increase the number of connections between the switch inputs and outputs. Representative examples of MSM algorithms are the parallel iterative matching (PIM) [13] and the iSLIP [14] algorithm, with corresponding commercial implementations [15] [16]. The disadvantage of MSM algorithms mainly comes from their poor performance dealing with non-uniform traffic patterns. In a nutshell, the scheduling algorithms in the IQ switches are either too complex to be practical or too simple to deliver satisfactory performance. The combined-input-cross-point-queued (CICQ) architecture is then proposed to overcome the bottleneck faced by the IQ switches. Through the introduction of the internal buffers deployed at each fabric cross point, the CICQ switch is enabled with distributed input schedulers (IS) and output schedulers (OS), operating in parallel to speed up the scheduling process. Each input scheduler select one cell from N VOQs and transfer it into the internal cross point buffer, while the output schedulers examine the internal buffers and move the selected cells to the outgoing links. Various scheduling algorithms are proposed for the CICQ switch, such as round-robin based input and output schedulers or weighted schedulers that favor long queues or old cells [17]. In general, the CICQ switch delivers performance beyond that of the IQ switch. Despite the advantages described just now, the CICQ switch has high hardware cost due to the expensive N^2 cross point buffers, where N is the number of ports of the switch, forming a limiting factor to the scalability of the CICQ architecture.

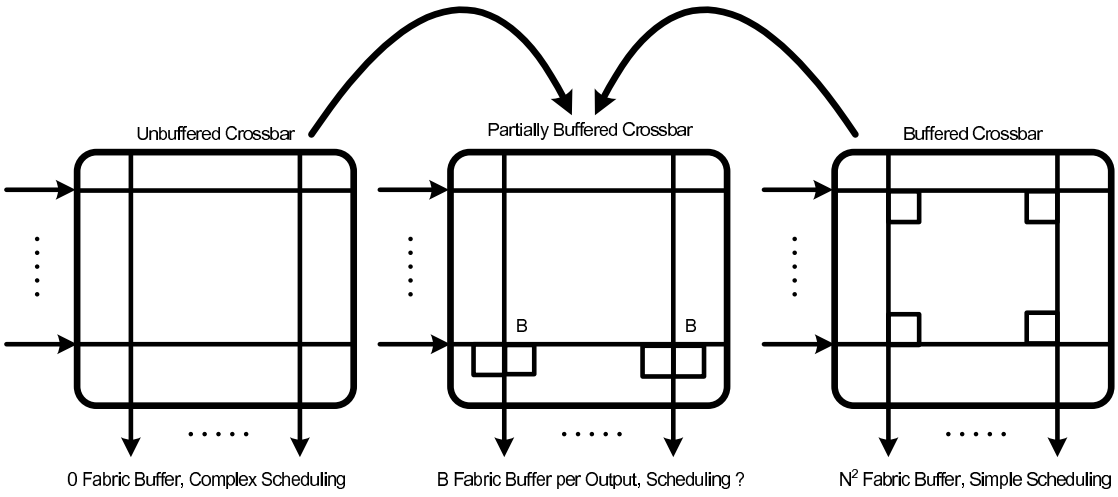

Figure 1.2: The parrtially buffered crossbar makes a good compromise between an unbuffered crossbar and a fully buffered crossbar, resulting in a significantly reduced hardware cost. However, the scheduling algorithm that can take full advantage of the PBC switch has not yet been addressed.

1.2 Motivation and Problem Statement

On one hand, the IQ switch has the simplest hardware design but a poor performance. On the other hand, the CICQ switch delivers qualified performance and scheduling simplicity, but has a poor scalability due to the quadratically increasing requirement of cross point buffers. Hence both architectures are less appealing in terms of candidate to next-generation switching architectures. The partially buffered crossbar (PBC) architecture [18] combines the advantages of both the IQ switch and the CICQ switch, providing satisfactory performance close to the CICQ switch and a relatively cheap hardware cost close to the IQ switch (see Fig. 1.2). Instead of maintaining N^2 buffers at each cross point, the PBC switch keeps a small number, $B(1 \leq B \ll N)$ of shared internal buffers at each fabric output, reducing the number of the internal buffers significantly. On the other hand, these shared internal buffers preserve the feature of distributed schedulers of the CICQ switch. Due to the fact that the number of the internal buffers is less than the number of the input ports, a mapping from N to B has to be carried out. Therefore, the grant schedulers are introduced and in charge of the access to the internal buffers. The scheduling process in the PBC switch is also a combination of that from the IQ switch and the CICQ switch. On one hand, input and output schedulers operate independently and concurrently similar to that of the CICQ switch. On the other hand, the grant decisions are produced using the request-grant-accept (RGA) handshaking that resembles the scheduling of the IQ switch. A class of round-robin based algorithms have been proposed, termed as (distributed-round-robin) DRR, DROP and (prioritized DROP) DROP-PR [18]. All the three algorithms have round-robin based input and grant schedulers, and a credit based mechanism to allow communication between input

schedulers and grant schedulers. Previous study showed that DROP-PR outperforms the rest and its latency converges to the output-queued (OQ) switch under Bernoulli i.i.d arrivals. Yet, using 8 internal buffers per output, it also achieves higher throughput than RR-OCF of the CICQ architecture under non-uniform traffic. However, we argue that RR-OCF is not the best algorithm of the CICQ architecture. Therefore, we seek for an algorithm that is capable of achieving a performance close to the LQF-RR, one of the best algorithms of the CICQ switch, yet using fewer internal buffers. This thesis addresses this issue and proposes an optimal unicast-scheduling algorithm for the PBC architecture.

Alongside the unicast scheduling, multicast support by backbone routers is becoming increasingly important due to the modern development of Internet applications, such as voice-over-IP (VoIP), network television and video conferencing. There have been considerable studies done on multicast support by the IQ and the CICQ architectures and on integration of multicast and unicast schedulers. Yet, little has been said regarding the multicast capability of the PBC switch. In this thesis, we also conduct an experimental study of the multicast capability of the PBC switch, and we compare the PBC switch performance to that of the IQ and the CICQ switch. The results suggest that the PBC architecture is capable of outperforming the IQ and CICQ architectures with a simple round-robin based multicast-scheduling algorithm.

1.3 Thesis Organization and Contributions

The layout of this thesis is as follows: in Chapter 2, we review the existing crossbar based switching architectures, demonstrating their advantages as well as shortcomings. Chapter 3 studies the unicast scheduling of the PBC switches, in which we propose a novel algorithm that delivers satisfactory performance under non-uniform traffic with 2 internal buffers required. Chapter 4 presents an experimental study on the multicast support of the PBC switches, and describes the multicast PBC switching architecture and a round-robin based scheduling algorithm. Finally Chapter 5 concludes this thesis and gives an outlook to some future research topics.

2 Background

The basic, yet important component to a packet-switching communication network is the router. The design of routers has evolved for the past two decades, and different packet-switching architectures have been extensively studied and implemented into commercial products. This chapter introduces the packet-switching paradigm, and describes the existing switching architectures and scheduling algorithms.

2.1 Packet Switching Technology

Communication networks can be classified into two categories: the circuit-switching [1] and the packet-switching [2] networks. The circuit-switching technology is mostly seen in the telephone and telegraph network. When a communication initiated, a dedicated channel (a connection, or a circuit) with a constant bandwidth was established between two end users through the network. Regardless of the actual usage, the channel will be reserved until the communication is finished. This type of technology network is efficient when a constant-rate data is transmitted, for example voice calls and transmission that has its duration of communication longer than that of establishing the circuit.

Within the packet-switching paradigm, data is segmented and traverses the network in the form of small units, termed as packets. Upon arrival at the destination, packets are re-assembled into a meaningful data representation. Fig. 2.1 shows an example of a packet-switching network. As shown in the Figure, in contrast to the circuit-switching network, a link is shared among several users by taking the advantage of unused bandwidth, known as statistical multiplexing [3]. Multiplexing different users on a single link improves the efficiency of link bandwidth utilization, however, gives birth to another

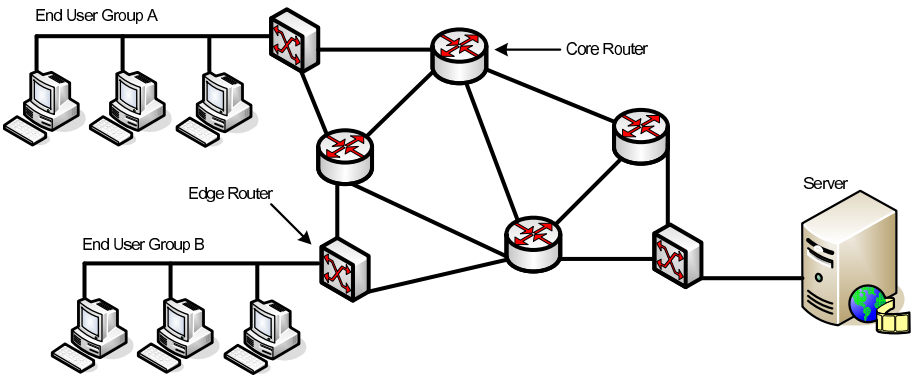

Figure 2.1: An example of packet-switching network.

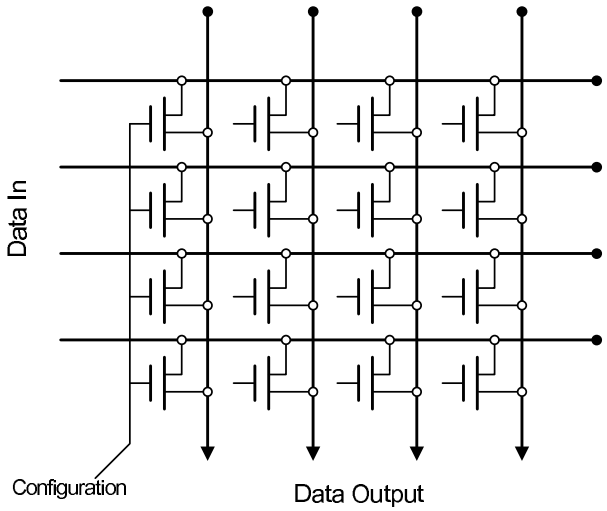

Figure 2.2: A 4x4 crossbar switching fabric.

serious problem, known as *network contention*. Contention rises when multiple packets desire to travel through a same link at the same time, however only one packet can be transmitted. Packets that cannot pass through the link in current time slot have to wait and keep contending in future time slots. Hence the queuing system is required in the packet-switching network, and the existence of the buffering system is one of the key differences between circuit-switching and packet-switching networks.

There are two existing packet-switching technologies, the Asynchronous Transfer Mode (ATM) [19] and the Internet Protocol (IP) [20]. The ATM technology employs fixed-length packets, named as cell. On the other hand, the IP technology adopts packets with variable length. The ATM is also known as a connection-oriented technology, where a virtual circuit is set-up before the actual transmission, therefore quality of service (QoS) is possible to be achieved with ATM technology. The resemblance to a circuit-switching network that requires a setup and teardown overhead makes the ATM technology less efficient in handling short bursts of data. The IP technology, on the other hand, is completely connectionless, making it the dominating packet-switching technology adopted in modern data communication networks. Though variable-length packets are allowed in the IP network, the common implementation of IP routers will only process fixed-length data unit. Throughout this thesis, unless stated otherwise, we use the terms cell and packet to refer to the same entity, namely fixed-size data unit. Variable length packets are segmented into fixed-size units on their entry to the router and re-assembled back to the original packet at when departing from the switch to the outgoing link.

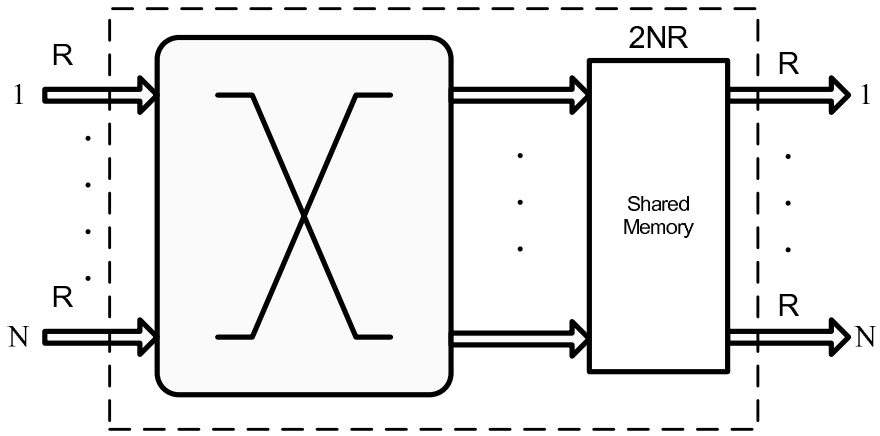

Figure 2.3: An NxN Shared-Memory Switch.

2.2 The Switching Architectures of Routers

The design of switching architectures has evolved over the years, and different architectures have been studied and implemented such as simple bus based switching architectures [21], multi-stage switching [22] architectures and crossbar based architectures. The crossbar based architectures are regarded as the most suitable ones for their scalability, non-blocking property, low implementation cost and simple multicast support. Fig. 2.2 depicts a 4x4 crossbar fabric. Similar to what was shown just now, an NxN crossbar fabric consists of N input and output lines, and N^2 crosspoints. A connection between input i and output j is made simply by closing the crosspoint at (i,j). The placement of switch queueing mechanism is a crucial factor that determines the performance of crossbar switches. In the following sections we will present different crossbar switch designs featured by their memory placement.

2.2.1 The Shared Memory Switch

As shown in Fig. 2.3, the presented switching architecture is known as shared memory switch [6], where one single piece of memory is used to store packets. For an NxN switch, the memory has to be able to store up to N incoming packets and putting up to N packets to the outgoing line each time slot. Assuming a line rate of R, the shared memory switch has to be equipped with the memory having the bandwidth of $2NR$. The shared memory switch has the advantages such as minimized memory for congestion buffering, providing performance guarantees [23]. However, it is difficult to build a high performance router from the shared memory architecture due to the fact that the memory bandwidth requirement cannot be met by the available dynamic random access memory (DRAM) technology.

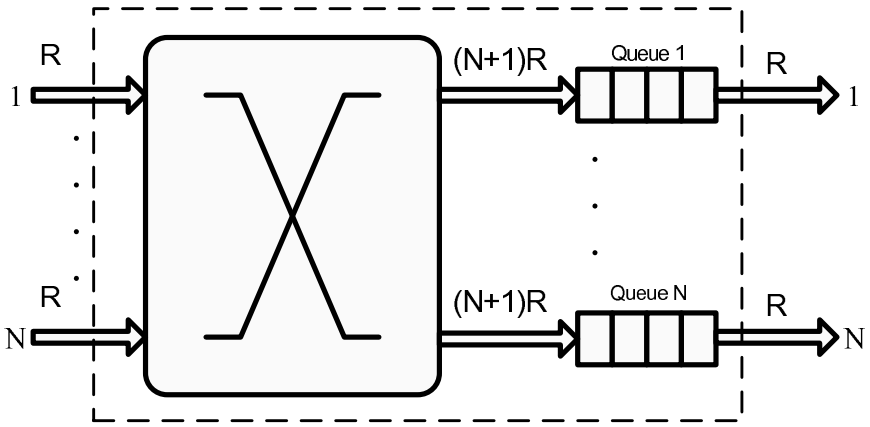

Figure 2.4: An NxN Output Queued Switch.

2.2.2 The Output Queued Switch

The shared memory architecture just discussed above is an example of a class of switches called the output queued (OQ) switches [24]. The OQ switch has been considered as the ideal switching architecture for its unparalleled performance. All the packets arrived at the switch will be immediately switched and placed to a queue located at each fabric output. The output queued switch is also *work conserving*, achieving the highest throughput among all the switching architectures. The switch is said to be *work conserving* if the outputs will never be idle when there are packets destined to them. The Switches in the early time of history were designed using the OQ switching model, and the QoS is also achievable with the OQ architecture [25]. The major disadvantage is that it takes extremely fast memory and high-bandwidth fabric to build the OQ switches. Fig. 2.4 depicts an NxN OQ switch, we can see that there are no queues maintained at the inputs, and total N memories (queues) are deployed after the crossbar fabric, one per output. Within one time slot, at most N packets can arrive at the switch. And in the case where N packets are destined to a same output port, the memory at the destination port has to perform N writes from the inputs and one read to dispatch a packet to the outgoing link. If we assume the link rate is R, then each of the memory should have a bandwidth of $(N+1)R$. While the bandwidth requirement is almost halved comparing to that of the shared memory switch, it is still a function of the switch size, which prohibits the usage of the OQ switch constructing the modern backbone routers. This has, in turn, caused the great interest in the input queued (IQ) switches as discussed in the next section.

2.2.3 The Input Queued Switch

Fig. 2.5 shows the structure of an input queued (IQ) switch with FIFO queuing structure. Different from the OQ switch, the IQ switch buffers the incoming packets at the input line cards. The line card is then connected to the crossbar switching fabric. During each time slot, each input port can send only one cell and each output can receive up to one cell, resulting in a memory access rate of $2R$. The further lowered memory bandwidth

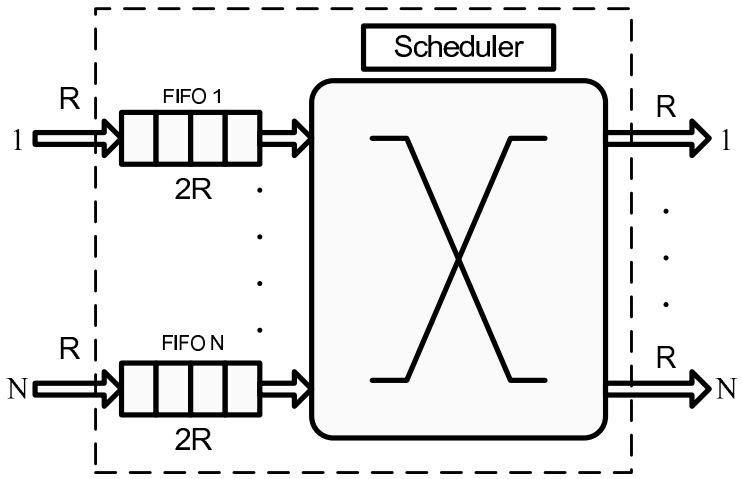

Figure 2.5: An NxN Input Queued Switch with FIFO queues.

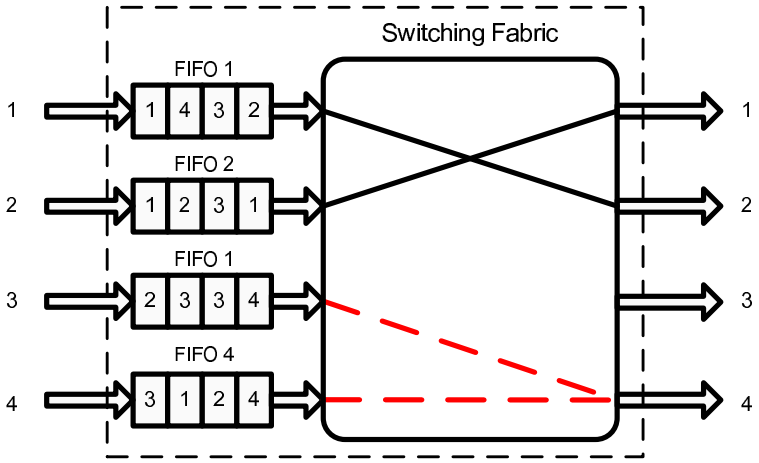

Figure 2.6: Head-of-Line Blocking Issue.

requirement enables the construction of switch, using the IQ architecture to support higher line rate and larger number of ports.

The type of queuing structure in the IQ switch will greatly affect its performance. The simple FIFO queuing architecture, as shown in Fig. 2.5, causes the head-of-line (HoL) blocking issue, limiting the throughput of a switch to only 58.6% [11]. For example, a 4x4 IQ-FIFO switch is shown in Fig. 2.6. Input 3 and input 4 are contending for output 4 (shown in red doted line). If the scheduler decides to make the connection between input 4 and output 4, then input 3 will be blocked since only one cell can be transmitted to one output in one time. However, the cell behind the HoL cell of input 3 is destined to the free output 3, which will remain idle because of the HoL blocking. Through this example, we see that the HoL reduce the potential throughput of this 4x4 switch from

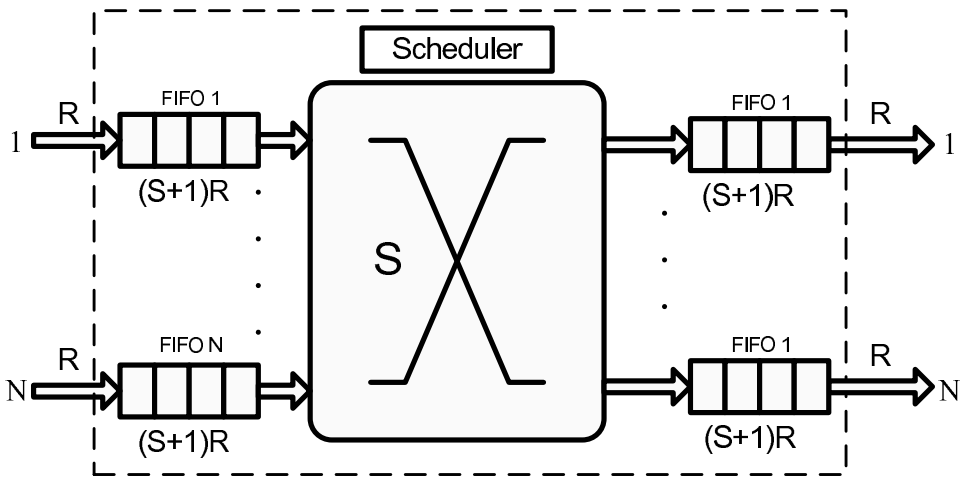

Figure 2.7: An NxN Combined Input Output Queued Switch with Speedup of S.

100% to 75%. The HoL issue can be completely eliminated by employing a queuing architecture known as virtual-output-queues (VOQs) [26]. Another way to overcome the HoL problem is to employ the CIOQ switching architecture with speedup, as will be discussed in the next section.

2.2.4 The CIOQ switch

A switch with a speedup of S can remove up to S packets from each input and deliver up to S packets to each output within a time slot, where a time slot is the time between packet arrivals at input ports. Therefore, for the OQ switch, the S = N, and hence there is no need to buffer packets at input port. For the IQ switch, the S is equal to 1, requiring buffers only at the input port. For a switch with the value of S between 1 and N, packet buffers are required before switching as well as after switching. This architecture is known as combined input output queued (CIOQ) switch, as depicted in Fig. 2.7. The CIOQ switch with a moderate speedup ($1 < S \ll N$) has received extensive studies. Common conclusions indicate that with a speedup between 4 and 5, the CIOQ switch with single FIFO queue at each input can achieve 99% throughput under Bernoulli independent identically distributed (i.i.d) packet arrivals. It is first questioned in [27] and proved, that the CIOQ can be designed to behave identically to the OQ switch with an internal speedup of 4. Later, in [28], it is proven that a speedup of $(2 - 1/N)$ is sufficient to exactly emulate the OQ switch. As appealing as it sounds, the cost of OQ emulation was the use of a more complex scheduling policy called Critical Cell First (CCF) [28]. This algorithm requires a push-in queuing structure (PIFO) along with an insertion policy called Last In Highest Priority (LIHP). An attempt to reduce the complexity of this algorithm was based on Delay Till Critical (DTC) strategy, to reduce the number of iterations from N^2 to N, along with an algorithm called Group-By-Virtual-Output Queue (GBVOQ), to reduce the information complexity. Unfortunately, these two solutions cannot be combined, since they are mutually exclusive. Therefore, these

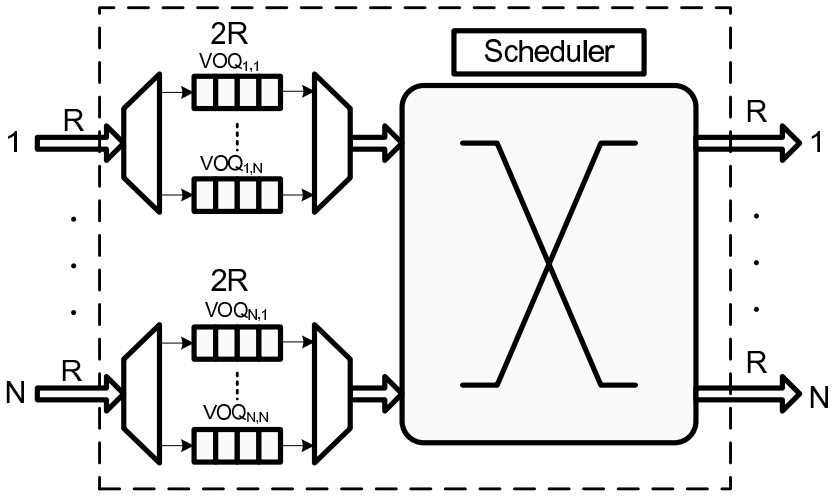

Figure 2.8: An NxN IQ switch with VOQ queueing structure.

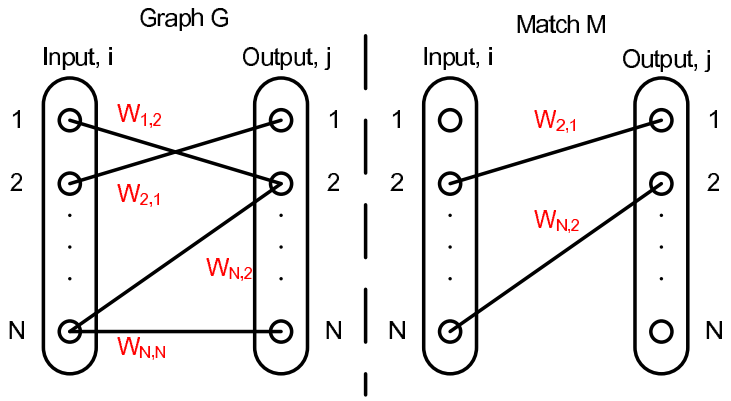

Figure 2.9: An example of bipartite matching.

results remained of theoretical nature.

2.2.5 IQ Switch with VOQ

Another simple way to avoid HoL blocking problem is to adopt the queuing scheme known as the virtual-output-queue (VOQ), the IQ switch with VOQ is shown in Fig. 2.8. Instead of keeping one FIFO, each input now keeps N separate FIFOs, one per output. The HoL is completely eliminated because a cell will no longer be held by another cell destined to a different output port. With the VOQ and an appropriate scheduling algorithm, the IQ switch can achieve 100% throughput [12].

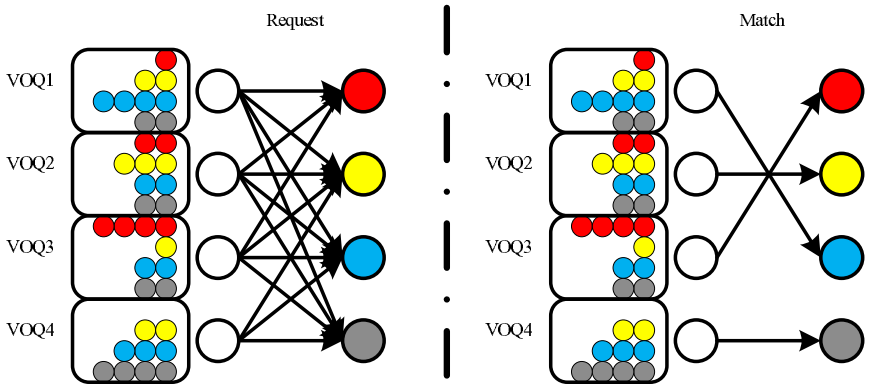

Figure 2.10: An example of the longest-queue-first (LQF) scheduling algorithm.

2.2.5.1 Scheduling in VOQ-IQ switches

Scheduling in the IQ switch is modeled as the bipartite matching problem. As shown in Fig. 2.9, the scheduling algorithms have to solve a bipartite graph to find a conflict-free matching M, due to the input and output contention. If the input i contains any cell destined to the output j, an edge between the input i and the output j will be drawn on the graph, denoted as $w_{i,j}$. The value of $w_{i,j}$ can be either an integer value to represent the occupancy of the corresponding VOQ or simply a binary value to represent whether the queue is empty. And the matching M is said to be conflict-free, if all remaining edges in the graph, have no common vertices. After the scheduling process, the decisions can be expressed by a binary service matrix S, in which $S_{i,j} = 1$ denotes the $VOQ_{i,j}$ will be served, otherwise $S_{i,j} = 0$.

Extensive research has been done to identify algorithms that lead to high performance of the IQ switch. Generally, a good algorithm will have the following properties:

- **Effeciency:** The scheduling decisions can be made fairly fast.
- **High Performance:** A good algorithm will always minimize the latency experienced by cells, and achieve a high switch throughput.
- **Fairness**: Each queue will eventually be served.
- **Simple Implementation**: Feasibility of being translated into hardware implementations.

The existing algorithms for the IQ switching architecture majorly fall into two categories: the maximum weight matching (MWM) algorithms and the maximal size matching (MSM) algorithms.

The objective of the maximum weight matching algorithm is to produce the service matrix S_{max} such that it maximize the sum of a pre-define weight. The mathematical expression is given as follows:

$$S_{max} = arg\ max(\sum_{i,j} w_{i,j} s_{i,j})$$

where, $w_{i,j}$ express the weight of $VOQ_{i,j}$, and $s_{i,j}$ express the service to the corresponding VOQ. Existing maximum weight matching algorithms for instance, the longest queue first (LQF), the oldest cell first (OCF) [12], the longest port first (LPF) and the oldest port first (OPF) [29] , use queue length and waiting time of HoL cell as the weight, and achieve 100% throughput under any admissible traffic. The major drawback of the maximum weight matching algorithms is the high algorithm computation complexity $O(N^3 log^N)$, which is too slow to support a high bandwidth switch. An example of the LQF scheduling is given in Fig. 2.10, where the connections are made such that the total weight in the terms of queue length is maximum.

Another class of algorithms, known as maximum size matching algorithms, produce service matrix that maximize the number of connections made every time slot. Intuitively, a maximum number of connections will lead to a maximum throughput of the switch. However, as shown in [26], this class of algorithms may lead to instability and unfairness under admissible traffic. In addition, the maximum size matching algorithms have a high computational complexity of $O(N^{\frac{5}{2}})$ [30], yet it is non-trivial to be implemented into hardware circuit, as maximum size matching involves removing previously made connections to further increase the number of connections. In a nutshell, maximum matching schemes generally lead to a lengthy searching and augmenting process to find the optimum solution. Therefore, researchers compromised to the maximal size matching algorithms (MSM), which forms the major research interest of the IQ switch scheduling.

The maximal size matching algorithms iteratively increase the number of connections made in one time slot. However, unlike the maximum matching schemes, once a connection is made, it cannot be removed later. Within an iteration, the request-grant-accept (RGA) handshaking is performed to establish connections. The specification of each step is as follows:

- **Request:** Each unmatched input sends requests to the outputs for which it has a queued cell. The request signal can be simply 1-bit binary value or log^N bits value indicating the length of the queue or the age of the HoL cell.
- **Grant:** Subject to a scheduling policy, each output send a grant to one of the requesting input.
- **Accept:** Each input choose one out of all the granting output in compliance with a specific scheduling policy.

Numerous maximal size matching algorithms have been proposed and studied [31] [32] [33]. The parallel iterative matching [13] (PIM) algorithm, developed by DEC Systems Research Center for a 16 ports, 1Gbps switch, serves as the basis of later-coming algorithms. Fig. 2.11 shows the operation of PIM within one iteration. In the grant stage, each output randomly picks an input that is requesting and each input randomly picks a granting output to setup a connection. In the next iterations, those unmatched ports will start the RGA again to further increase the number of connections. There are majorly three advantages of the PIM algorithm: *i*) it converges to a maximal connections within $O(LogN)$ iterations, *ii*) the randomness on the grant and accept

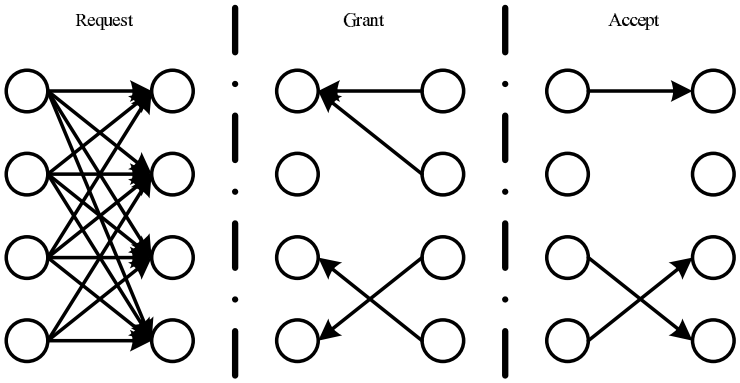

Figure 2.11: Parallel iterative matching algorithm.

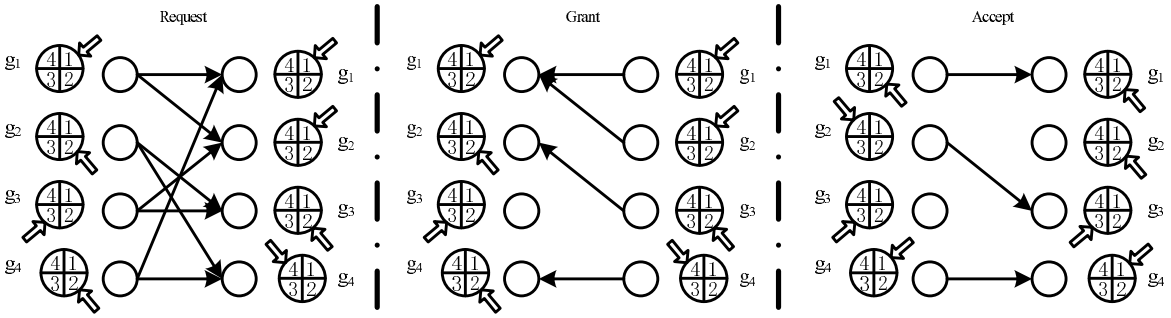

Figure 2.12: Round robin matching algorithm.

stage ensures each flow will eventually be served, and hence fairness is guaranteed, *iii*) due to the randomness, no state has to be kept by the scheduler and hence no memory is required. Though the mentioned advantages, the PIM has significant drawbacks. First it is difficult to be implemented in hardware with high speed, since the hardware must be capable of selecting randomly the inputs and outputs. Second, PIM behaves poorly

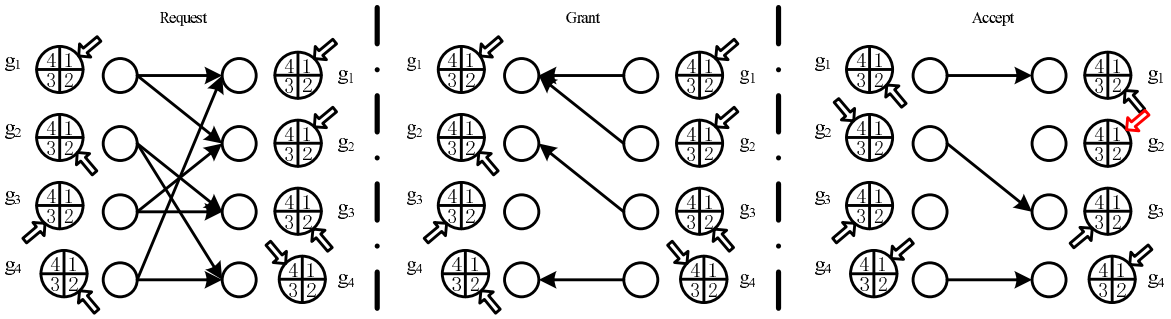

Figure 2.13: The scheduling process of iSLIP algorithm.

with a single iteration, achieving only 63% throughput [26].

The round-robin matching (RRM) scheduling algorithms replace the random mechanism with the round-robin pointers that prioritize the input/output ports. A single iteration of RRM is depicted in Fig. 2.12. The pointers at each output port help to select one of the requesting inputs in the order of starting from the current pointed position. Similarly the pointers at the input ports help to select one of the granting outputs. After each grant or accept is performed, the pointer goes to one location beyond the granted or accepted port. Comparing with the PIM algorithm, the round-robin scheme has a lower hardware complexity, as the round-robin pointer can be easily implemented by a programmable prioritized encoder (PPE). And the rotating priority of the pointer explicitly ensures the fair allocation of switch bandwidth over all the ports. However, the throughput achieved by a RRM algorithm is similar to that of PIM with a single iteration, due to the issue known as pointer synchronization. As can be seen from Fig. 2.12, the RR pointer of the output 1 and output 2 are both pointing at position 1 that correspondingly prioritize the input 1. Thus, when the input 1 has cells destined to the output 1 and 2, both outputs will grant to the input 1. However, only one of the grants can be accepted later, wasting the bandwidth of the other port unnecessarily.

The well-known iSLIP algorithm is based on RRM with a tiny yet significant modification to the pointer update scheme to reduce the pointer synchronization problem. The pointer of output in iSLIP algorithm will get an update only when the grant sent to an input port is later accepted. An iSLIP iteration is depicted in Fig. 2.13, under the same traffic pattern as the RRM algorithm, the pointers of output 1 and 2 become desynchronized after the iteration. The **if condition** added to the pointer update policy results in the properties of the iSLIP: *i*)a recent made connection will become the lowest priority later, since pointers at both input i, and output j gets simultaneously updated, *ii*)fairness, each requesting input will be served within N^2 timeslots, *iii*)when the switch is heavily loaded, the bandwidth is equally spread over all the inputs [26]. Though the capability of achieving 100% throughput using only one iteration under Bernoulli i.i.d arrival, iSLIP behaves poorly under non-uniform traffic patterns that emulates the real network traffic scenarios.

To summarize, the MWM algorithm provides the optimum performance among all, however difficult to be implemented in hardware. On the other hands, the maximal size matching algorithms are more practical and possible to implement in hardware, yet need a couple of iterations to deliver a certain level of performance. Furthermore, as the line rate keeps doubling every 22 month, it is getting more difficult for the centralized scheduler to run multiple iterations of RGA handshaking per time slot. To tackle the performance bottleneck faced by IQ switches, alternative switching architectures have been studied and the buffered crossbar switching architectures are considered as promising candidates.

2.2.6 The CICQ switch

The combined-input-cross-point-queued (CICQ) architecture [34] has been proposed to overcome the performance bottleneck faced by the IQ switch. Fig. 2.14 shows the infrastructure of the CICQ architecture. Different from the IQ architecture, the CICQ archi-

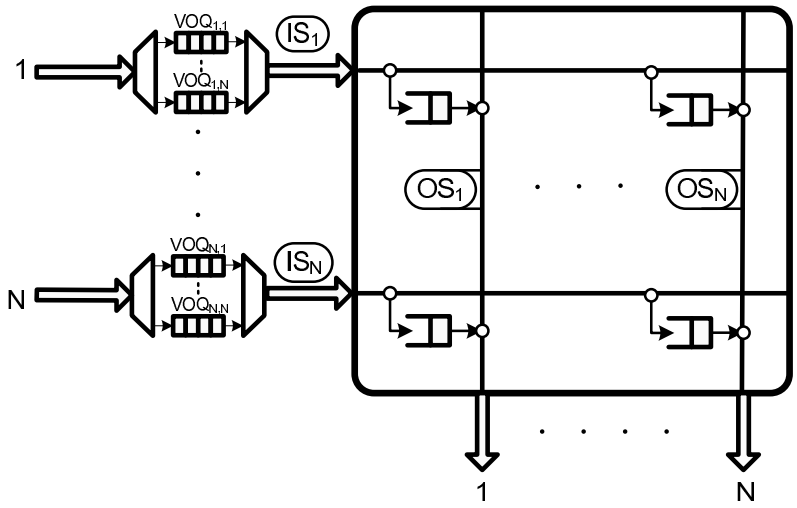

Figure 2.14: An NxN CICQ switch with the VOQs.

tecture introduces a small amount of buffer at each fabric cross point, termed as cross point (XP) buffers. A cell traversing through the CICQ switch will be first transferred to the internal buffer belonging to the destined output, where the cell gets transferred to the outgoing link. Maintaining the cross point buffers mainly bring two advantages: *i*) a relaxed output contention, allowing multiple inputs to transfer cells into the same output in one time slot. *ii*) a distributed manner of scheduling, allowing the design of simple and effective scheduling policies. Putting it together, the CICQ architecture leads to a shortened latency experienced by cells, and decoupled scheduling process that increase the speed at which the scheduling decisions are produced.

For an NxN CICQ switch, there are N input schedulers and N output schedulers. The efficient placement of the schedulers has been discussed in [35]. At each time slot, each input scheduler (IS) chooses a VOQ according to a specific input scheduling policy and transmits the HoL cell of the selected VOQ into the internal buffer located inside the crossbar fabric. On the other hand, each output scheduler (OS) selects a cell from all non-empty XP buffers and moves it outside the switch. There has been extensive studies on the design of scheduling policies for the CICQ architecture [36] [37] [38]. These algorithms generally exhibited performance beyond that of IQ switch.

The simplicity and high performance of the CICQ switch comes at the expense of N^2 internal buffers that grows quadratically with the size of the switch, imposing a limiting factor on the switch scalability. In the next section, we introduce the partially buffered crossbar (PBC) switching architecture that combines advantages from both the IQ and the CICQ switching architectures.

2.2.7 The PBC switch

The partially buffered crossbar (PBC) architecture [18] (depicted in Fig. 2.15) has been proposed aiming at delivering the optimal performance of the CICQ switch, while main-

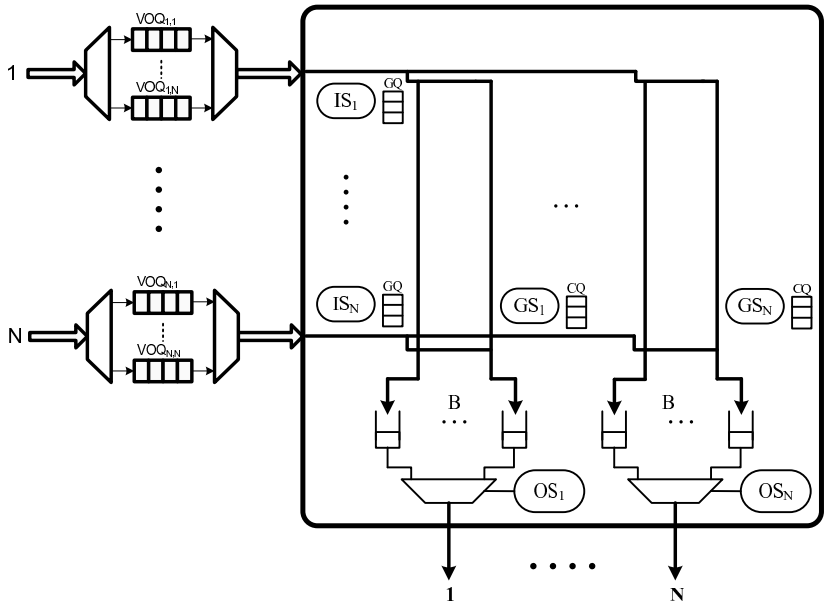

Figure 2.15: The PBC switching architecture.

taining a hardware cost comparable to that of the IQ switch. The PBC switch keeps a small number, B ($B \ll N$) of internal buffers per fabric output, reducing the total number of internal buffers to NB, instead of N^2 in the case of the CICQ switch. Similar to the CICQ switch, the PBC switch has N input schedulers (IS) and N output schedulers (OS) located at each fabric input and output. Additionally N grant schedulers (GS) are introduced managing the access to the internal buffers.

2.2.8 Scheduling in PBC Switches

A scheduling cycle of the PBC switch consists of input and output scheduling phases. While the output-scheduling phase remains identical to that of the CICQ switch, input-scheduling phase of the PBC architecture resembles to the scheduling of the IQ architecture, since the request-grant-accept (RGA) handshaking strategy is adopted to create a mapping between requesting inputs and available internal buffers. At each time slot, each input scheduler may request up to N outputs, and requests from all the inputs are sent to the grant schedulers located at each fabric output. Depending on the scheduling policy and the availability of internal buffers reported by credit queue (CQ), a grant scheduler may issue up to B grants to the requesting input ports. Later, each input port selects and accepts one of the grants and transfers the HoL cell to the internal buffer. In the mean time, each output scheduler selects a cell from all non-empty internal buffers in compliance to the output scheduling policy and transfers the cell outside the switch. While each input can still send one cell in per time slot, the output contention is relaxed by the presence of internal buffers. Previously, only first-come-first-serve (FCFS) policy was allowed to be used as the output scheduling policy to secure the in-order delivery of

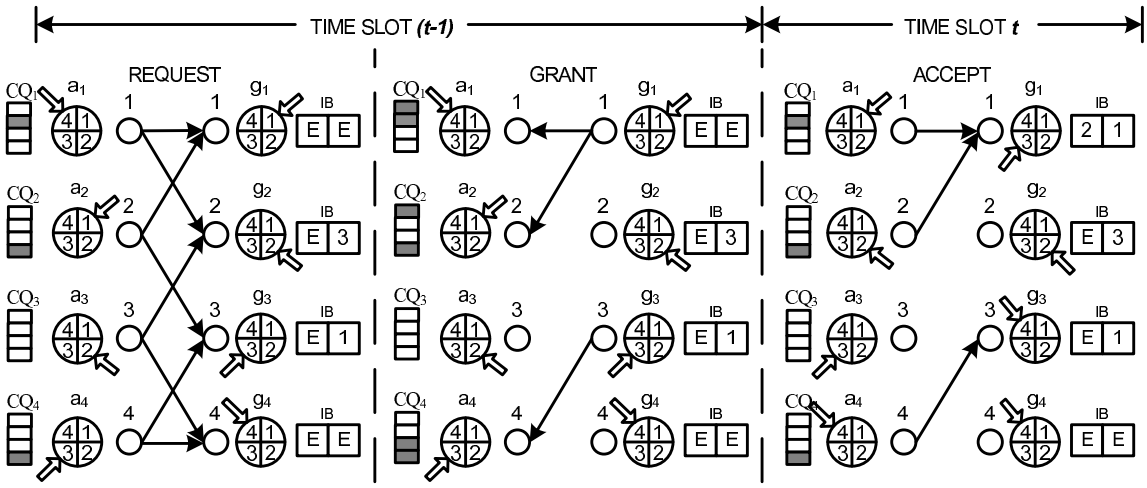

Figure 2.16: The input scheduling phase of DRR for a 4x4 PBC switch with $B = 2$.

cells. In this thesis, we propose a round-robin based output scheduling policy without compromising the order of delivered cells.

A class of round-robin based input scheduling algorithms have been proposed with an FCFS output scheduling policy, termed as distributed round robin (DRR), DROP and prioritized DROP (DROP-PR) [18]. The DRR algorithm adopts fully desynchronized round-robin pointers on input schedulers, and the grant schedulers behave identically to the iSLIP algorithm as discussed earlier. The grants issued to an input port, if not immediately accepted, will be stored and eventually get consumed. The stored credits then lead to a problem known as credit release delay degrading the performance of DRR severely. While an input port may receive multiple grants in one time slot, only one will be returned. It may take up to N time slots before a credit is returned to the grant schedulers, reducing the rate at which the grant schedulers issue credits to the other inputs. In Fig. 2.16, both internal buffers of output 4 are empty, yet its credits are held by input 2 and input 4 but not accepted, resulting in an unnecessary waste of switch throughput. To reduce the credit release delay, [39] proposed a threshold scheme that prevent an input port from receiving credits, when the number of stored credits goes beyond a pre-defined threshold. Furthermore, two augmented versions of DRR algorithm, DROP and DROP-PR eliminate the credit release delay completely.

As suggested by the name of the algorithms, credits, not accepted by the inputs, are dropped (forgotten) instead of stored (reset the $GQ_{*,*}$ to zero). Hence, every time slot each grant scheduler will guarantee to have one credit returned back. Since the input ports no longer store the credits, the grant scheduler pointers are changed to be fully desynchronized to avoid performance degradation from pointer synchronization (also known as static round-robin SRR). The DROP-PR algorithm further improves the performance of DROP by giving priority to the unloaded output ports. When an output has no cell in its buffer, its priority bit will be set to 1 and attached to the credits issued to the input ports. The inputs then favor the credits with priority bit

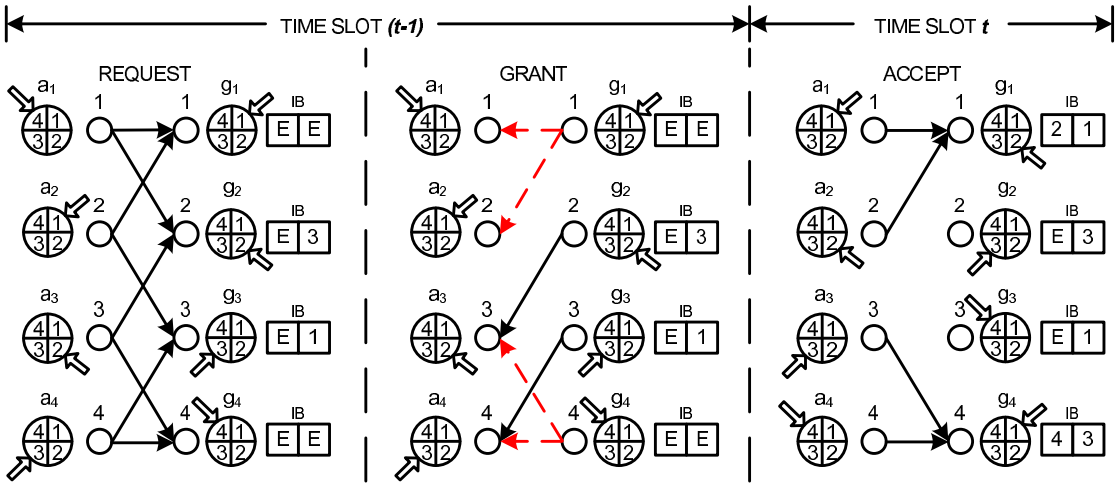

Figure 2.17: The input scheduling phase of DROP-PR for a 4x4 PBC switch with $B = 2$.

being 1 over the normal ones. Hence, empty output ports will have good chances to become loaded in next time slot. To better show the idea, Fig. 2.17 provides a graphic view of the DROP-PR input scheduling process. The red dotted arrows represent the credits with a higher priority. As a result, both the output 1 and output 4, which were originally empty, become fully loaded after the input scheduling stage. The DROP-PR algorithm with 8 internal buffers per output converges to the performance of the OQ switch under Bernoulli i.i.d traffic, regardless of the size of the switch[18]. However, its performance under non-uniform traffics is less appealing, without a sufficient number of internal buffers. In the next chapter, we will propose a unicast-scheduling algorithm that improves the performance of the PBC switch under non-uniform traffic scenarios.

2.3 Multicast Problem

The rapid development of Internet applications, such as IPTV, voice-over-IP (VoiP) and video conferencing, raises the need of multicast support by the backbone routers. In contrast to unicast cells, multicast cells may have a number of destinations from 1 to N, where N is the number of the output ports of a switch. In fact, supporting multicast traffic is to design a switch capable of copying one cell to multiple outputs. The crossbar based switching architectures natively support multicasting, as a cell can be transferred to multiple output ports simultaneously by closing multiple cross points of the fabric.

The set of destinations of a multicast cell is defined as the fanout set. For an N x N switch, the total number of possible fanout sets is $2^N - 1$. Avoiding HoL blocking completely under multicast context requires $2^N - 1$ FIFOs per input. This queuing scheme is known as the multicast VOQ (MC-VOQ) and is impractical to be used in practice. As a result, researchers proposed to maintain a single FIFO (also referred as multicast queue 'MQ') at each input ports. While being a practical scheme, it leads to a poor performance due to the HoL blocking issue. The k-FIFO queueing scheme is then

proposed to make a good compromise between 1 and $2^N - 1$, where a small number, k of FIFOs are maintained at each input port. It has proven to be an effective solution that boosts the switch performance with affordable hardware cost. The only overhead brought by k-FIFO queuing scheme is the introduction of a circuit that maps the total number of fanout sets into k FIFOs. The mapping is then known as the cell assignment policy. A good cell assignment policy should have following properties: *i*) HoL cells should contain diverse fanout sets that can span over a large number of output ports. *ii*) Cells with the same or similar fanout sets should be placed in the same multicast queue. This reduces the HoL issue and avoids the out of sequence delivery of cells. Different cell placement policies have been proposed, such as Majority and Split [40], Minimum Distance Queue (MDQ), Load Balanced Queuing (LBQ) [41], and Modulo [42].

The multicast scheduling algorithms have also been investigated for both the IQ and the CICQ switching architectures, with either the single FIFO or the k-FIFO queuing scheme. The idea of fanout splitting was introduced in [43]. When the fanout splitting is allowed during scheduling, HoL cell in a FIFO is possible to be partially discharged. Otherwise, the HoL cell can only be discharged when all its destinations are available. Consider the example in Fig. 2.18. If fanout splitting is allowed, both MQ 1 and MQ 2 can discharge part of the HoL cells, leading to a higher switch throughput. In general, allowing the fanout splitting will have the switch tend to be more work conserving. The set of destinations remained after the scheduling is defined as the residue of the fan out sets. Two disciplines are proposed to deal with the residue distribution, namely the concentrated and the distributed policies. When the concentrated policy is applied, the schedule will be made such that the residue will get concentrated onto as few numbers of ports as possible. When the distributed policy is applied, the residue will be spread over as many numbers of ports as possible. Take Fig. 2.18 for example, the residue is $2, 3$, if we concentrate it to MQ 1, MQ 2 will discharge the HoL cell completely and expose the next cell behind. If we distribute the residue over these two MQs, for example 2 on MQ1 and 3 on MQ2, none of the HoL cells can be fully discharged, limiting the switch throughput in the next time slot to be 50% only. Apparently, the concentrated policy will lead to a better performance of the switch, as it tries to fully discharge as many HoL cells as possible.

Representative multicast scheduling algorithms of the IQ switching architecture are TATRA [43] and multicast RRM (mRRM) [44]. TATRA is inspired by the popular block-packing game Tetris and is regarded as one of the state-of-the-art scheduling algorithms. TATRA concentrates the residue and guarantees that at least one cell will be fully discharged during each time slot. Unfortunately, TATRA is a complex algorithm and cannot be parallelized, making it impractical to implement in hardware. The mRRM algorithm, on the other hand, is a simple algorithm that uses round-robin pointers to resolve the contention. Different from what we have discussed in the unicast scheduling, the pointers of mRRM are fully synchronized to concentrate the residue and discharge cells completely. For the multicast scheduling in CICQ switching architecture, the multicast cross point round robin (MXRR) algorithm was proposed in [45]. The MXRR employs fully desynchronized pointers in the input schedulers to provide a diverse exposure of HoL cells, and uses fully synchronized pointers in the output schedulers to discharge cells entirely from the internal buffers. While being a simple round-robin based scheduling

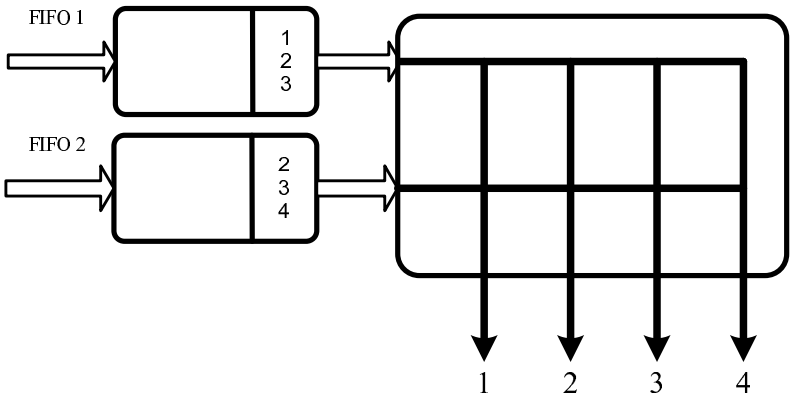

Figure 2.18: A 2x4 crossbar switch with multicast cells.

algorithm, it outperforms the complex TATRA algorithm mentioned above, exhibiting an unmatched performance.

The recently proposed PBC switching architecture has emerged to be a combination of optimal performance and low hardware cost when scheduling unicast traffics. However, its multicast capability has not yet been explored. We will conduct an experimental study on the PBC multicast capability in Chapter 4 and propose a round-robin based algorithm to compete with the aforementioned MXRR algorithm of the CICQ architecture.

2.4 Summary

In this chapter, we provided an overview of previous development of crossbar based switching architectures. The OQ switch can provide optimal performance among all, but impractical to implement with a high line rate. Therefore, the interest of research shifted to the IQ architecture where memory bandwidth requirement is much lower. Since the centralized scheduler of the IQ switch cannot make scheduling decisions fast enough and with high performance, the CICQ switch emerged to break the bottleneck confronted by the IQ switch. By introducing the internal buffers and the distributed schedulers, the scheduling process becomes simple and the performance of the CICQ switches goes beyond that of the IQ architecture. However, the N^2 cross point buffers inside the fabric limit the scalability of the CICQ architecture, N is the size of the switch. The PBC switching architecture combines the advantages from the IQ and the CICQ architectures, and appears to be one of the attractive candidates to the next-generation router architecture. In the next two chapters, we will study the unicast and multicast scheduling issues of the PBC switch, and improve its performance.

Scheduling Unicast Traffic in PBC switches

3

The partially buffered crossbar (PBC) switch maintains a small number of buffers per output. While having a cost close to unbuffered crossbars, a PBC switch overcomes the centralized scheduling complexity by means of distributed schedulers resulting in high speed switching and simplicity in its scheduling. Previously, a class of round-robin algorithms has been proposed for the PBC and demonstrated similar performance to an output queued (OQ) switch under Bernoulli uniform traffic. However, it fails to deliver satisfactory performance under non-uniform traffic unless a high number of internal buffers is used. In this chapter, we propose a novel scheduling algorithms, named ELSRR (Exhaustive-LQF-SRR) that enhances the performance of a PBC switch under non-uniform traffic. Through experimental study, we show that it is capable of delivering high throughput under non-uniform traffic with low requirement on the internal buffer per output; as few as two internal buffers per output irrespective of the switch size.

3.1 Introduction

The crossbar-based architecture has been favored as the high-speeding switching fabric for its scalability and non-blocking property. Therefore, most commercial implementation and literature study are based on crossbar architecture with Virtual Output Queues (VOQs)[26]. The crossbar switching fabric falls into two categories: unbuffered and internally buffered crossbar switches.

Unbuffered crossbar switches, also known as input queued (IQ) switches, employ a centralized scheduler to resolve packet contention at the inputs and outputs of the switch. The Maximum Weight Matching (MWM) and Maximal Size Matching (MSM) algorithms have been proposed to achieve 100% throughput for an IQ switch [12][14][16][46][47]. However, with ever increasing line rates, scheduling in IQ switches is either too complex to be practical or too simple to deliver satisfactory performance. The Buffered crossbar (CICQ) architecture[48] has been proposed to effectively address the scheduling complexity faced by IQ switches. With a small number of buffers at each cross point of the crossbar fabric, the scheduling in a buffered crossbar is divided into distributed input and output schedulers at each input and output port of the switch. This design significantly simplifies the scheduling and reduces the time required to produce a good configuration of the siwtch. Various scheduling algorithms have been proposed for buffered crossbars[48][17][49][50], exhibiting performance beyond that of IQ switches. However, the boost of performance and scheduling simplicity comes at the prices of expensive cross point buffers that grow quadratically with the switch size.

In order to achieve high-performance while maintaining a cost close to an unbuffered switch, the Partially Buffered Crossbar switching architecture (PBC) has been proposed [18], in which B ($B \ll N$) shared internal buffers are deployed at each fabric output port.

This design reduces the growth of internal buffers to a linear function of the switch size N and hence a significantly lower cost compared to the buffered crossbar architecture. The scheduling process in the PBC architecture is a fusion of that in buffered and unbuffered crossbar switches: distributed schedulers located at each input and output (as in CICQ switches) and a request-grant-accept (RGA) input scheduling that maps requesting inputs into the shared internal buffers (as in IQ switches). A class of round-robin algorithms has been proposed in [18] that showed optimal performance close to that of an output-queued(OQ) switch under bernoulli uniform traffic irrespective the switch size. On the other hand, its performance under non-uniform traffic is less appealing without the support of internal buffer as much as eight per output. The degradation of performance under non-uniform is due to the lack of ability to identify a congested input queue demanding for more bandwidth and the dynamic of internal buffer sharing scheme without reservation.

In this chapter, we propose the ELSRR algorithm that enhances the performance of the PBC switch under non-uniform traffic patterns. The motivation is to reserve internal buffers for congested VOQs until they are emptied and service is equally delivered between VOQs with reservation at output. Through simulation we show that, with as few as two internal buffers per output, the proposed algorithm is capable of outperforming the previous algorithms, delivering high throughput and scalability under various non-uniform traffic scenarios. The reminder of this chapter is organized as follows: in Section 3.2 we introduce the PBC switching fabric and review the previous scheduling algorithms. Section 3.3 we describe the proposed ELSRR algorithm and its properties. Section 4.4 shows the experimental study. In Section 4.5, the hardware design of the ELSRR is presented, and Section 4.6 summarizes the whole chapter.

3.2 Background

3.2.1 The Switching Model

The infrastructure of a partially buffered crossbar (PBC) switch is shown in Fig. 3.1. It has N inputs and N outputs, and each input contains N VOQs, one per output. Incoming packets are segmented into fixed-length cells in order to be processed by the switch, and cells are later re-assembled back into a packet upon departing the switch. The PBC switch has $2N$ distributed schedulers: N input schedulers and N output schedulers located at each input and output, functioning in parallel and resulting in a simplified, high-speed scheduling. Due to the fact that the number of internal buffers maintained per output, B, is much less than the switch size N, an input scheduler of PBC switch cannot make decision by itself to admit a cell into the fabric. Instead, the decision is made in coordination with a grant scheduler, which manages the availability of internal buffers via the information provided by a credit queue (CQ). To make input scheduler visible to granting information, a grant queue (GQ) is deployed at each input scheduler, which is a table with N entries, representing the N outputs. For instance, if an input i is granted by output j, the entry $GQ_{i,j}$ will be set to one, otherwise zero.

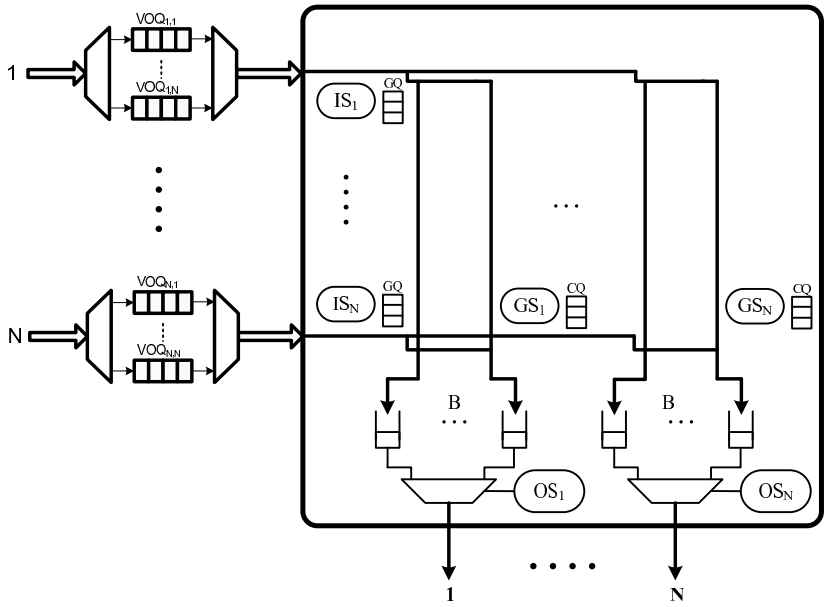

Figure 3.1: The PBC switching architecture.

3.2.2 Scheduling in PBC

A scheduling cycle of a PBC switch consists of input and output scheduling phases. While the output-scheduling phase remains similar to that of a CICQ switch, the input-scheduling phase resembles to that of unbuffered crossbars, as the request-grant-accept (RGA) handshaking strategy is adopted. At each time slot, an input scheduler may request for up to N grant schedulers located at each fabric output. Depending on the scheduling policy and the availability of internal buffers shown as (CQ), a grant scheduler may issue up to B grants towards the input ports requesting for it. An input port later may select and accept one grant and transfer the HoL cell of the selected VOQ to the internal buffer of the output port granting it. In the mean time, each output scheduler selects a non-empty internal buffer in compliance to the output scheduling policy and ejects the cell inside to the output. While each input can send at most one cell per time slot, the output contention is relaxed by the presence of internal buffers, which means more than one cells can be received in one time slot resulting in a simpler scheduling. Previously, the output scheduler uses FCFS policy to ensure the in-sequence delivery of cells that are coming from a same queue. We will later show that with a different buffer sharing strategy, the output scheduler can adopt other scheduling algorithms with in-sequence delivery guaranteed as well.

The DRR, DROP and DROP-PR are a class of pipelined round-robin algorithms proposed in [18]the PBC switch. Particularly, the DROP-PR algorithm shows the best performance over the others as it gives priority to unloaded output ports and eliminates the credit release delay suffered by DRR. The specification of the DROP-PR algorithm is given below:

Algorithm 1. DROP-PR

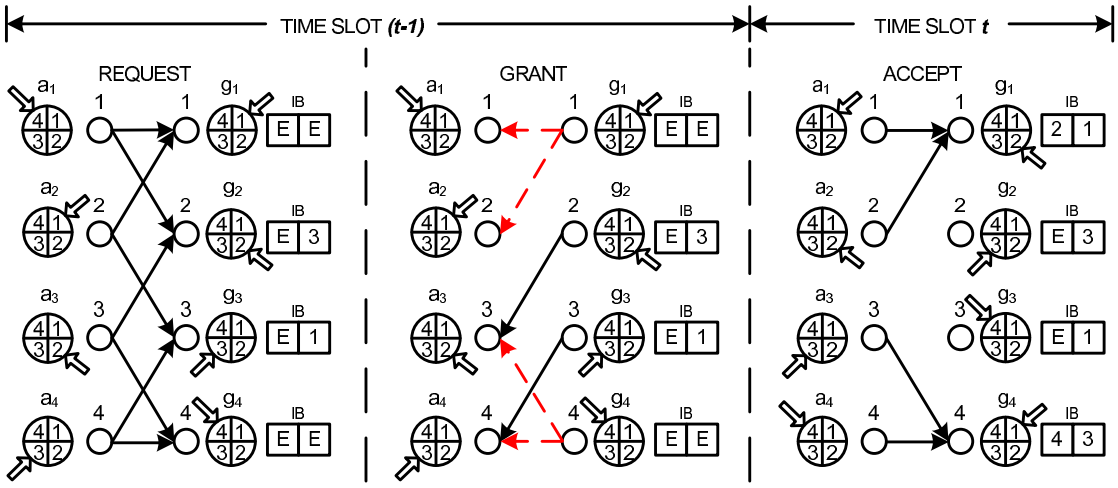

Figure 3.2: A DROP-PR input scheduling cycle for a 4x4 partially buffered crossbar switch with $B = 2$. Each input sends requests to all outputs for which the VOQ has a queued cell. At the grant stage, each grant scheduler issues credits to requesting input in a round-robin pointer order and inform the granted inputs with its occupancy (shown in red arrow). When it comes to the accept stage, each input scheduler favors the grant from a empty output port to balance the workload among all the output ports. Later the unaccepted credits are dropped to avoid the credit release delay that degrades switch performance.

Grant Phase:
All grant pointers, g_j, are initialized to different positions.
For each grant scheduler, j, do

- Set CQ_j equal to the number of available internal buffers of output j.
- Set priority bit, P, to the logic OR of CQ_j entries.
- While there are credits in CQ_j, do:
  - Starting from g_j index, send a grant to the first input, i, that requested this output (set $GQ_{i,j} = 1$ and add bit P).
  - Decrement CQ_j by one.
- Move the pointer g_j to location $(g_j + 1)(mod\ N)$.

Input Scheduling Phase
All accept pointers, a_i, are initialized into different positions.
For each input scheduler, i, do:

- Starting from a_i index, select the first nonempty $VOQ_{i,j}$ for which $GQ_{i,j} = 1$ AND $P = 1$, send its HoL cell to the internal buffer.
- If no VOQ is selected in the previous step, then

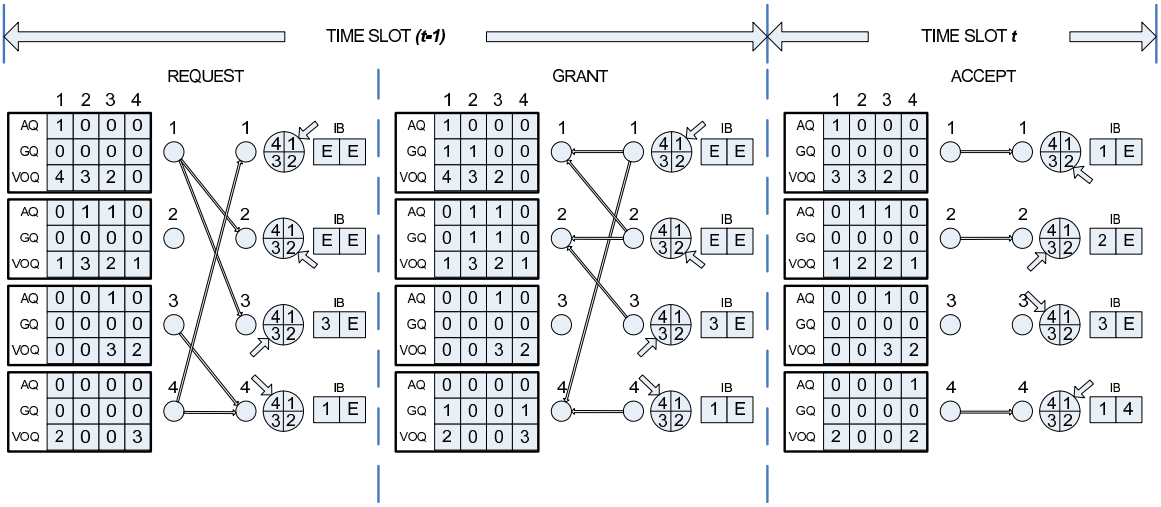

Figure 3.3: The ELSRR input scheduling cycle for a 4x4 partially buffered crossbar switch with $B = 2$.

– Starting from a_i index, select the first nonempty $VOQ_{i,j}$ and send its HoL cell to the internal buffer.

- Drop the remaining grants (reset GQ: $GQ_{i,*} = 0$).
- Move the pointer a_i to location $(a_i + 1)(mod\ N)$.

A graphic demonstration is shown in Fig. 3.2. Both the input scheduler and the grant scheduler of DROP-PR adopt round-robin algorithm with fully desynchronized pointers, resulting in a solid fairness amongst all VOQs. However, strong fairness will leave a congested queue growing indefinitely, and hence its performance under non-uniform traffic is degraded.

By observing the DROP-PR algorithm, we argue that there are two main reasons degrading its performance under non-uniform traffic. First, as a fair scheduling algorithm, it cannot identify a congested queue and provide more bandwidth to it. Secondly, with limited number of internal buffers, service of an output port is still delivered to all the VOQs, leading to a poor internal buffer management and hence each VOQ can only receive a small fraction of bandwidth.

3.3 The Proposed Algorithm

With the aforementioned observation, the motivation of the ELSRR is that instead of sharing B buffers of an output among all N inputs as in the way of DROP-PR, we explicitly reserve internal buffers to maximum B inputs, one buffer per input. Amongst these selected inputs, services are equally delivered. The scheduling algorithm should then be able to identify a congested queue and serve it more frequently. To actually

fulfill the idea, we first classify all the N^2 VOQs into two classes: *admitted* VOQs and *non-admitted* VOQs.

Defination 1: A VOQ is admitted if its destined output reserves one buffer for it.

Being admitted, a VOQ will receive a grant from a grant scheduler as long as the VOQ has no cell currently stored inside the output. In opposition to the DROP-PR, this policy ensures that no VOQ can occupy more than one buffer of an output, and hence a finer buffer management. We say an admission to a VOQ is cancelled when there is no longer any cell in it. For a non-admitted VOQ, it may receive grants only when the destined output still has free internal buffers to reserve. On the input scheduler side, grant priority is given to an admitted VOQ over non-admitted one. To prevent an input port from being over-admitted and resulting in an unbalanced resource reservation, no input can have more than B VOQs being admitted at one time. This is done by introducing an Admission Credit Counter (ACC) at each input scheduler. The ACC decrements by 1 when a new admitted VOQ is promoted, and incremented by 1when an admission to a VOQ is cancelled. Once the ACC counts down to zero, the input scheduler stops posting requests for any of the VOQs, and thus no more admission can be made to that input. Similar to the GQ maintained at each input port that signals the input scheduler whether a VOQ is granted, we introduce an Admission Queue AQ that notifies the input scheduler about the VOQs being admitted by the output ports. The algorithm specification is given below:

Algorithm 2. ELSRR

Grant Phase:

All grant pointers, g_j, are initialized to different locations. For each grant scheduler, j, do:

- Check all inputs, i, that have $AQ_{i,j} = 1$.
  - If $VOQ_{i,j}$ is empty, set $AQ_{i,j} = 0$ and ACC_i increment by 1.
- Set the CQ_j equal to the number of available internal buffers of output j.
- Send grants to all inputs, i, with $AQ_{i,j} = 1$ and no cell in the internal buffer.
- while there are credits left in CQ_j, do
  - Starting from g_j index, send a grant to the first non-admitted input, i, that requested this output (set $GQ_{i,j} = 1$).
- Move the pointer a_i to location $(a_i + 1)(mod\ N)$.

Input Scheduling Phase:

For each input scheduler, i, do:

- Accept the grant to the longest $VOQ_{i,j}$ with $AQ_{i,j} = 1$.
- If no grant is accepted,
  - Accept the grant to the longest $VOQ_{i,j}$ with $AQ_{i,j} = 0$, Set $AQ_{i,j} = 1$ and ACC_i decrement by 1.

- Drop the remaining grants (reset GQ: $GQ_{i,*} = 0$).

Output Scheduling Phase:
For each output scheduler, j, do:

- Starting from o_j index, select the first non-empty internal buffer, i, and ejects the cell out of the switch.
- Move the pointer o_j to location $(i + 1)(mod\ B)$

Note that we adopt a round-robin output scheduler here to move internally buffered cells out of the switch. This is a crucial as it will help the grant scheduler to grant each of the admitted VOQs within $K - 1$ time slots, where K is the number of VOQs admitted to this output. Recall that a grant scheduler will not issue any grant to a VOQ if that VOQ already has one cell buffered inside the output port. Therefore it is safe to employ a round-robin output scheduler while the in-sequence delivery of cells is ensured. This is simpler than the previously adopted FCFC output scheduling policy.

Fig. 3.3 illustrates an input scheduling phase of the ELSRR for a 4x4 switch with $B = 2$ internal buffers per output. Note that the scheduling process is pipelined into two stages: the request and grant stage and accept stage. Grant decisions made by the grant schedulers (GS) at time slot $(t - 1)$ will be available for the (IS) in time slot t. In the Figure, AQ, GQ and IB represent *admission queue*, *grant queue* and *internal buffer*, while E represents an empty internal buffer. At request stage, each input requests for non-empty VOQs that are not admitted to corresponding output. Note that input 2 posts no requests even $VOQ_{2,1}$ and $VOQ_{2,4}$ are not empty and non-admitted, this is because there are already two admitted VOQs in input 2, and hence it can no longer request for more internal buffer reservation. At grant stage, each grant scheduler will first issue grants to the VOQs that are admitted to the output (i.e. those with $AQ_{i,j} = 1$) and have no cells currently stored in the internal buffers. For example, grant scheduler 2 issues a grant to input 2 since $VOQ_{2,2}$ complies to the aforementioned conditions. With all admitted VOQs served, grant scheduler 2 issues the remaining credit to the requesting input 1. Note that grant scheduler 3 does not issue a grant to the admitted $VOQ_{3,3}$, since there is already a cell from it stored in the buffer. In the accept stage, that happens in a successive time slot to the request and grant stage, each input scheduler first checks and accepts the grant to the longest admitted VOQ. If no grant is selected, it accepts the grant to the longest non-admitted VOQ and marks the VOQ as admitted. Once a VOQ is admitted, it will always receive a grant from the grant scheduler within $(B - 1)$ time slots.

3.4 Experimental Result

In this section, we present the performance of the ELSRR algorithm and compare it to the following algorithms: the DROP-PR of the PBC architecture, the iSLIP of the IQ architecture and the LQF-RR of the CICQ architecture. The LQF-RR algorithm is proven to be stable under any admissible uniform traffic; therefore it also serves as an optimal performance reference in our performance studies. All algorithms are simulated

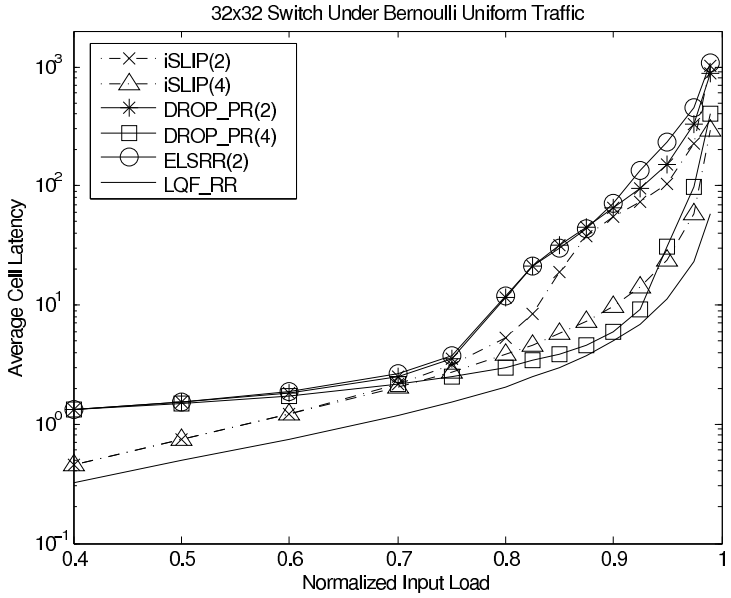

Figure 3.4: Performance Under Bernoulli Uniform Traffic.

for 1.000.000 time slots under the following traffic patterns: Bernoulli uniform, Bursty uniform, Unbalanced Traffic, Double Diagonal Traffic and Logarithmic Diagonal Traffic, and the performance is evaluated in the term of average cell latency and switch throughput. More information about the simulation environment and the traffic definitions can be found in Appendix A.

3.4.1 Uniform Traffic

We test the performance of our proposed algorithm under two uniform traffic scenarios: Bernoulli uniform and Bursty uniform traffic with burst length of 16.

Fig. 3.4 demonstrates the average cell latency of a 32×32 switch under Bernoulli uniform traffic, running the selected collection of algorithms. We fixed the number of internal buffers used by our algorithm to two and compare its latency to that of iSLiP with 2 and 4 iterations, denoted as iSLIP(2) and iSLIP(4), DROP-PR with 2 and 4 internal buffers, denoted as DROP-PR(2) and DROP-PR(4) and LQF-RR. We observed that our algorithm shows slightly higher latency than iSLIP(2) and DROP-PR(2). Unlike iSLIP and DROP-PR where fairness of service is guaranteed, the ELSRR only serves B flows at a time, which implies a weaker fairness of service. Under uniform traffic, leaving most of the traffic flows unserved translates immediately into a higher latency, although not significant. For the performance under Bursty uniform traffic, as shown in Fig. 3.5, our algorithm outperforms iSLIP and DROP-PR, having the best performance that is very close to LQF-RR. With the ELSRR, a burst of cells, once being admitted to the output, will be served consecutively. Hence the latency is reduced. Note that, with the performance close to LQF-RR, there are only 2 buffers deployed at each output port, as compared to 32 for LQF-RR.

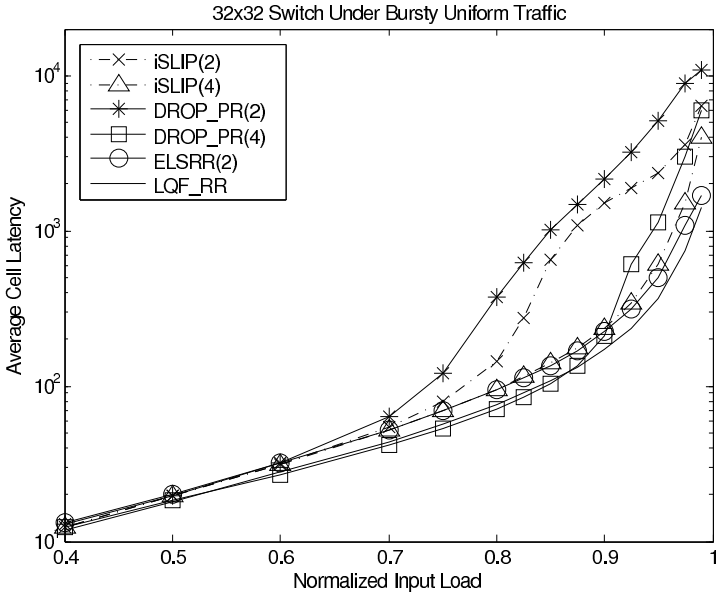

Figure 3.5: Performance Under Bursty Uniform Traffic.

3.4.2 Nonuniform Traffic

The ELSRR algorithm is tested under three non-uniform traffic scenarios: the unbalanced traffic, the double diagonal traffic and the logarithmic diagonal traffic [51]. These three traffic patterns are considered difficult to schedule, hence a decent benchmark to evaluate the performance of a scheduling algorithm.

To evaluate the performance of our algorithm in terms of cell latency under unbalanced traffic, we fixed the unbalanced coefficient ω to 0.5 corresponding to the most unbalanced case and vary the input traffic load. Fig. 3.6 shows the result of the experiment. As we can see, with only two buffers per output, our algorithm outperforms both iSLIP and DROP-PR, while having a inferior performance to that of LQF-RR.

We study the stability of the proposed algorithm by fully loading the switch with a maximum arrival rate at each input port and varying the unbalanced coefficient ω from 0 to 1 with a step size of 0.1. As shown in the Fig. 3.7, our algorithm is stable within all the range of ω together with LQF-RR, yet using 960 less internal buffers. Since the input scheduler always favors the longest granted queue, a congested VOQ will eventually be admitted to its destined output port and regularly receiving service until the VOQ is emptied. And this is critical feature for an algorithm to provide good performance under unbalanced traffic scenarios.

Fig. 3.8 presents the latency performance under double diagonal traffic. This traffic is known to be critical and hard to schedule since there are only two matching patterns that are good. While the performance of iSLIP saturates at 0.8 input traffic loads, both the proposed algorithm and DROP-PR can provide desirable performance. Having performance same as the LQF-RR, the proposed algorithm has a slightly higher latency than DROP-PR. We think the extra latency is caused by the weaker fairness provided

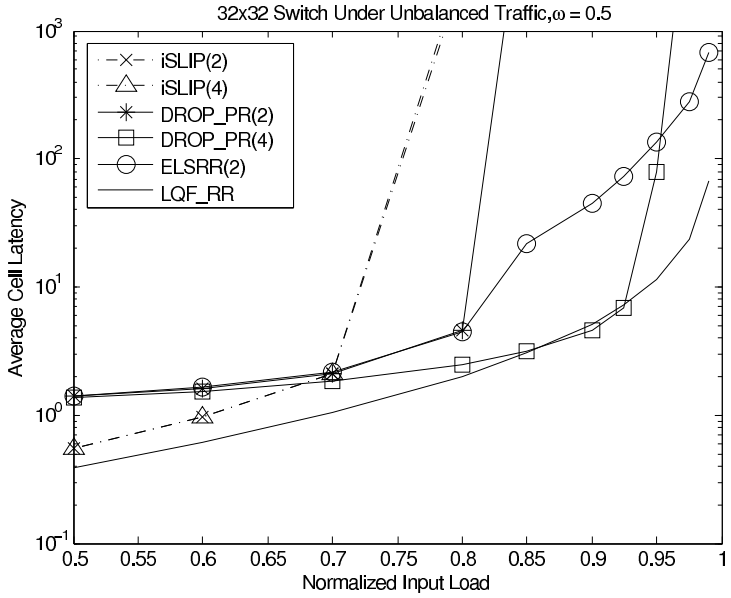

Figure 3.6: Performance Under Unbalanced Traffic.

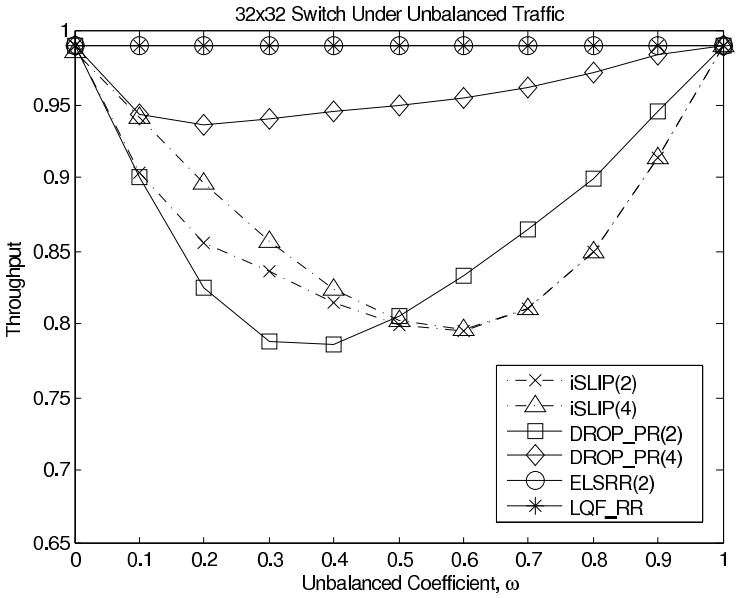

Figure 3.7: Switch Throughput Under Unbalanced Traffic.

by our algorithm. On one hand, the grant scheduler is fair and spread the grant credit equally on all admitted VOQs. On the other hand, however, the input scheduler may not accept a grant to a lightly loaded VOQ.

The throughput under logarithmic diagonal traffic is shown in Fig. 3.9. While the

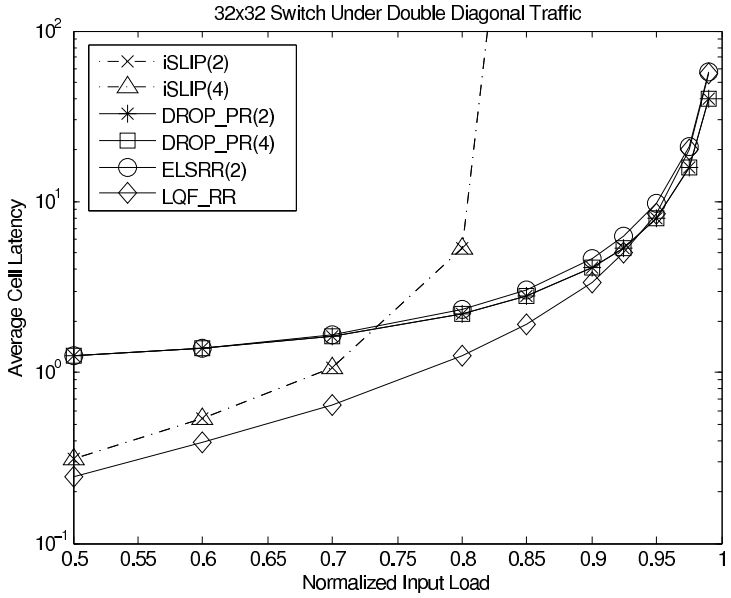

Figure 3.8: Performance Under Double Diagonal Tarffic.

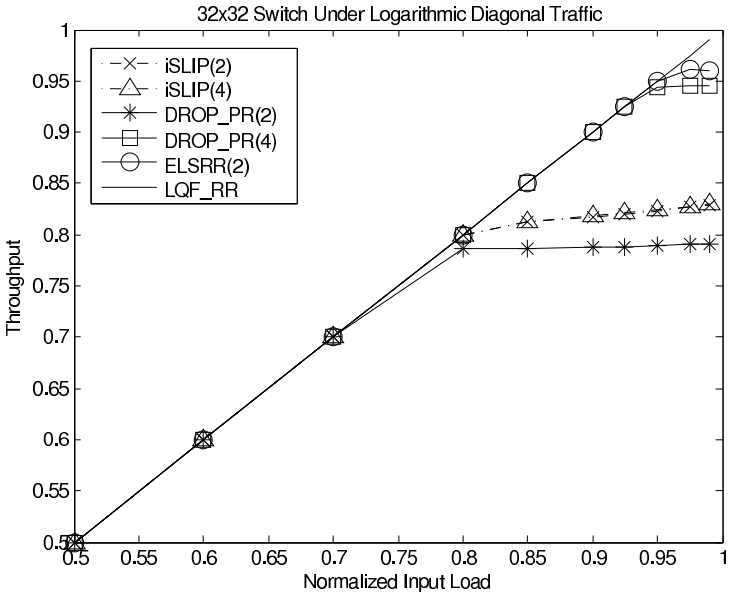

Figure 3.9: Performance Under Logarithmic Diagonal Tarffic.

ELSRR with $B = 2$ achieves 96% throughput, the throughput of DROP-PR(2) saturates at 80%. The proposed algorithm also surpasses DROP-PR(4) by 1% of throughput at maximum traffic load. The LQF-RR outperforms the rest, achieving a 99% throughput. However, comparing the ELSRR to the LQF-RR, we consider loosing 3% throughput for

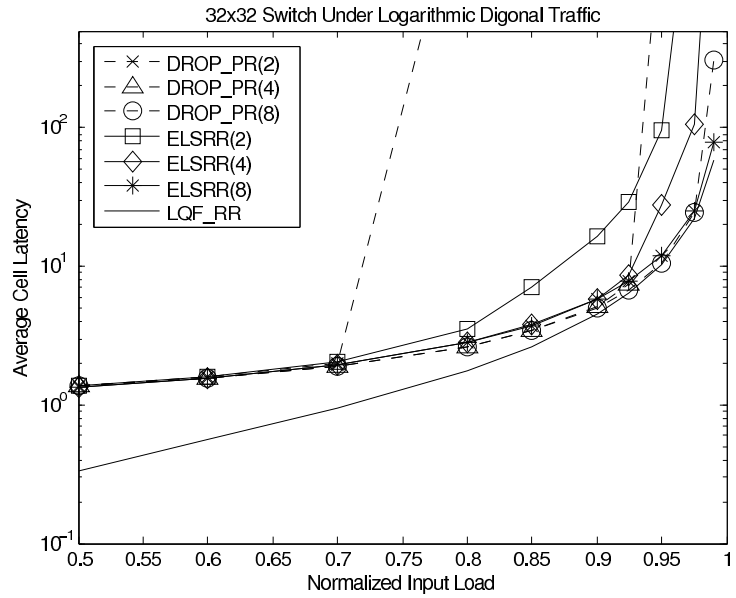

Figure 3.10: Performance Under Logarithmic Diagonal Tarffic.

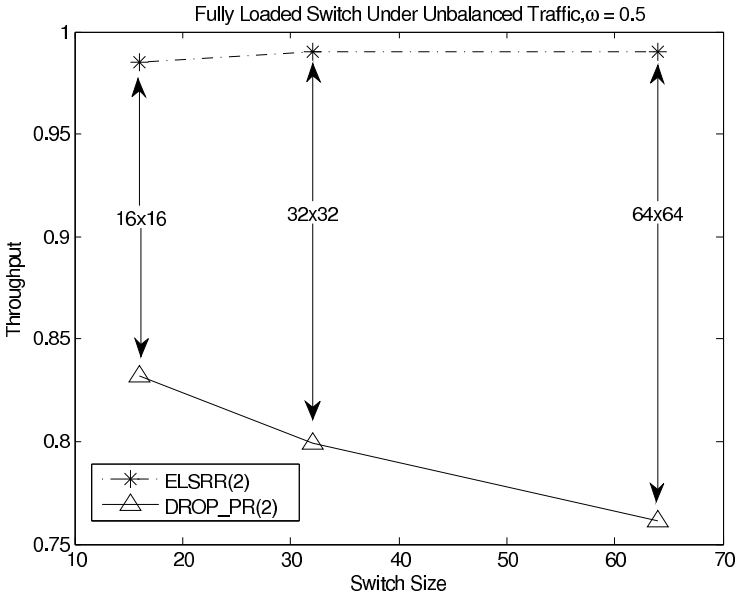

Figure 3.11: Performance Under Unbalanced Traffic.

saving 93.8% internal buffers a fair trade-off. In Fig. 3.10, we see that with a increased number of internal buffers per output, the ELSRR algorithm is also capable of achieving 99% throughput with a desirable latency very close to that of LQF-RR. The boost of performance is a result of allowing more VOQs being served by an output with a increased

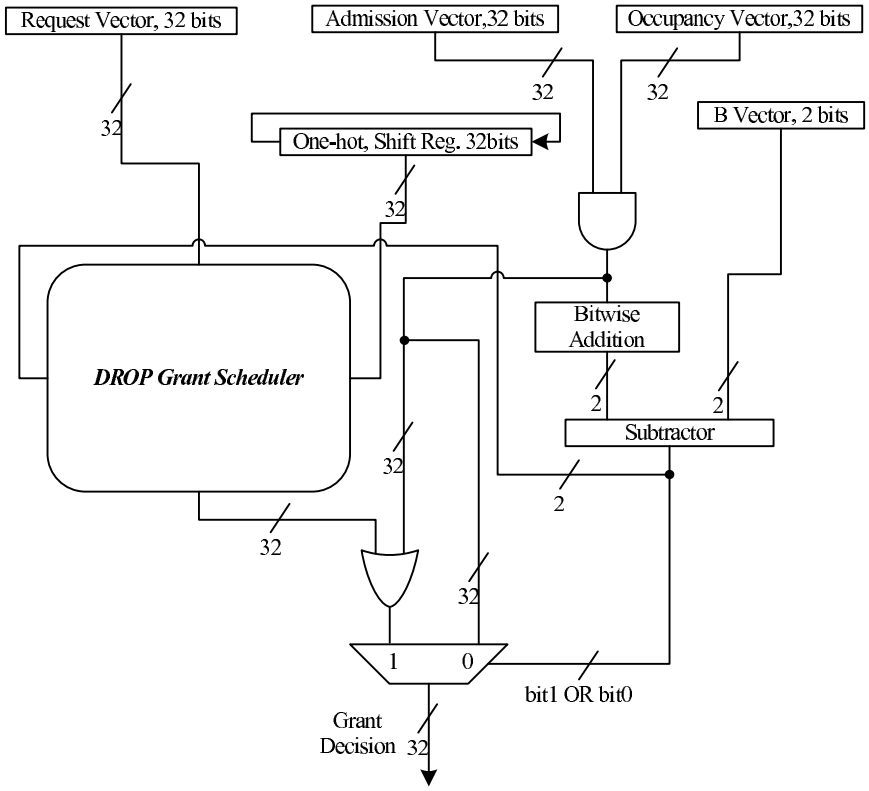

Figure 3.12: Hardware Designed of the ELSRR Grant Scheduler.

buffer budget.

Fig. 3.11 presents the scalability of the proposed algorithm in terms of throughput under the unbalanced traffic with unbalanced coefficient $\omega = 0.5$ and maximum arrival rate. We conduct the test for switch with $N = 16, 32$ and 64 and compare the result to DROP-PR(2). As shown in the figure, the proposed algorithm is able to maintain high through, which is not sensitive to a growing number of ports. On the other hand, the throughput of DROP-PR(2) keeps falling as the switch size scaling up.

3.5 Hardware Design

In this section, we present the hardware design of the ELSRR scheduling algorithm, including the design of the grant schedulers and the input schedulers. The design of output schedulers is not presented, since it is identical to the plain round-robin scheduler design that has been discussed in [14]. The design assumes a 32x32 PBC switch with two internal buffers per output. ($B = 2$). This means a grant scheduler can issue up to two grants to the input ports within one time slot. Fig. 3.12 presents a block diagram of the proposed ELSRR grant scheduler design. The circuit has five inputs, including 1) the request vector that collects request signals from all the input ports to output j (i.e. $REQ_{*,j}$), 2) the admission vector that collects admission signals from all input ports to

output j (i.e. $AQ_{*,j}$), 3) the occupancy vectors that shows the information of the inputs, from which cells are sent and stored (i.e. $OV_{i,j} = 0$ represents there is a cell from input i currently stored in output j, otherwise '1'), 4) the B vector showing the number of the available internal buffers in output j, and 5) an one-hot shift register that records the highest priority position. The grant scheduler has two data paths that calculate the grant decisions to non-admitted requesting ports and the decisions to admitted ports in parallel. In the data path that forms the grant decision to admitted flows, the admission vector is ANDed with the occupancy vector to ensure that only the admitted flow, with no cell currently stored inside the internal buffers, is left, filtering out those flows that are admitted but already having a cell within the port. According the algorithm, the admitted flows remained from the previous step receive service definitely from the subscribed port. Hence, the result from the previously described AND operation directly forms the grant decision to the admitted flows. Then the number of 1s in the admitted grant decision is calculated and subtracted from the B vector to form the remaining credits available for non-admitted flows. Using the grant scheduler design of the DROP algorithm as a building block, the remaining credits, the request vector and the value from the one-hot shift register are fed into the DROP grant scheduler function block to generate the grant decisions for the non-admitted flows (we refer the reader to [18] for the detailed design of a DROP grant scheduler). Final to combine the grant decisions from these two data paths, an OR operation is performed. Finally, a selection between the pure admitted decision and combined decision is done by a multiplexer using the least significant bit (LSB) of the remained credits. We believe the ELSRR grant scheduler can operate at a comparable speed to that described in [18], as the DROP grant scheduler dominates the critical path.

Fig. 3.13 shows the design of the ELSRR input scheduler, under the same assumption of a 32x32 PBC switch with $B = 2$. The inputs to the input scheduler circuits consist of the following items, the admission queue (AQ) that records the VOQ being admitted, the grant queue (GQ) that provides information on received credits and the input buffer table (IBT) [18] that records the number of cells stored in each of the VOQs. We assume each entry of the IBT is 32-bit wide, corresponding to 4096 Mega cells storage capacity per VOQ. Similar to the design of grant scheduler, there are two data paths operating in parallel. One is used to compute the longest queue index among admitted flows, while the other calculates the longest queue index among the non-admitted flows. The circuit works as follows, the GQ vector is ANDed with AQ and the inverted AQ to separately obtain the received grant credits to admitted and non-admitted flows. The extracted grant vectors are then used to select out the length of queues belonging to admitted flows and non-admitted flows in parallel, and feed the length to the maximum index function to sort out the longest ones of both parts. The maximum index function is built from multiple stages (5 stages here) of smaller comparison blocks that includes a carry propagation adder (CPA), with which an addition is carried out between operand A and inverted operand B, if later the carry bit of CPA becomes '1' then operand B $<$ operand A (details described in [35]). To select the final index to the selected VOQ, we perform a bitwise OR operation to the result of AND operation between AQ and CQ, figuring out whether there are admitted queues granted by outputs. If true, then the index to the longest queue from admitted flows will be selected; otherwise the longest

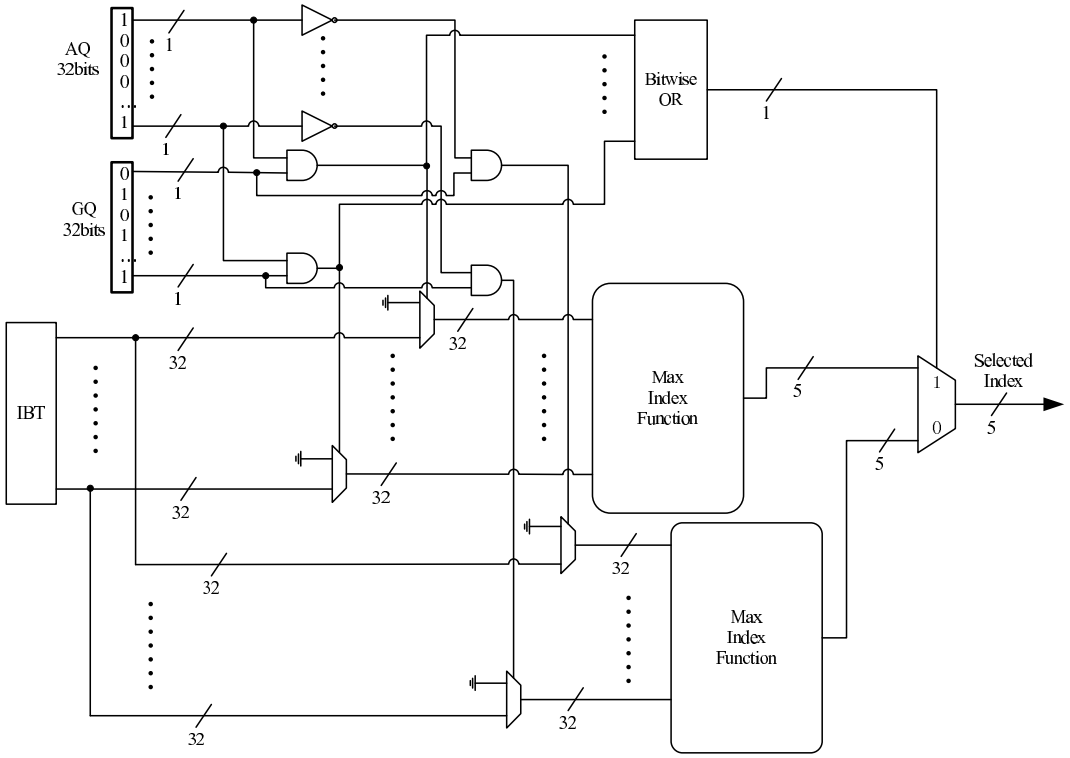

Figure 3.13: Hardware Designed of the ELSRR Input Scheduler.

queue index of non-admitted queues will be used.

3.6 Summary

The PBC switching architecture has shown a high potential in being the architecture of choice for next-generation routers. However, the previously proposed algorithms for the PBC did not take full advantage of its features. In this chapter, we proposed a novel-scheduling algorithm (named the ELSRR) for the PBC, which is able to deliver high-performance. In particular, employing the ELSRR algorithm, a PBC switch: *i*) can deliver low cell latency and high throughput under both uniform as well as non-uniform traffic patterns, while maintaining a very small internal buffer size ($B = 2$) per output; *ii*) can optimally manage the internal buffers (and bandwidth) share amongst competing flows; and *iii*) has a simple round-robin based output scheduling as opposed to the previously required FCFS policy. Extensive simulations have shown that the ELSRR outperforms previous proposals under various traffic inputs.

Scheduling Multicast Traffic in PBC Switches

4

The tremendous increase of multicast traffic applications on the Internet has led to intensive studies to seek an efficient way to support multicast traffic in Internet routers and switches. Various solutions have been proposed for the unbuffered (IQ) and buffered crossbar (CICQ) crossbar switches. However, they were either too complex to run at high speed or too expensive to be readily implemented in hardware. The Partially Buffered Crossbar (PBC) has been proposed and shown to achieve the performance of a buffered crossbars while having a low cost close to that of unbuffered crossbars under unicast traffic. This chapter conducts the first study on multicast traffic support in PBC switches and proposes a round-robin based scheduling algorithm termed MSRR. Through simulations, we show that with the proposed algorithm, a PBC switch outperforms its unbuffered and buffered counterparts with a small number of internal buffers, making it highly attractive for next generation networks

4.1 Introduction

Traditionally, Internet routers were designed for point-to-point (unicast) communication only. However, due to the rapid development of the internet application, such as IPTV, video conference and voice-over-IP (VoIP), there is an increasing demand for high-speed routers to support point-to-multipoint (multicast) communication. As a result, different architectures have been investigated and implemented to effectively support multicast traffic [52][53][54].

The crossbar switching architecture has been widely considered the most suitable switching architecture due to its low-cost, scalability and intrinsic multicast support. It has two variations: unbuffered crossbar (IQ) and buffered crossbar (CICQ). Abundant research have been conducted on the unbuffered crossbar [43] as well as buffered crossbar [45] architecture with a single multicast FIFO queue per input. As with the unicast traffic, the single FIFO strategy suffers from the HoL blocking issue and limits the potential throughput of a switch. Avoiding the HoL problem mandates using up to $(2^N - 1)$ FIFO queues per input (N is the switch port count), known as the Multicast-Virtual Output Queuing (MC-VOQ) architecture. The MC-VOQ architecture is clearly impractical for medium to large switching systems. As a result, researchers have proposed to use a small number of queues, $K(1 \leq K \ll 2^N - 1)$ per input [41] [40] to improve the performance of a switch. As for the switching fabric interconnect, the IQ is attractive due to its low cost; however it suffers low performance due to its centralized scheduling. On the other hand, buffered crossbar switches exhibit high-performance with simple and distributed scheduling but comes at the price of N^2 cross point buffers that grows quadratically with the switch size. Hence, both architectures are less appealing.

In order to achieve high performance while maintaining a cost close to an unbuffered

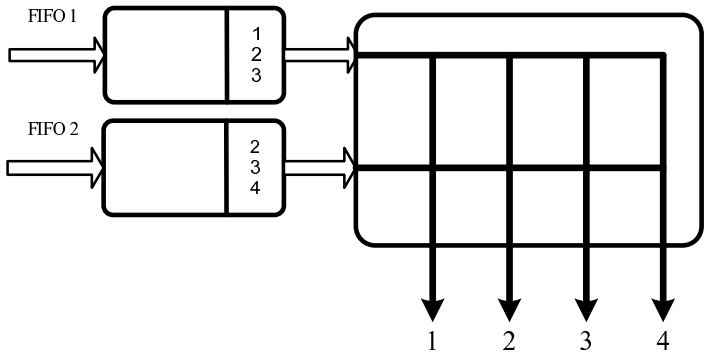

Figure 4.1: A 2x4 unbuffered crossbar switch with multicast FIFO queue.

switch, the partially buffered crossbar switching architecture (PBC) has been proposed [18]. Different from a buffered crossbar switch, a PBC switch maintains B ($B \ll N$) shared internal buffers at each fabric output. This design reduces the growth of internal buffers to a linear function of the switch size N and hence a significantly lower cost compared to the buffered crossbar architecture. The scheduling process in the PBC architecture is a fusion of that in buffered and unbuffered crossbar switches: distributed schedulers located at each input and output (as in buffered crossbar) and a request-grant-accept (RGA) input scheduling that maps requesting inputs into the shared internal buffers (as in unbuffered crossbar). In [18], it is shown that with a small number of internal buffers per output, the performance of a PBC switch converges to an Output Queued (OQ) switch under i.i.d arrival traffic. Based on the observation to the unicast performance, we expect a PBC switch to exhibit desirable performance under multicast traffic.

In this chapter, we study the multicast capability of a PBC switch and propose a round-robin scheduling algorithm. It is simple, yet capable of outperforming the existing algorithms from buffered as well as the unbuffered crossbars. The proposed algorithm is flexible to be used under single multicast FIFO architecture as well as the K-FIFO architecture. Furthermore, we explore the trade-off between the number of multicast queues K and internal buffers per output B.

This chapter is structured as follows: in Section 4.2 we present background knowledge and related work, Section 4.3 introduces the multicast PBC switching architecture and the proposed MSRR algorithm, Section 4.4 demonstrates an experimental study, Section 4.5 presents the hardware design of proposed algorithm. Finally, Section 4.6 summarizes this chapter.

4.2 Background

Switch supporting multicast traffic is essentially capable of sending multiple copies of a same cell to several destinations. Since the crossbar based switching architecture supports the copy of a cell and multiple-destination transaction intrinsically, intensive research has been done on unbuffered (IQ) and buffered (CICQ) crossbar switches.

4.2.1 Queuing Schemes and Cell Placement Policy

Different from an unicast cell, a multicast cell can have a number of destinations ranging from 1 to N, known as the fan-out set of a cell. The queuing scheme for multicast traffic has a significant impact on the performance of a switch. As a result, different queuing structure has been proposed in attempt to effectively buffer multicast cells. Most of the studies have been based on a single multicast FIFO architecture, where only one queue is maintained at each input port. While being the most practical buffering scheme, it suffers from HoL issue and provides a poor performance. To address the HoL problem, the best solution is known as multicast VOQ (MC-VOQ), where $2^N - 1$ separate FIFOs are kept at each input port. With MC-VOQ, the HoL problem is completely eliminated. However, due the large number of queues required by this scheme, it has a high cost and scheduling complexity that prevent it from being practically applied to switches even with small number of ports. In [55], a window-based scheme were proposed allowing multiple cells at the head of a queue being considered for scheduling. However, it requires a complicated hardware implementation. The K-FIFO architecture has been proposed and studied under unbuffered and buffered crossbar context [41] [56] [42], where a small number, k of multicast queues are maintained at each input. While being a practical solution, it greatly improved the switch performance comparing to that of using only one FIFO per input. As the number of multicast queues, k is much smaller than the total cardinality of fanout set, a mapping from $2^N - 1$ to k has to be handled. This mapping is known as a cell placement policy [41] [40] [42] [57]. A good cell assignment policy should have the following properties: *i*) HoL cells should contain diverse fanout set that can span over a large number of output ports for witch the input holds packet. *ii*) Cells with the same or similar fanout sets should be placed in the same multicast queue. This reduces the HoL issue and avoids the out of sequence delivery.

4.2.2 Service and Scheduling Disciplines

Scheduling algorithms supporting multicast traffic have also been studied and proposed. Since the crossbar fabric operates at the same speed as the external links, during one time slot an input can send at most one cell and each output can receive only one cell. As crossbar switching architectures have natural property to support multicast traffic. By closing multiple cross point of the fabric, a multicast cell transaction can be completed within a single time slot. However, as a usual case, a cell may not have access to its entire destination in one time slot. Hence, service disciplines known as no fanout splitting and fanout splitting have been proposed [43]. When the no fanout splitting discipline is applied, a cell can only traverse the fabric once, which means the cell will not be scheduled for a fabric switching if not its entire destination is available.

When the fanout splitting service is used, a cell can be partially transferred to its destinations. Copies that lose contention to destinations in current time will continue competing for their destinations in future time slots. Hence, the completion of transaction may spread over multiple time slots. The flexibility of partly switching a cell results in a higher switch throughput because it is work conserving [58]. Consider the 2x4 unbuffered crossbar switch with one FIFO per input shown in Fig. 4.1. The fanout of HoL cells from the two input covers all the output ports. Input 1 is requesting for output

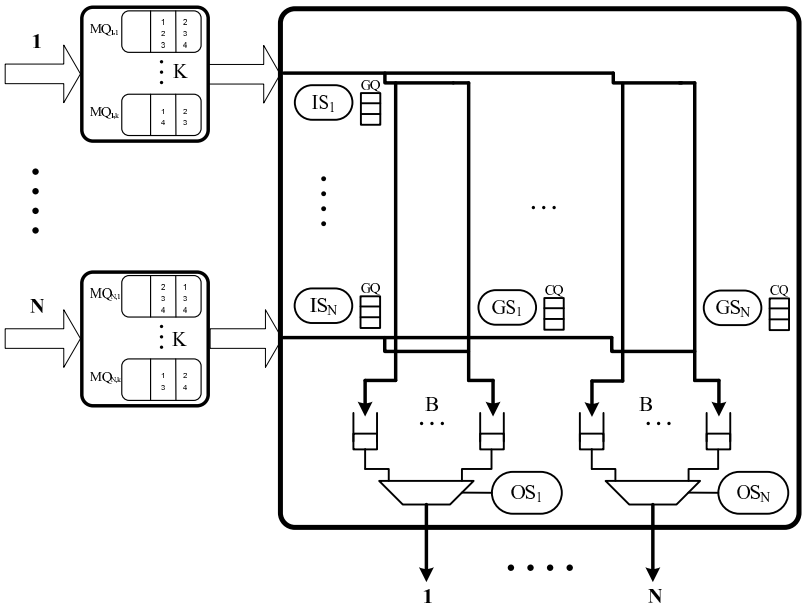

Figure 4.2: The multicast PBC switching architecture

{1, 2, 3} while input 2 is requesting for output {2, 3, 4}. As both cell wishes to send a copy to output 2, 3, if no fanout splitting discipline is applied, then only one of the cell will be discharged in this time slot resulting in 75% throughput. On the other hand, the switch achieves 100% throughput with fanout splitting enabled, since it allows a cell to be partially discharged. The remaining destinations of cells are defined as the residue, in the example the residue is {2, 3}. Depending on the scheduling policy, the residue can be either concentrated on a smallest number of input ports or spread over a largest number of input ports. As suggested by [43], adopting a scheduling algorithm that concentrates the residue provides a better performance than algorithm that distributes the residue over multiple inputs. For instance, for a scheduling algorithm that adopts the concentration strategy, one of the HoL cell from FIFO 1 or FIFO 2 will be completely discharged leaving another cell bearing the residue of {2, 3}. Otherwise, none of the two HoL will be completely discharged and each bears one of the destinations from the residue set.

4.3 The multicast PBC architecture and its scheduling

We consider the PBC switch model depicted in Fig. 4.2. Upon arrival at a switch, packets are segmented into fixed-length cells and switched by the switch; later cells are reassembled back to packets when departing the switch. We consider the fanout splitting service discipline when scheduling the multicast traffic.

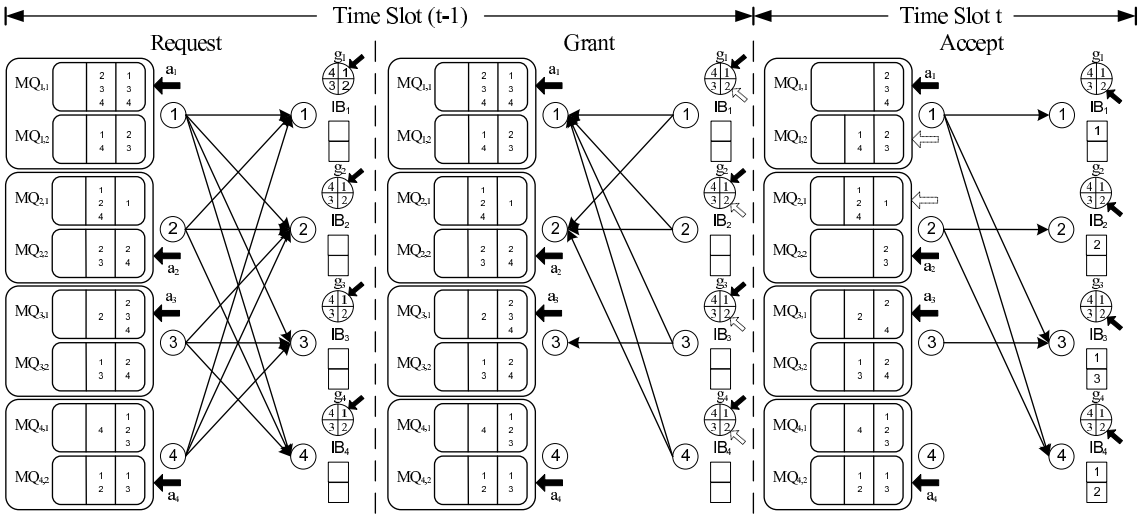

Figure 4.3: An MSRR input scheduling cycle for a 4x4 partially buffered crossbar switch with $B = 2$ and $K = 2$. The dotted arrow shows the future position of a pointer in a following time slot.

4.3.1 The Multicast PBC Architecture

The proposed switching architecture has N inputs and outputs, each output keeps B ($B \ll N$) shared internal buffers. Similar to the buffered crossbar, there are N input schedulers (IS) and output schedulers (OS) located at each fabric input and output, operating separately and in parallel. Additionally, there are N grant schedulers (GS) deployed inside the fabric, managing the availability of the internal buffers (GQ) and mapping B out of N requesting ports into the internal buffers. Through the credit queue (CQ), an input scheduler is notified about whether it receives any credits.

Scheduling process in multicast PBC architecture is split into two phases: input scheduling phase and output scheduling phases. During the input-scheduling phase, depending on the multicast FIFO architecture adopted, an input scheduler (IS) may request for one or more HoL cells and the requests are seen by the grant schedulers (GS). Subject to the availability of internal buffers (CQ) of an output, a grant scheduler may issue grants to no more than B inputs. This relaxes the output contention and allows multiple inputs send copies of HoL cells to the same output within one time slot. After receiving grants, the input scheduler will transmit a selected HoL cell to its all desired destinations from which credits have been received. Copying a multicast cell to multiple destinations is natively supported by crossbar-based architecture, and it is achieved by simply closing multiple cross point of the crossbar fabric. During the output-scheduling phase, output schedulers, located at each fabric output, select a non-empty internal buffer on a first-come-first-serve (FCFS) basis and transfer the cell outside the switch. Adopting a FCFS policy at output schedulers ensures the in-sequence delivery of cells.

In the following sections, we present a round-robin based multicast scheduling al-

gorithm for PBC architecture, and conduct the first study on the capability of a PBC switch supporting multicast traffic comparing it to the existing crossbar based switching architectures.

4.3.2 The MSRR Algorithm

The MSRR (Multicast-Static-Round-Robin) is a fair and flexible algorithm that can provide hard bound on a waiting time experienced by a packet and can function with both multicast FIFO and K-FIFO queuing architecture. The MSRR uses a fully synchronized pointer [45] in the grant schedulers that increment regularly in every time slots. This feature ensures that each input can have a fair chance to discharge a complete cell. On the other hands, each input scheduler keeps an accept pointer that is used to decide which multicast queue should be prioritized to discharge a cell. An accept pointer is updated only when a multicast FIFO discharges a complete cell, which essentially guarantees that all queues within an input have a fair chance to completely discharge its HoL cell. After each accept stage, the remaining credits received by an input scheduler will be discarded to eliminate the credit release delay that otherwise suffered by grant schedulers[18]. The specification of algorithm is as following:

Algorithm 1: MSRR

Each input scheduler, i, considers all its HoL cell(s) and sends requests to outputs.

Grant Phase:

All the grant pointers g_j are, arbitrarily, set to the same initial position and incremented, in each time slot, by 1 *mod* N.

For each grant scheduler j, do:

- Set CQ_j equal to the number of available internal buffers.
- While there are credits in CQ_j, do:
  - Starting from g_j index, send a grant to the first input, i, that requested this output (set $GQ_{i,j} = 1$)
  - Decrement CQ_j by 1.

Input Scheduling Phase:

For each input scheduler i, do:

- Starting from a_i index, select the first non-empty multicast queue k, and transmit its copies to internal buffers of outputs, j that granted input i (with $GQ_{i,j} = 1$).
- If the HoL cell of selected queue k is completely discharged,
  - Update a_i to $(k + 1)$ *mod* K.
- Drop credits that are not accepted (Set $GQ_{i,*} = 0$).

Fig. 4.3 is an graphic demonstration of a MSRR input scheduling cycle for a 4x4 PBC switch with two multicast queues per input and two internal buffers (IB) per output.

The input scheduling cycle is pipelined into two stages, the request and grant stage that takes place in time slot $(t-1)$ and the accept stage happens in time slot t. Suppose that at time slot $(t-1)$ all internal buffers are available. In the request stage, each input scheduler considers HoL cells from all its non-empty multicast queues and sends up to N requests to the the grant schedulers located at each output. Since the pointers of grant schedulers are synchronized to input 1, it receives full service from all the port it is requesting. The remaining credits of each grant scheduler may be spent on other input that are requesting. In this case, beside input1, input 2 also receives grants from all the output it is requesting and hence able to discharge a complete cell, while input 3 receives only one credit from output 3.

In the successive time slot, each input scheduler select one multicast queue based on the position of its accept pointer and transmit copies of HoL cell to the internal buffers of granted outputs. Due to the small number of internal buffers introduced to each output, a PBC switch can discharge up B HoL cells completely in one time slot. As shown in the figure, both input 1 and 2 completely discharged one cell from a multicast queue and the accept pointer is incremented by one beyond the served queue, while input 3 only discharged partly of a HoL cell and the accept pointer is not updated. The round-robin mechanism of MSRR allows it to be fair and starvation free while maintaining the simplicity hardware implementation. In the worst case, the latency experienced by a HoL cell will be no longer than $(N \times K + B - 1)$ time slots, where N is the number of output ports, K is the number of multicast FIFOs kept within an input and B being the number of internal buffers at each fabric output.

4.4 Experimental Result

In this section, we present the performance of MSRR with different queueing and switching configuration. The performance results are collected after 1,000,000 time slots and from different sizes of the switch, including 8x8, 16x16 and 32x32. The mean fanout size of a multicast cell is $N/2$ throughout the simulation, where N is the number of ports of a switch.

We conduct the performance analysis under two traffic scenarios: Bernoulli uniform (uncorrelated) traffic and Bursty uniform (correlated) traffic, and experimental results are arranged into three parts. In the first part, we study the behavior of MSRR algorithm with a single multicast FIFO per input, and compare its performance to TATRA [43] and a multicast SLIP-like round-robin algorithm [26], denoted as MSLIP, from the unbuffered crossbar architecture, and to MXRR [45] from the buffered crossbar architecture. We consider the buffered crossbar has only one buffer at each crosspoint of the fabric, i.e. N^2 total number of internal buffers. In the second part, we study the performance of MSRR with multiple multicast queues per input; the result is again compared to the MXRR algorithm from buffered crossbar architecture. In the third part, we focus on MSRR and the PBC switching architecture, illustrating the switch performance as a function to the number of multicast queues, K and number of internal buffers per output, B. For the performance experiments involving multiple multicast FIFOs per input, the cell assignment policy in [42] is applied.

Fig. 4.4 shows the average latency of TATRA, MSLIP and MXRR comparing to

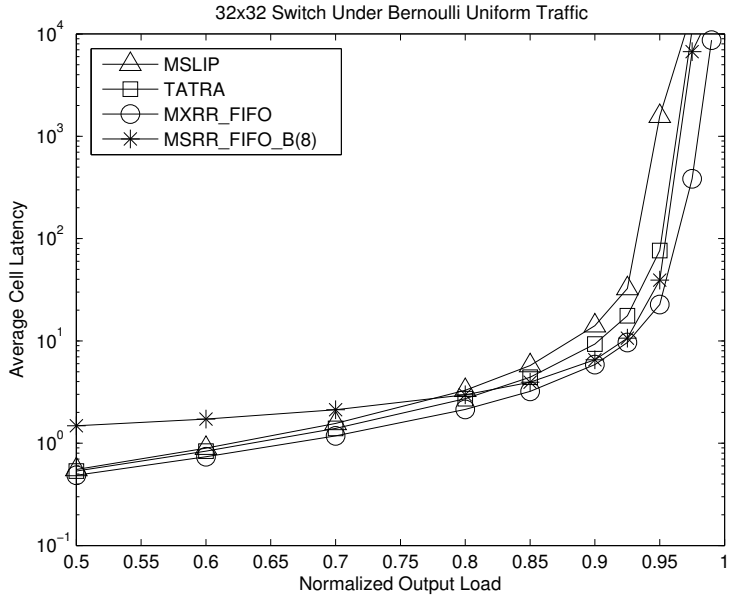

Figure 4.4: Average cell latency of switch with one multicast FIFO per input under bernoulli uniform traffic.

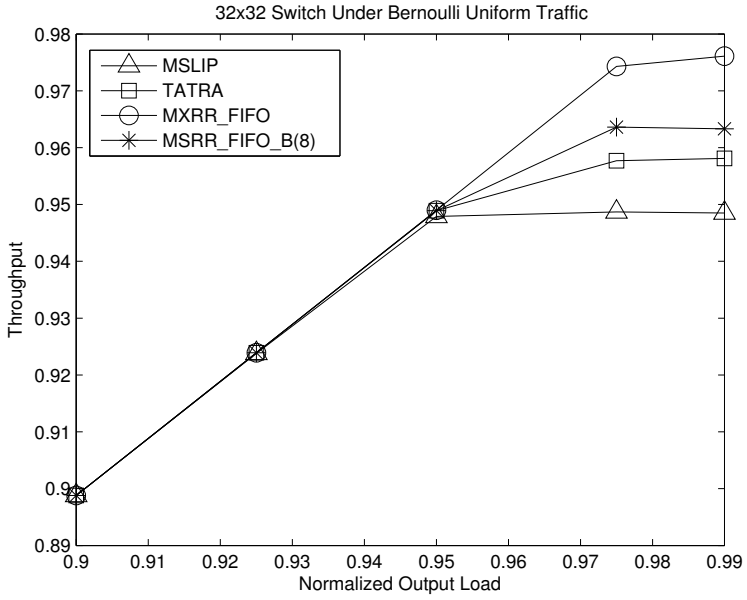

Figure 4.5: Throughput of switch with one multicast FIFO per input under bernoulli uniform traffic.

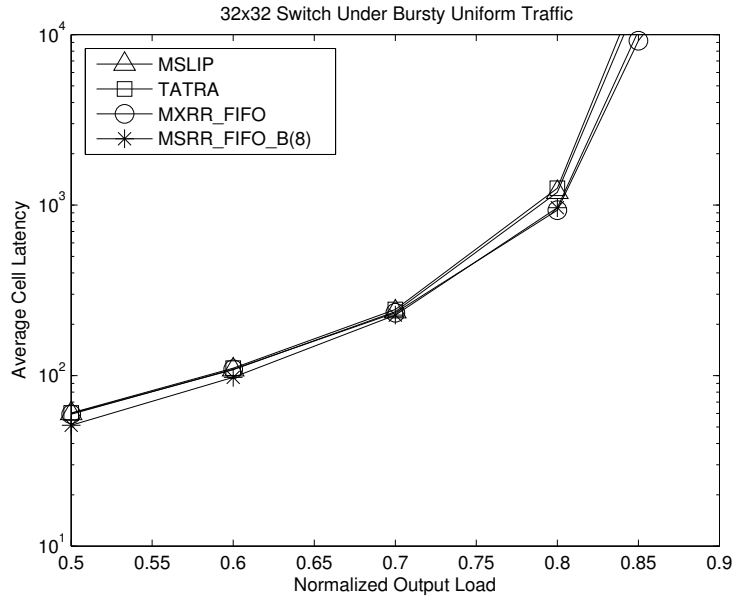

Figure 4.6: Average cell latency of switch with one multicast FIFO per input under bursty uniform traffic.

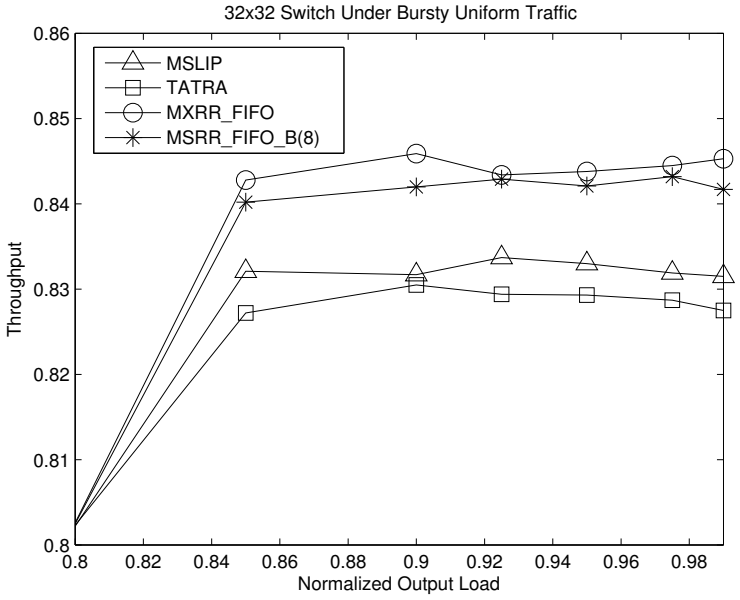

Figure 4.7: Thoughput of switch with one multicast FIFO per input under bursty uniform traffic.

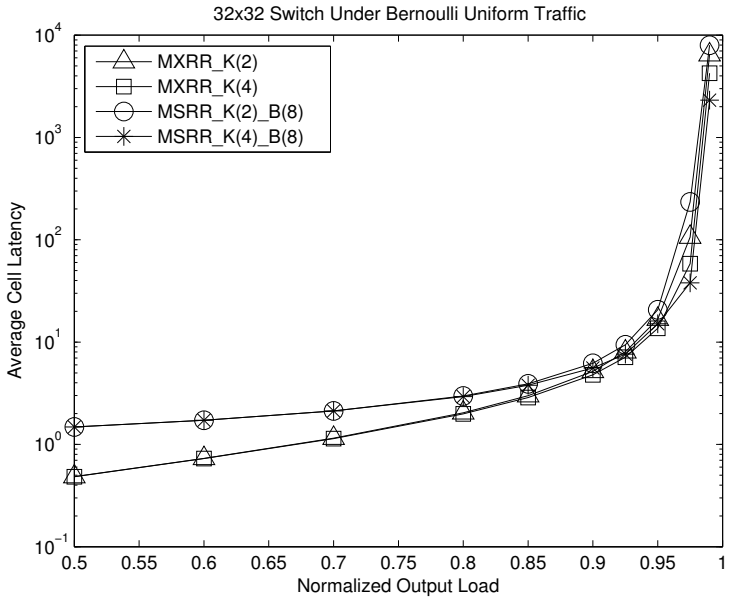

Figure 4.8: Average cell latency of switch with multiple multicast FIFOs per input under bernoulli uniform traffic.

MSRR with eight internal buffers per output under Bernoulli uniform traffic. The size of switch is 32x32 and each input has only one multicast FIFO. Note that the higher latency experienced by MSRR at low arrival rate is due to the pipelining design of a PBC switch. As the traffic load increase, MSRR outperforms TATRA, MSLIP and maintains a performance close to MXRR. The introduction of internal buffers to PBC switch allows more than one HoL cells to be discharged in one time slot, which directly translates into a lower latency. We also evaluate the performance from the aspect of switch throughput as shown in Fig. 4.5. MSRR achieves a higher throughput than TATRA and MSLIP, however the achieved throughput is 1% less than MXRR. If we take into the hardware factors into the consideration, i.e. the simple multicast FIFO queuing scheme and the small number of internal buffers maintained per output, it is a fair trade-off between performance and hardware cost.

With identical switching and queuing configuration, the same trend can be observed under Bursty uniform traffic with average burst length of 16. As shown in Fig. 4.6 and Fig. 4.7, MSRR has undistinguishable performance compare to MXRR, while having a lower latency and higher throughput than TATRA and MSLIP.

Fig. 4.8 and Fig. 4.9 illustrates performance of 32x32 switch with multiple multicast FIFOs available at each input port. Here, we only consider two and four multicast FIFOs per input, for the reason that keeping large number of multicast FIFOs is conventionally undesirable and leads to an increased latency of the cell placement circuit [42]. In Fig. 4.8, behavior of the MSRR under Bernoulli uniform traffic is studied and compared to that of MXRR. We can observe that with two multicast queues per input, the MSRR has a

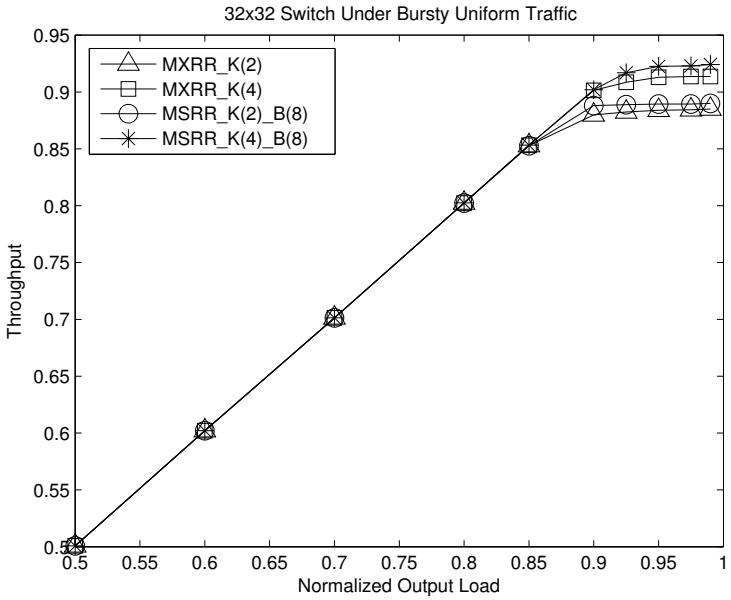

Figure 4.9: Throughput of switch with multiple multicast FIFOs per input under bursty uniform traffic.

slightly higher average latency than MXRR. However, when the number of FIFOs per input increased to four, the MSRR results in a better performance. With an increased number of multicast queues, the MSRR also yields more performance improvement than MXRR. Furthermore, the superior performance is achieved with 768 less internal buffers comparing to a buffered crossbar switch and a affordable number of multicast queues at each input.

The throughput statistics under Bursty uniform traffic shown in Fig. 4.9 indicates that while an increased number of multicast queues improves the performance of the MXRR and the MSRR, the latter achieves a higher throughput than the MXRR with a same number of queues per input.

In the reminder of this section, we study the effect of varying the number of multicast queues per input and the size of internal buffers output with different size of switches. We denote the number of multicast queue as K, and the number of internal buffers at each output as B. Fig. 4.10 and Fig. 4.11 shows the MSRR under Bernoulli uniform traffic.

In Fig. 4.10 shows the average cell latency of three different size of switches with different number of multicast queues and internal buffers. We can observe that irrespective the size of switch, having more number of internal buffers per output will generally lead to a lower latency than having more number of multicast queues per input. To confirm the observation, in Fig. 4.11 we observer the throughput of switches with different queuing and internal buffering configuration. As can be seen from the figure, a 32x32 switch with two queues per input and eight internal buffers per output achieves 98%

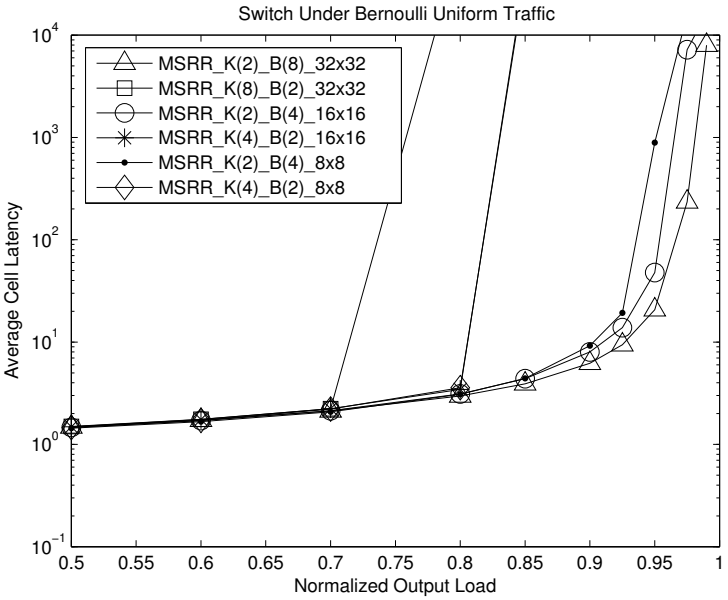

Figure 4.10: Average Cell Latency of the MSRR with different MQ numbers, K and internal buffers, B, under bernoulli uniform traffic.

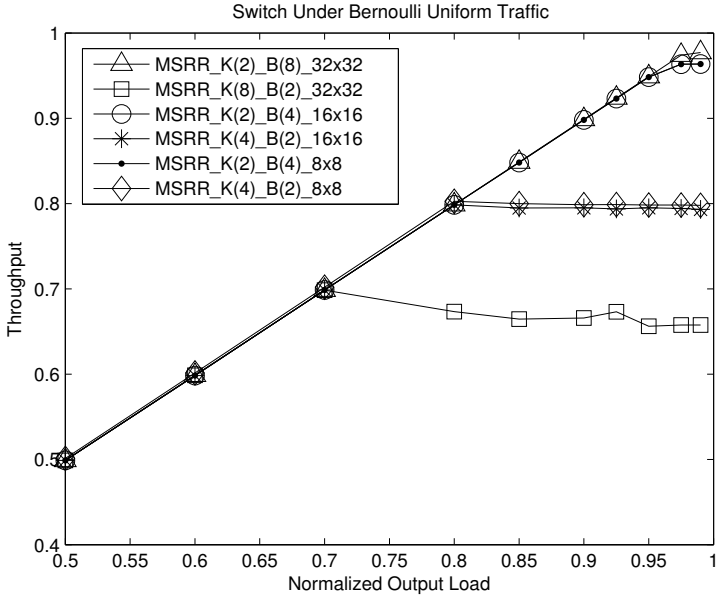

Figure 4.11: Throughput of the MSRR with different MQ numbers, K and internal buffers, B, under bernoulli uniform traffic.

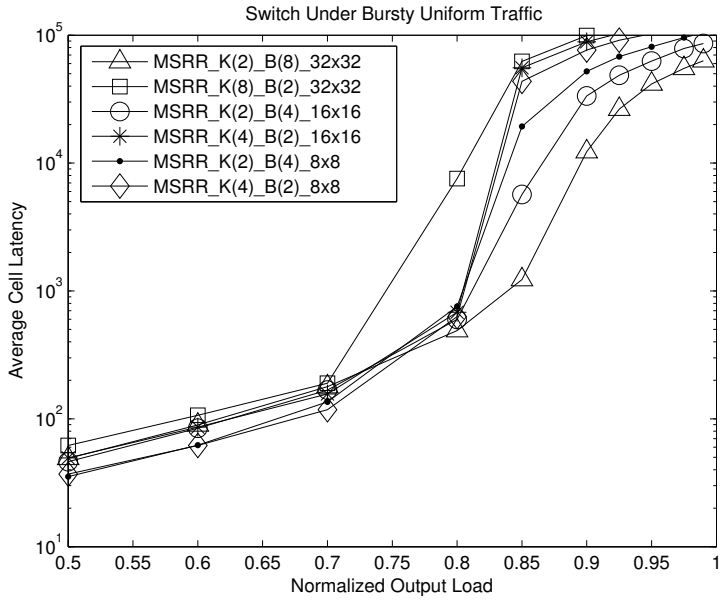

Figure 4.12: Average Cell Latency of the MSRR with different MQ numbers, K and internal buffers, B, under bursty uniform traffic.

throughput at maximum traffic arrival rate, while the switch in the same size with eight queues per input and two internal buffers per output only achieves a throughput of less than 70%. This observation is within our expectation, since having more internal buffers per output enables the switch to remove more HoL cell completely in one time slot. On the other hand, a large number of multicast queues will only provide a increase of the fanout diversity available to the grant scheduler, which does not directly contribute to maximum number of cells that can be completed removed/served from the HoL position.

Fig. 4.12 and Fig. 4.13 conducts the same study under Bursty uniform traffic. The same trend can be observed as switches with larger number of internal buffers achieve a lower latency and higher throughput comparing to switches with larger number of multicast queues. The result implies that when designing a multicast PBC switch, deploying more internal buffers at each output and a small number of multicast queues per input will result in an optimal performance. Also, keeping less number of multicast queues helps simplify the hardware design of the MSRR scheduler and allows the multicast cell placement operating in a higher speed.

4.5 Hardware Design

In this section, we present the hardware design of the MSRR input scheduler. Assuming a 32x32 multicast PBC switch with k multicast FIFOs per inputs. Since the design of the input scheduler is independent of the number of internal buffers, we do not impose any specification on the parameter B here. Fig. 4.14 presents the schematic diagram of

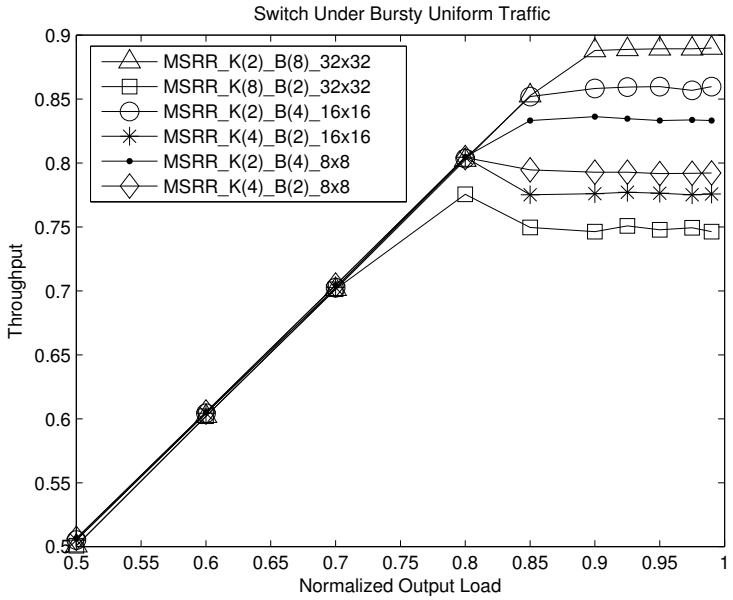

Figure 4.13: Throughput of the MSRR with different MQ numbers, K and internal buffers, B, under bursty uniform traffic.

the proposed design. The input scheduler circuit consists of the following components, the input buffer table (IBT) which keeps track of the number of cells in each of the multicast FIFOs belonging to an input, the multicast queue fanout (MQF) vectors that represent the fanout of HoL cells, a programmable priority encoder (PPE) implementing the round-robin prioritization, the grant queue (GQ) that holds the granted credits from outputs, a register that holds the highest priority position, and different arithmetic units. The circuit functions as follows: the bitwise OR of each IBT entry is used to express whether a multicast queue is non-empty and requesting for service, generating the k bits signals that are fed into the PPE. Depending on the current highest priority, an index to the maintained k multicast FIFO is computed. With the computed index, we can select one set of fanout among the k fanout vectors. The selected vector is then ANDed with the GQ vector to produce the destinations that the selected HoL cell will be copied to. In order to determine the completion of a cell discharge, we compare the number of the destinations of the selected HoL cells to the number of granted credits. If there are more credits than the destinations, the selected HoL cell is considered as fully discharged and the PPE gets an update on the highest priority position.

4.6 Summary

The PBC switching architecture has shown a high potential in being the architecture of choice for next-generation routers under the unicast traffic. However, its potential under multicast traffic has not yet been studied. In this chapter, we conduct the first study on

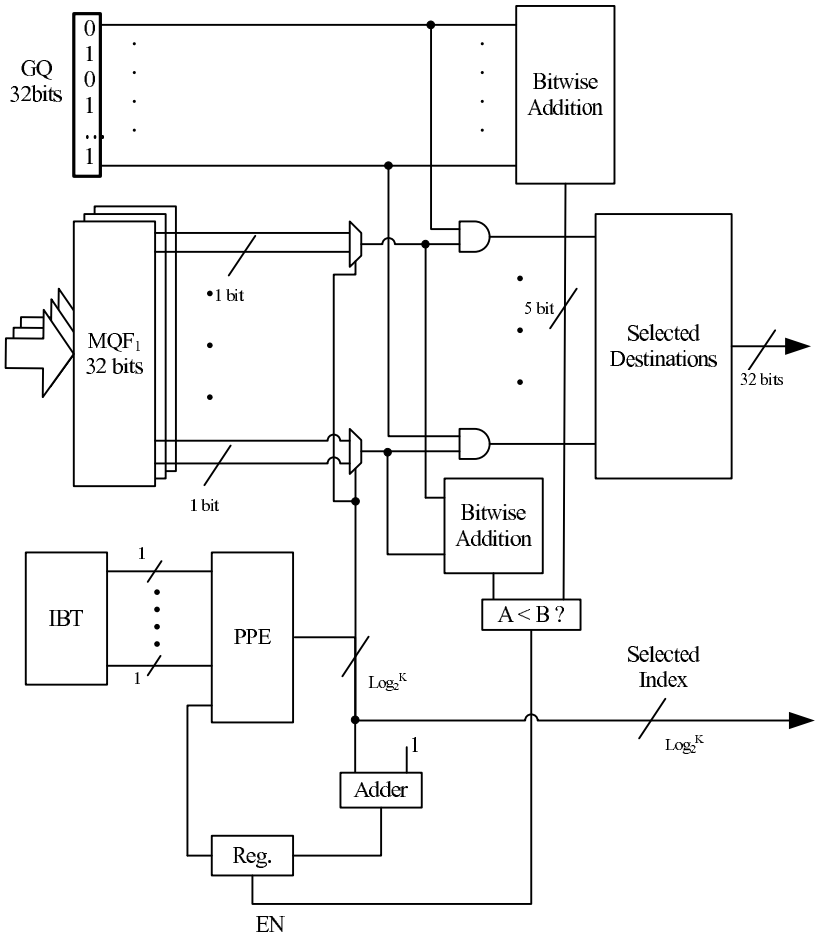

Figure 4.14: Block diagram of the MSRR input scheduler.

multicast capability of a PBC switch, and proposed multicast scheduling algorithm for PBC architecture: MSRR. Through experiments, we show that a PBC switch is capable of outperforming the unbuffered crossbar switch as well as the buffered crossbar switch, adopting a simple yet practical distributed round-robin scheduling algorithm and using up to 75% less of expensive internal buffers.

5 Conclusion

The PBC switch has been considered an appealing candidate for next-generation switching architectures. It has distributed input and output schedulers that overcome the scheduling complexity bottleneck faced by the IQ switch, and it has significantly lower hardware cost than the CICQ switch to deliver satisfactory performance. This thesis studies the unicast-scheduling problem for the PBC switch to further boost its performance under non-uniform traffic, while maintaining fewer internal buffers per output. Multicast support by the PBC switch is also studied in the thesis and an appropriate algorithm is proposed. This chapter is organized as follows: in Section 5.1, we summarize the content presented in this thesis, in Section 5.2, we conclude the major contributions to the scheduling in the PBC switch. Finally, in Section 5.3, we provide a list of open issues to be studied in the future regarding the PBC architecture.

5.1 Summaries

We begin with Chapter 2, a survey of existing crossbar based switching architectures and scheduling algorithms. By presenting the advantages as well as the shortcomings, we showed the evolution from the shared-memory switch to the CICQ switch, and eventually to the PBC switch. The thesis aims at improving the non-uniform performance of the PBC switch and explores the multicasting support of the PBC switch.

With the aforementioned motivation, we proceed to Chapter 3, in which the ELSRR algorithm is proposed to improve the performance of a PBC switch, especially aiming at achieving a higher throughput under unbalanced traffic. Based on observation from the previously proposed round-robin algorithms, the ELSRR algorithm is designed to be capable of identifying the congested queue, while maintaining fairness in the bandwidth allocation. Through extensive simulations, we show that the ELSRR can achieve high throughput under a collection of unbalanced traffic scenarios, and achieve high stability in a range of different switch sizes, including 16x16, 32x32 and 64x64 and two internal buffers per output.

In Chapter 4, we study the multicast capability of the PBC switch, in particular we: *i*) proposed the multicast PBC switching architecture, *ii*) proposed the simple yet effective scheduling algorithm MSRR, *iii*) experimentally investigate the performance of the proposed algorithm compared to the existing algorithms of the IQ and the CICQ architecture.

5.2 Major Contributions

The major contributions of this thesis are as follows:

- **Improving the performance of the PBC switch under non-uniform traffics:** we proposed a novel unicast scheduling algorithm that can achieve high performance under non-uniform traffic comparable to the LQF-RR of the CICQ architecture, but saving up to more than 90% of the expensive internal buffers. Also, the output-scheduling algorithm is reduced from the first-come-first serve (FCFS) policy to round-robin policy, without compromising the in-sequence delivery of cells. Extensive experiments showed that the proposed algorithm outperforms the previously proposed round-robin algorithms.

- **Study the multicast capability of a PBC switch:** we conducted an experimental study on the multicast support by the PBC switch, proposing the multicast PBC architecture and the round-robin scheduling algorithm MSRR. Particularly we address the question of whether the multicast PBC switch can provide comparable performance to the multicast CICQ switch. The experimental results suggest that with simple multicast FIFO queuing structure, the PBC switch can closely follow the performance of the CICQ switch, and can outperform the CICQ architecture when k-FIFO queuing structure is adopted.

5.3 Future Work

There is still much left unsaid with regards to the PBC architecture:

- **Hardware implementations of the proposed algorithms:** In this thesis, we provided only a high level design of the hardware circuits of the ELSRR and the MSRR algorithms. The implementation details and the operation speed of designed schedulers have not been investigated.

- **OQ emulation by a PBC switch:** The question of whether the PBC switch can be designed to behave identically to the OQ switch, has not yet been answered. Though [59] studied the performance guarantees of the PBC switch, there has not been any research on the OQ switch emulation by the PBC switch.

Bibliography

[1] P. O'Reilly, "The case for circuit switching in future wide bandwidth networks," in *IEEE International Conference on Communications, 1988. ICC '88. Digital Technology - Spanning the Universe. Conference Record.*, jun 1988, vol. 2, pp. 899 –904.

[2] L.G. Roberts, "The evolution of packet switching," *Proceedings of the IEEE*, vol. 66, no. 11, pp. 1307 – 1313, nov. 1978.

[3] H. Saito, M. Kawarasaki, and H. Yamada, "An analysis of statistical multiplexing in an ATM transport network," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 3, pp. 359 –367, apr 1991.

[4] L.G. Roberts, "Beyond moore's law: Internet growth trends," *Computer*, vol. 33, no. 1, pp. 117 –119, jan 2000.

[5] F.K. Liotopoulos, "A high-capacity, scalable video-on-demand system architecture, based on a 3-stage Clos network," in *IEEE International Performance, Computing and Communications Conference, 1999. IPCCC '99.*, feb 1999, pp. 363 –369.

[6] N. Endo, T. Kozaki, T. Ohuchi, H. Kuwahara, and S. Gohara, "Shared buffer memory switch for an ATM exchange," *IEEE Transactions on Communications*, vol. 41, no. 1, pp. 237 –245, jan 1993.

[7] F.E. El-Khamy, H.M.H. Shalaby, M. Nasr, and H.T. Mouftah, "Optical-label-based all-optical packet switching routers," in *The Fourth Workshop on Photonics and Its Application, 2004.*, may 2004, pp. 90 – 96.

[8] W. Wang, L. Dong, and W. Wolf, "A distributed switch architecture with dynamic load-balancing and parallel input-queued crossbars for terabit switch fabrics," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2002, vol. 1, pp. 352 – 361.

[9] Jian Wang and T.H. Szymanski, "Power analysis of input-queued and crosspoint-queued crossbar switches," in *Canadian Conference on Electrical and Computer Engineering, 2009. CCECE '09.*, may 2009, pp. 273 –278.

[10] Xiao Zhang, S.R. Mohanty, and L.N. Bhuyan, "Adaptive max-min fair scheduling in buffered crossbar switches without speedup," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, may 2007, pp. 454 –462.

[11] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Transactions on Communications*, vol. 35, no. 12, pp. 1347 – 1356, dec 1987.

[12] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, August 1999.

[13] Thomas E. Anderson, Susan S. Owicki, James B. Saxe, and Charles P. Thacker, "High speed switch scheduling for local area networks," *ACM Transactions on Computer Systems*, vol. 11, pp. 319–352, 1993.

[14] N McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, April 1999.

[15] Thomas E. Anderson, Susan S. Owicki, James B. Saxe, and Charles P. Thacker, "High speed switch scheduling for local area networks," *ACM Transactions on Computer Systems*, vol. 11, pp. 319–352, 1993.

[16] N McKeown, "A fast switched backplane for a gigabit switched router," *Bussiness Comm. Rev.*, vol. 27, no. 12, 1997.

[17] Robert Magill Tara Javidi and Terry Hrabik, "A high-throughput algorithm for a buffered crossbar switch fabric," in *IEEE International Conference on Communications, 2001*, 2001, vol. 5, pp. 1586–1591.

[18] L. Mhamdi, "PBC: A partially buffered crossbar packet switch," *IEEE Transactions on Computers*, vol. 50, no. 11, pp. 1568–1581, November 2009.

[19] M. De Prycker, "Impact of data communication on ATM," in *IEEE International Conference on Communications, 1989. ICC '89, BOSTONICC/89. Conference record. 'World Prosperity Through Communications'*, jun 1989, vol. 2, pp. 705 –712.

[20] V. Cerf and R. Kahn, "A protocol for packet network intercommunication," *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 637 – 648, may 1974.

[21] Song Chong, Ramesh Nagarajan, and Yung-Terng Wang, "Flow control in a high-speed bus-based ATM switching hub," in *2nd IEEE International Workshop on Broadband Switching Systems Proceedings, 1997. IEEE BSS '97*, dec 1997, pp. 137 –147.

[22] H.S. Kim, "Multinet switch: multistage ATM switch architecture with partially shared buffers," in *IEEE INFOCOM '93. Proceedings.Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future.*, 1993, vol. 2, pp. 473 –480.

[23] K. Padmanabh and R. Roy, "Expelling policies for shared memory fast packet switches with variable size packets of multiple priority," in *Workshop on High Performance Switching and Routing, 2005. HPSR.*, may 2005, pp. 332 – 335.

[24] G. Kesidis and N. McKeown, "Output-buffer ATM packet switching for integrated-services communication networks," in *IEEE International Conference on Communications, 1997. ICC 97 Montreal, 'Towards the Knowledge Millennium'*, jun 1997, vol. 3, pp. 1684 –1688.

[25] Mei-Hwey Hou and Chienhua Chen, "Service disciplines for guaranteed performance service," in *Proceedings - Fourth International Workshop on Real-Time Computing Systems and Applications*, oct 1997, pp. 244 –250.

[26] N McKeown, *"Scheduling Algorithms for Input-Queued Cell Switches"*, Ph.D. thesis, Stanford University, May 1995.

[27] B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," in *IEEE International Symposium on Information Theory.*, aug 1998, p. 165.

[28] Shang-Tse Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input output queued switch," in *Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, mar 1999, vol. 3, pp. 1169 –1178.

[29] A. Mekkittikul and N. McKeown, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *Proceedings. IEEE INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, apr 1998, vol. 2, pp. 792 –799.

[30] John E. Hopcroft and Richard M. Karp, "A $N^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite," in *12th Annual Symposium on Switching and Automata Theory, 1971.*, oct. 1971, pp. 122 –125.

[31] E. Oki, R. Rojas-Cessa, and H.J. Chao, "A pipeline-based approach for maximal-sized matching scheduling in input-buffered switches," *IEEE Communications Letters*, vol. 5, no. 6, pp. 263 –265, jun 2001.

[32] V. Tabatabaee and L. Tassiulas, "MNCM a new class of efficient scheduling algorithms for input-buffered switches with no speedup," in *IEEE Societies INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications.*, march-3 april 2003, vol. 2, pp. 1406 – 1413.

[33] S. Mneimneh, "An iterative switching algorithm with (possibly) one iteration," in *Third IEEE International Symposium on Network Computing and Applications, 2004. (NCA 2004). Proceedings.*, aug.-1 sept. 2004, pp. 223 – 231.

[34] N.T. Kung and R. Morris, "Credit-based flow control for ATM networks," *IEEE Network*, vol. 9, no. 2, pp. 40 –48, mar/apr 1995.

[35] Lotfi Mhamdi, *"Scheduling in High Performance Buffered Crossbar Switches"*, Ph.D. thesis, Delft University of Technology, Oct 2007.

[36] K. Yoshigoe, "Threshold-based exhaustive round-robin for the cicq switch with virtual crosspoint queues," in *IEEE International Conference on Communications, 2007. ICC '07.*, june 2007, pp. 6325 –6329.

[37] Cheng-Shang Chang, Yu-Hao Hsu, Jay Cheng, and Duan-Shin Lee, "A dynamic frame sizing algorithm for cicq switches with 100% throughput," in *IEEE INFOCOM 2009*, april 2009, pp. 747 –755.

[38] Zi Yun, Laixian Peng, Wendong Zhao, and Chang Tian, "RR-LQD: A novel scheduling algorithm for cicq switching fabrics," in *15th Asia-Pacific Conference on Communications, 2009. APCC 2009.*, oct. 2009, pp. 846 –849.

[39] N. Chrysos and M. Katevenis, "Scheduling in switches with small internal buffers," in *IEEE Global Telecommunications Conference, 2005. GLOBECOM '05.*, dec. 2005, vol. 1, pp. 614–619.

[40] S. Gupta and A. Aziz, "Multicast scheduling for switches with multiple input-queues," in *10th Symposium on High Performance Interconnects.*, 2002, pp. 28 – 33.

[41] A. Bianco, P. Giaccone, E. Leonardi, F. Neri, and C. Piglione, "On the number of input queues to efficiently support multicast traffic in input queued switches," in *IEEE Workshop on High Performance Switching and Routing*, 2003, pp. 111–116.

[42] L. Mhamdi, "On the integration of unicast and multicast cell scheduling in buffered crossbar switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 6, pp. 818 –830, 2009.

[43] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, pp. 855 –866, June 1997.

[44] Balaji Prabhakar and Nick McKeown, "Designing a multicast switch scheduler," in *PROC of the 33rd Annual Allerton Conference, Control and Computing*, 1995, pp. 984–993.

[45] L. Mhamdi and M. Hamdi, "Scheduling multicast traffic in internally buffered crossbar switches," in *IEEE International Conference on Communications*, 2004, vol. 2, pp. 1103 – 1107.

[46] Ying Jiang and M. Hamdi, "A fully desynchronized round-robin matching scheduler for a VOQ packet switch architecture," in *IEEE Workshop on High Performance Switching and Routing*, 2001, pp. 407 –411.

[47] Yihan Li, S.S. Panwar, and H.J. Chao, "Exhaustive service matching algorithms for input queued switches," in *HPSR. 2004 Workshop on High Performance Switching and Routing.*, 2004, pp. 253 – 258.

[48] Nabeshima M, "Performance evaluation of a combined input-and crosspoint-queued switch," *IEICE Transactions on Communications*, vol. E83-B, no. 3, pp. 737–741, March 2000.

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

[50] I. Radusinovic and Z. Veljovic, "Exhaustive limited service round robin matching algorithm for buffered crossbar switches," in *7th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services*, Sept. 2005, vol. 2, pp. 360 – 363.

[51] P. Giaccone, *Queueing and Scheduling Algorithms for High Performance Routers*, Ph.D. thesis, Politecnico Di Torino, February 2002.

[52] H.J. Chao, B.-S. Choe, J.-S. Park, and N. Uxun, "Design and implementation of Abacus switch: a scalable multicast ATM switch," in *Global Telecommunications Conference, GLOBECOM '96.*, Nov. 1996, vol. 2, pp. 854 –861.

[53] Isaac Keslassy, Shang-Tse Chuang, Kyoungsik Yu, David Miller, Mark Horowitz, Olav Solgaard, and Nick Mckeown, "Scaling internet routers using optics," *ACM SIGCOMM*, vol. 3, pp. 189–200, 2003.

[54] M.A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Multicast traffic in input-queued switches: optimal scheduling and maximum throughput," *Transactions on Networking, IEEE/ACM*, vol. 11, no. 3, pp. 465 – 477, 2003.

[55] K.J. Schultz and P.G. Gulak, "Distributed multicast contention resolution using content addressable fifos," in *IEEE International Conference on Communications, ICC.*, May 1994, vol. 3, pp. 1495 –1500.

[56] Min Song and Weiying Zhu, "Throughput analysis for multicast switches with multiple input queues," *IEEE Communications Letters.*, vol. 8, no. 7, pp. 479 – 481, 2004.

[57] Shutao Sun, Simin He, Yanfeng Zheng, and Wen Gao, "Multicast scheduling in buffered crossbar switches with multiple input queues," in *Workshop on High Performance Switching and Routing, 2005.*, May 2005, pp. 73 – 77.

[58] Joseph Y. Hui and Thomas Renner, "Queueing analysis for multicast packet switching," *IEEE Transactions on Communications*, vol. 42, pp. 723–731, 1984.

[59] Nikolaos Skalis and Lotfi Mhamdi, "Performance guarantees in partially buffered crossbar switches.," in *GLOBECOM'10*, 2010, pp. 1–6.

Simulation Environment

A

The appendix describes the simulation environment used throughout this thesis to evaluate the performance of different switching architectures. In the first section, we introduce the simulation software used to study and evaluate a particular scheduling algorithm under a particular switching architecture. In the second section, we provide definitions of different traffic models. Finally, we provide a description of the parameters used to benchmark the performance of the switch.

A.1 Simulation Tool

We use SIM, a fixed-length packet simulator, developed by computer engineering department of Stanford University. It is a time-slotted switch simulator written in ANSI-C and supports simulation of IQ crossbar switching architectures. Its software structure is shown in Fig. A.1. The traffic generator module will generate virtual cells according to a traffic model specified by the user, and the generated cells are passed to the input action module where cells are placed into the input buffers and wait to be switched. Before the switching of cells, the scheduling module will solve the contention and determine the cells that will be switched in the later step. Finally, the switching fabric module switches the scheduled cells from input queues to the output queues, where the statistics are collected and the virtual cells get destroyed. We have added the PBC switching architecture and corresponding scheduling algorithms to the simulator in order to evaluate its performance.

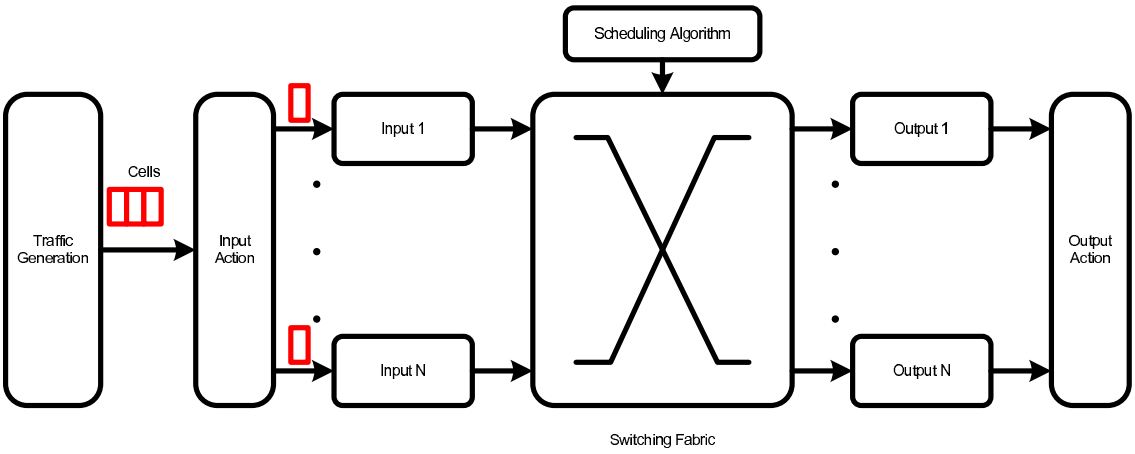

Figure A.1: The Architecture of SIM.

A.2 Traffic Models

A.2.1 Uniform Traffic Models

There are two types of uniform traffic scenario used to test the performance of a switch – bernoulli uniform and bursty uniform traffic. The bernoulli uniform is one of the most common test cases in literature studies. Cells are generated independently with a probability of ρ(also known as the normalized switch input load), and the arrival rate is spread uniformly on all the output ports N.

The bursty uniform traffic is considered an important traffic model due to its resemblance to the real Internet traffic pattern [26]. Also, the segmentation of incoming packets into cells of fixed size will also form a burst of traffic. The bursty uniform traffic is modeled by a two-state markov chain, denoted as "BUSY" and "IDLE". In the BUSY state, packets are generated only to a selected output port with a mean length of b, known as the size of burst, and no packets are generated during the IDLE state.

A.2.2 Non-Uniform Traffic Models

Traffic non-uniformity refers to the variation in the distribution of input traffic over the destination output ports. Internet traffic is, generally, non-uniform and asymmetric. Many Internet traffic examples confirm this property, such as client-server applications, where a number of clients communicate with a small number of servers. Since it is nearly impossible to simulate all such workloads, there exist some representative and commonly used non-uniform traffic models. In our simulations, we used three known non-uniform models which we describe next.

The unbalanced traffic is defined by an unbalanced coefficient ω. For an $N \times N$ switch, the traffic load at each input port is defined by ρ. Then the traffic load on $VOQ_{i,j}$, $\rho_{i,j}$ is given by:

$$\rho_{i,j} = \begin{cases} \rho(\omega + \frac{1-\omega}{N}) & \text{if } i = j, \\ \rho\frac{1-\omega}{N} & \text{otherwise.} \end{cases}$$

The double diagonal and logarithmic diagonal traffic matrix is given below for a 4×4 switch,

$$\lambda(DoubleDiagonal) = \frac{\rho}{3}\begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 1 & 0 & 0 & 2 \end{pmatrix}$$

$$\lambda(LogDiagonal) = \frac{\rho}{2^4 - 1}\begin{pmatrix} 8 & 4 & 2 & 1 \\ 1 & 8 & 4 & 2 \\ 2 & 1 & 8 & 4 \\ 4 & 2 & 1 & 8 \end{pmatrix}$$

A.3 Performance Assessment Parameters

There are commonly three metrics to evaluate performance of a switch, the average cell latency, switch throughput and the input buffer occupancy.

A.3.1 Average Cell Latency

The cell latency refers to the number of time slots a cell waited in a switch. Depending on the switching architecture, the latency experienced by a cell may consist of the time spent at the input buffer (for IQ switch), the fabric buffer (for CICQ and PBC) and the output buffer (for OQ and CIOQ). Note that the latency collected by the simulator is an average of latency over all the cells generated by the traffic generator module.

A.3.2 Throughput

The switch throughput is defined as the ratio between the output load and the input load of the switch. The maximum throughput is defined as the maxi- mum input load after which the switch becomes unstable. Instability means that the input load is higher than the throughput of the switch, hence queues will keep growing indefinitely. The maximum throughput is also known as the saturation throughput of the switch and indicates the switch capacity. If the saturation throughput of a switch with a given scheduling algorithm equals to one, which is the maximum value due for a speedup of one, then the given scheduling algorithm is said to achieve 100% throughput.

A.3.3 Input Queues Occupancies

The input queues occupancies can be used to evaluate the stability of a scheduling algorithm and a switching architecture. We use the Euclidean vector or L2 norm to express the occupancy. Let the $VOQ_{i,j}(n)$ denote the number of cells remained in the queue, then the input queues occupancies is calculated as following:

$$||L_2(n)|| = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} VOQ_{i,j}(n)^2}$$

Curriculum Vitae

Di Cao was born on 17th of May 1987 in Beijing, China. He received the Bachelor of Engineering (BEng) degree from China Agricultural University, Beijing, 2009. In the same year, he was enrolled to the Master Program of Embedded Systems at the Delft University of Technology (TUD), Delft, The Netherlands. During the 2-year study at Delft, he fought bravely for his master degree and trained by courses from a mixed background including: computer engineering, image processing and most important embedded systems. In the mean time, he collected a precious memory together with his friends of life. After the graduation from Delft University of Technology, he will look for a full-time job and start his career life in the near future.