Practical Scheduling Algorithms for High-Performance Packet Switches*

Lotfi Mhamdi and Mounir Hamdi
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
Email: lotfi@cs.ust.hk

Abstract— **As buffer-less scheduling algorithms reach their practical limitations due to higher port numbers and data rates, buffered crossbars have gained a lot of interest recently because of the great potential they have in solving the complexity and scalability issues faced by their buffer-less predecessors. In particular, the internally buffered switching architecture was shown, through distributed scheduling algorithms, to be able to sustain the current and expected increases in Internet throughput rates.**

In this paper, we propose a class of distributed scheduling algorithms for the internally buffered crossbar switching architecture. As will be shown, the distributed nature of these algorithms makes them of high practical value. That is, they can be implemented in real-time for high-speed input traffic. In addition, we will demonstrate, through simulation, that these scheduling algorithms outperform state-of-the-art related algorithms in this area.

Index terms—**scheduling, internally buffered fabric.**

I. INTRODUCTION

Input queued switches (IQ) are desirable because they are scalable and they do not require internal speed up. With the benefit of input queues, or buffers, used to store those unselected cells or packets while awaiting the appropriate output destination, the switch core has the same speed as the input ports. It is well known that, if FIFO queues are used to hold arriving packets, head-of-line (HoL) blocking problem limits the throughput to only 58,6%[1]. HoL blocking problem can be entirely overcome by maintaining separate queues, one for each output. This queuing system is called Virtual Output Queuing (VOQ) [2]. Adopting VOQ architecture, if a packet arrives at an input port, it is held in the *VOQ* corresponding to its outgoing port, avoiding the possibility of being blocked by other packets, ahead of it, corresponding to a busy output.

A plethora of algorithms has been proposed for scheduling VOQ crossbar-based switches. Maximum weight matching algorithms such as Longest Queue First (LQF) and Oldest Cell First (OCF) were proposed and have been proven to be stable for any admissible input traffic [3]. Unfortunately, the high complexity of these MWM algorithms makes them of no practical use since it prohibits the switch from scaling to large N (N is the switch size). Maximal size matching schemes were then proposed and considered as an alternative to MWM schemes. Examples include iSLIP [4], FIRM [5] and SRR [6]. These algorithms are based on the so-called request-grant-accept steps. The performance using just one iteration is usually not good enough to achieve good performance. While these algorithms yield 100% throughput under uniform traffic, they perform much less under non-uniform traffic pattern.

Buffered Crossbar Switches (BCS) have been considered as a viable alternative to buffer-less crossbar switches to improve the switching performance. While there have been many architectures for the (BCS) [7][8][9], our focus in this article is on BCS with VOQ. A buffered crossbar switch combined with input VOQ was first introduced in [10]. In the rest of this article, we will refer to this architecture as VOQ/BCS. It was proved in [11] that 100% throughput could be achieved under uniform traffic.

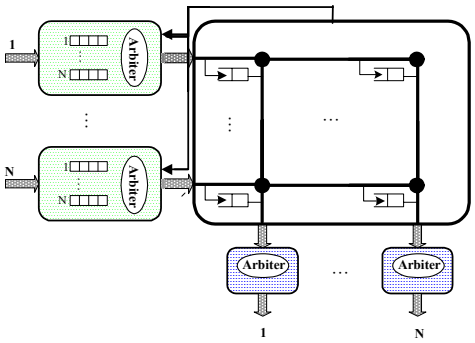

Figure 1. The VOQ/BCS Architecture.

The VOQ/BCS architecture, Fig. 1, has significant advantages over the buffer-less architecture. This is irrespective of the scheduling algorithm used for arbitration. In [10], an OCF input and output scheduling scheme was used. While this scheme achieves high throughput under uniform Bernoulli arrivals, the same benefits were not achieved for the non-uniform case. A scheme using round robin was described in [11], and was shown through simulation, to achieve very good performance. This scheme is desirable because of its simplicity, but experienced the same problem as in [10] and performs poorly under non-uniform traffic patterns. A scheme based on the LQF in the input and a round robin arbitration a the output was presented in [12] and was proven, through a fluid model, to be stable under any admissible input traffic that obeys the strong law of large

* This work was supported in part by the Hong Kong Research Grant Council (Grant Number: RGC HKUST6181-01E).

0-7803-7802-4/03/$17.00 © 2003 IEEE

numbers. In addition to its quite complex input scheduling (LQF), this scheme may lead to permanent queue starvation.

In this paper, we introduce a class of practical scheduling algorithms for the VOQ/BCS. They are named *Current Arrival First–Priority ReMoVal* (CAF-PRMV). These algorithms are, in fact, an approximation of LIFO (Last-In-First-Out). Their main advantage lies in their stateless information exchange (queue occupancies, HoL waiting time, etc.) making them simple to implement. As will be shown through simulation, they achieve very high throughput under both uniform and non-uniform distributed Bernoulli arrivals. They outperform the existing algorithms under all traffic patterns (uniform Bernoulli, bursty and non-uniform Bernoulli arrivals). In addition to the delay throughput performances, other relevant performance metrics were studied such as input VOQs occupancies. Our scheme has much less queues occupancies among all the schemes presented.

The rest of the paper will be as follows: the following section contains a presentation of the existing scheduling schemes. Then we present the group of our new CAF-PRMV algorithms in section III. The simulation results will then be given in section IV. Finally, section V concludes the paper.

II. EXISTING SCHEDULING ALGORITHMS

Recently, there has been an increasing interest in the VOQ/BCS architecture. In [11], a round robin scheme has been presented in both the input and the output scheduling. This scheme is simple in implementation, fair and achieves 100% throughput under Bernoulli traffic. The main problem with this scheme is that it doesn't perform well under non-uniform traffic patterns and can't achieve high throughput. A scheme consisting of Oldest Cell First (OCF) at both input and output scheduling was presented in [10]. Even though it is more complex than that of the round robin scheme, unfortunately the OCF-OCF performance was the same as that of RR-RR. An input scheduling algorithm based on Longest Queue First (LQF), followed by a round robin output scheme was presented in [12]. The LQF-RR scheme performs well. It was proven to be stable under any admissible input traffic that obeys the strong law of large numbers. However, as with the buffer-less crossbar case, LQF may lead to permanent starvation of nonempty queues. To see this, let's consider a 2*x*2 switch with $VOQ_{0,0} = 2$, $VOQ_{0,1} = 1$, $VOQ_{1,0} = 1$, $VOQ_{1,1} = 2$. Consider that $XP_{i,j}=0$, $\forall\, 0 \le i,j \le N-1$, and continuous arrivals occur to $VOQ_{0,0}$ and $VOQ_{1,1}$ respectively. $VOQ_{0,0}$ and $VOQ_{1,1}$ will be always selected for input scheduling and $XP_{0,0}$ and XP_{11} will be always selected for the output scheduling. Then $VOQ_{0,1}$ and $VOQ_{1,0}$ will remain unserved indefinitely. Moreover, as we mentioned earlier, the VOQ/BCS has key advantages that can serve to ensure that the scheduling algorithm can be simple and efficient at the same time. So far, both OCF and LQF schemes require sorting – to compute the oldest cell or for computing the longest queue respectively—which is undesirable.

As we will see in the next section, our proposed algorithms are simple in implementation, and don't require sorting. In fact, our algorithms are stateless and do not use any kind of state information in order to make a scheduling decision. Yet, they are capable of extremely good performance with 100% throughput under uniformly distributed as well as non-uniformly distributed traffic arrivals. They outperform all previously presented schemes under many traffic patterns.

III. THE CURRENT ARRIVAL FIRST-PRIORITY REMOVAL ALGORITHMS

In this section, we propose our group of new algorithms: *Current Arrival First-Priority Removal* (CAF-PRMV). These algorithms are an approximation of LIFO. The input scheduling gives priority to the new arriving packets while the output scheduling completes this task by serving the recently arrived packets to the internal buffer. The intuition behind this is to overcome the lack of performance under the non-uniform traffic without using any weight functions or state information. The existing algorithms either perform poorly under non-uniform traffic or require sorting. The idea of serving the newly arriving cells favors the input that has more often cells coming in and didn't punish the uniformly arriving cells, hence tackle the non-uniform traffic while being stateless. From above, we knew that an input (respectively output) scheduling scheme couldn't perform well so long as it is not matched with the appropriate output (respectively input) scheduling scheme. To this end, CAF-PRMV was designed to be a matched pair of input/output scheduling. That is the output scheduling, PRMV, is complementary to the input scheduling, CAF. To better understand this, some different schemes are presented and will be analyzed. The input scheduling, CAF, will remain the same, the changes are done at the output side. First, we give some terminology that will be used in the rest of this article.

A. Terminology

As shown in Fig.1, the VOQ/BCS switch architecture consists of N input cards, with each one maintaining N VOQs, one per output. The fabric part is the main characteristic of the VOQ/BCS and this differentiates it from the IQ/VOQ architecture. Fixed size packets, or cells, are considered. Upon their arrival to the switch, variable length packets are segmented into cells for internal processing and re-assembled before they leave the switch. A processing cycle has fixed length, called cell or time slot.

There are N input cards; each one maintains N logically separated VOQs. When a packet, destined to output j, $0 \le j \le N-1$, arrives to the input card i, $0 \le i \le N-1$, it is held in $VOQ_{i,j}$. A $VOQ_{i,j}$ is said to be eligible for being scheduled in the input scheduling process if it is not empty and the internal buffer $XP_{i,j}$ is empty (or not full).

The internal fabric consists of N^2 buffered crosspoints (XP). Each crosspoint has one-cell buffer. A crosspoint $XP_{i,j}$, holds cells coming from input i and going to output j.

The crosspoint buffer XPB_i is set of the internal buffers ($XP_{i,0} + \ldots + XP_{i,N-1}$) that corresponds to the same input, i, and holding cells for all outputs. Likewise, XPB_j is the set of the internal buffers ($XP_{0,j} + \ldots + XP_{N-1,j}$) that corresponds to the same output, j, and receiving cells from all inputs. $LXPB_j$ is the number of cells held in XPB_j.

B. Specification of CAF

At each time slot, the *Current Arrival First* (CAF) algorithm checks if there is a new cell arriving at the input port. To accomplish this task in the output scheduling, CAF assigns a priority level to each cell leaving the input port. This level will decide the priority (urgency) of that cell in the output scheduling phase. The use of priority levels is an efficient choice. First, the output-scheduling phase will be much simpler and faster than sorting for example. Second and most importantly, the adoption of priority levels makes the implementation easy and the hardware requirement simple. The specification of the input scheduling CAF is as follows:

For each input i:
If there is a currently arriving packet, P, to an eligible $VOQ_{i,j}$
Then send its HoL packet to $XP_{i,j}$ With two priority bits as follows:
If $VOQ_{i,j}$ contains other packet(s) than P
Then set the two priority bits to 11[1].
Else set the two bits to 10.
Else based on the highest priority pointer location, serve the next eligible $VOQ_{i,j}$ corresponding to $min(LXPB_j)$. The highest priority pointer is incremented (modulo N) to one location beyond the selected input $VOQ_{i,j}$.
If $VOQ_{i,j}$ contains other packet(s) than P
Then set the two priority bits to 01.
Else set these bits to 00.

We can see that the two priority bits create four priority levels for a packet. Thus, when a packet, P, leaves the input card, it has along with it its priority level for being scheduled in the output scheduling phase. According to these priority levels, a packet could be new with a nonempty corresponding input $VOQ_{i,j}$ (priority 11), or it could be not new "old" with nonempty corresponding input $VOQ_{i,j}$ (priority 10), or it could be new but with empty corresponding input $VOQ_{i,j}$ (priority 01), or it could be not new with empty corresponding input $VOQ_{i,j}$ (priority 00). These priority levels are summarized in the following table.

TABLE I. PRIORITY LEVEL OF A PACKET.

Priority Level	Nonempty VOQ	New Packet
P1	1	1
P2	1	0
P3	0	1
P4	0	0

From the table above, many combinations could be envisioned (4!). However, to perfectly accomplish the task of the input scheduling, CAF, many combinations should be eliminated. For example, priority level P4 can only be the lowest level. This is because any output scheduling which gives priority to an old internally buffered cell with empty corresponding input VOQ can't help CAF doing its job. Since the input VOQ is empty, there should be no rush in getting out that cell (queue stable and not congested). It is more urgent to send out any other packet with non-empty input VOQ. On the other hand, priority level number one (P1) can only be the highest priority among the four priority levels. This is because any output scheduling scheme which favors any priority level to the first one, leads to a decrease in the performance of CAF, and therefore will not be appropriate. An internally queued packet that comes from a busy input VOQ (having currently new packet coming in) should be sent out urgently. This avoids the VOQ from being congested or unstable. Doing so, there are only 2! different combinations left (depending on P2 and P3 ranking of table I) and are as follows:

- C1 = (P1,P2,P3,P4): using this combination means that packets are served based on the priority order[2] P1, P2, P3, and P4.
- C2 = (P1,P3,P2,P4): using this combination means that packets are served based on the priority order P1, P3, P2, and P4.

The output scheduling PRMV will then be consisting of two different schemes depending on the combination used. These two schemes are called PRMV1 and PRMV2, respectively.

C. Specification of PRMV1

The specification of the output scheduling PRMV1 is as follows:

For each output j: Starting from the highest priority pointer's location, select the next nonempty internal buffer $XP_{i,j}$ giving preference based on C1. The highest priority pointer is incremented (modulo N) to one location beyond the selected internal buffer ($XP_{i,j}$).

D. Specification of PRMV2

The specification of the output scheduling PRMV2 is as follows:

For each output j: Starting from the highest priority pointer's location, select the next nonempty internal buffer $XP_{i,j}$ giving preference based on C2. The highest priority pointer is incremented (modulo N) to one location beyond the selected internal buffer ($XP_{i,j}$).

E. PRMV Variations

In this section, two other versions of PRMV are investigated. Recall that the input scheduling and the output scheduling are performed independently. As shown before, when a packet, P, is scheduled at the input, it takes along with it two priority bits which will decide its priority for being scheduled in the output scheduling phase. However, the priority bit relative to the state of $VOQ_{i,j}$ that used to hold P might not be accurate. To see this, let's consider the following example: suppose that the current time slot is T_{now}. Suppose a packet P has entered the switch at time T_{past} and was scheduled in the input scheduling during the same time slot, T_{past}, --since CAF favors newly arriving cells-- and its $VOQ_{i,j}$ was empty at time T_{past}. Therefore, its two priority bits are set to 10 (1: new packet and 0: empty VOQ). Thus, during each time slot T, $T_{past} \le T \le T_{now}$, so long as P is still in the internal buffer, $XP_{i,j}$, it is considered as having an empty corresponding input $VOQ_{i,j}$. However, $VOQ_{i,j}$ might receive new cells during the time interval $[T_{past}+1, T_{now}]$. With this situation

[1] The first bit means that the packet, P, is new, while the second bit indicates the occupancy of the $VOQ_{i,j}$ holding P.

[2] The highest priority level is inversely proportional to the priority level index that is, P1 is the highest and P4 is the lowest.

being happening, P is treated unfairly among other packets since its priority levels do not match the reality. To avoid this problem, an alternative to PRMV was proposed. The idea is that the bit that records the state of a $VOQ_{i,j}$ is set at the moment of the output scheduling and not during the input scheduling. Proceeding this way solves the inaccuracy problem. One way to do this is by checking the corresponding input $VOQ_{i,j}$ of every internally buffered packet that is considered for an output scheduling. Therefore, we have the following two new versions of PRMV, called Pr_Check1, Pr_Check2 one for each priority scheme.

1) *Specification of Pr_Check1*

The specification of the output scheduling Pr_Check1 is as follows:

At each time slot, T, Do

For each output j: Starting from the highest priority pointer's location, select the next nonempty internal buffer $XP_{i,j}$ giving preference based on C1. The highest priority pointer is incremented (modulo N) to one location beyond the selected internal buffer ($XP_{i,j}$).

2) *Specification of Pr_Check2*

The specification of the output scheduling Pr_Check2 is as follows:

At each time slot, T, Do

For each output j: Starting from the highest priority pointer's location, select the next nonempty internal buffer $XP_{i,j}$ giving preference based on C2. The highest priority pointer is incremented (modulo N) to one location beyond the selected internal buffer ($XP_{i,j}$).

While Pr_Check seems to be better than PRMV, this solution is not a practical solution. The reason is that the output scheduler needs to perform a checking cycle during each time slot. This is undesirable due to the time wasted during the checking phase and the amount of information exchanged. Moreover, Pr_Check, performs slightly better than PRMV and the difference is only seen under light load. However, under heavy load, their performances are very close, due to steady state of the VOQs. This means that under heavy load, almost all the VOQs have more than one packet during each input scheduling cycle and therefore the unfairness problem is almost self-removed.

IV. PERFORMANCE STUDY

We evaluated our proposed scheduling algorithms using extensive simulation experiments. The simulation results are gathered from a 32*x*32 switch. Delay is measured as the period of time a cell spends waiting in an input/output/internal buffer before being scheduled. Each point in the resulting figures is obtained for 500,000 time slots (cell time), and the statistics are gathered from the 50,000th time slot. The performance evaluation is done using three traffic models: Bernoulli uniform, bursty and non-uniform traffic. We define the non-uniform (unbalanced) traffic by using an unbalanced probability w. Consider the traffic load ρ for each input port. Then, for each input port s, and output port d, the traffic load, $\rho_{s,d}$, is given by:

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

When $w=0$, the offered load is uniform, and when w=1, the offered load is completely unbalanced.

A stability performance study was carried out along with the delay study. As was presented in [3], the input queues occupancies can serve to prove the stability of the scheduling algorithm. That is, if under a service policy (scheduling algorithm) X, one can show that $E(||L(n)||) < \infty$,[1] he can conclude that X is stable. $||L(n)||$ is the l-two norm vector representing the occupancy of the VOQs a time n and defined as follows:

$$|| L(n) ||= \sqrt{Q_{1,1}(n)^2 + ... + Q_{1,N}(n)^2 + ... + Q_{N,1}(n)^2 + ... + Q_{N,N}(n)^2}\ .$$

In this section, we just present, through simulation, the occupancy of the VOQs under the above-mentioned scheduling algorithms. This study can be used to find a practical upper bound on the input buffer space that a scheduling algorithm needs to prevent congestion.

Fig.2 shows the performance evaluation of the iSLIP, for non-buffered crossbar, the RR-RR, LQF-RR, and OCF-OCF along with our group of proposed algorithms. The performance of iSLIP is very low when compared to all the buffered crossbar algorithms. All our proposed algorithms have shorter queuing delay than all existing schemes. CAF-Pr_Check has the best performance among all, with very small difference when compared to CAF-PRMV1.

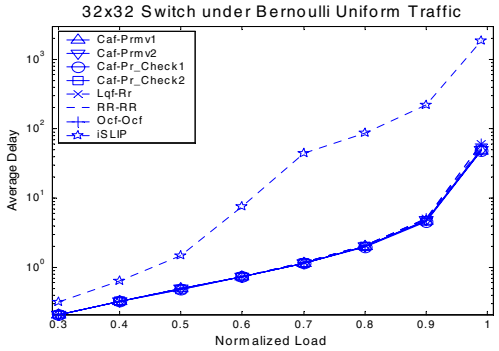

Figure 2. Average Delay under Bernoulli I.I.D. uniform traffic.

As for the performance under bursty traffic, Fig.3, Our group of proposed algorithms has the best performance among all the algorithms tested. The largest delay among our algorithms was always under 800. However, the delay for LQF-RR and OCF-OCF is more than 900 each, and RR-RR is 888.

Fig.4 shows the stability performance of the input VOQs under bursty traffic. CAF-Pr_Check2 has the worst performance among all. The reason of this low performance is due to the highest priority given to the new arriving packet irrespective of

[1] The expected value of the l-two norm vector, $L(n)$, representing the input $VOQs$ occupancy is finite.

the state of its VOQ. However, in the event of no arrivals, CAF serves packets based on the minimum occupied internal buffer.

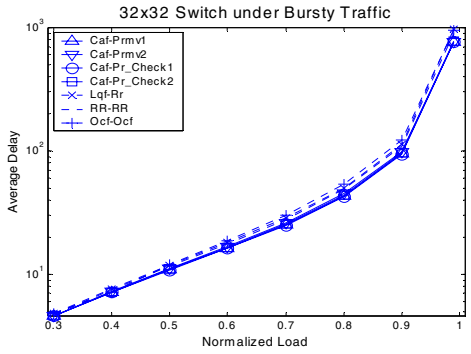

Figure 3. Delay performance under Bursty uniform traffic.

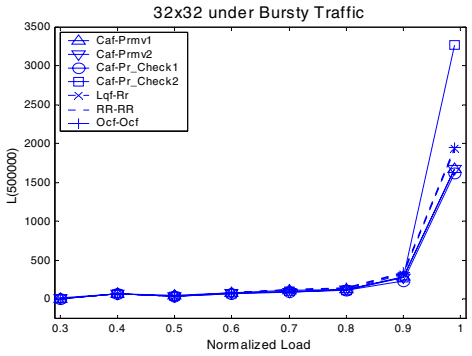

Figure 4. VOQs Occupancies under Bursty uniform traffic.

In Fig.5, the unbalanced coefficient, *w*, is fixed to 0.5. The output scheduling algorithms PRMV2 and Pr_Chech2 perform less than the others because they give priority to the newly coming packet irrespective of whether its VOQ is empty or not, and this is not appropriate to the input scheduling CAF. This is because, in many cases, CAF serves packets based on the minimum occupied internal buffer and not on the newly coming packet. Among all, CAF-Pr_Check1 has the best performance, thereafter CAF-PRMV1 and then LQF-RR and OCF-OCF.

As for the VOQs occupancies, CAF-PRMV1 and CAF-Pr_Check1 have the minimum queues occupancies among all the algorithms. Fig. 6 shows the VOQs occupancies under the unbalanced traffic pattern.

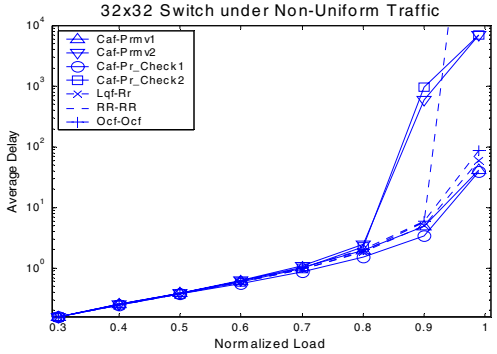

Figure 5. Delay performance under unbalanced traffic (*w=0.5*).

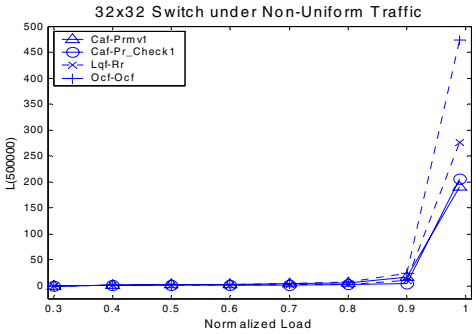

Figure 6. VOQs Occupancies under unbalanced traffic (*w=0.5*).

V. CONCLUSION

In this paper, we proposed a group of practical algorithms for the VOQ/BCS switch. We illustrated their performance by comparing them with the previously proposed algorithms. The simulation results showed that our newly proposed algorithms outperform state-of-art algorithms in this area. In particular, the CAF-PRMV1 algorithm performs very well under all traffic patterns, and was shown to be the best among all others. Its main advantage lies in the fact that it is totally stateless, which makes it simple in hardware implementation while running at very high speed.

REFERENCES

[1] M. Karol, M. Hluchyj, "Queuing in High-performance Packet-switching," *IEEE J. Selected Area Communications, Vol. 6, pp. 1587-1597, December 1988.*

[2] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High-speed Switch Scheduling for Local Area Networks," *ACM Trans. Comput. Syst., pp. 319-352, Nov. 1993.*

[3] N. McKeown, A. Mekkittikul, V. Anantharam, J. Walrand. "Achieving 100% throughput in Input-Queued Switch," *IEEE Trans. On Communications, Vol.47, No. 8 August 1999.*

[4] N. McKeown, "iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Transactions on Networking, Vol.7, No.2, pp. 188-201, April 1999.*

[5] D.N. Serpanos, P. I. Antoniadis, "FIRM: A Class of Distributed Scheduling Algorithms for High-Speed ATM Switches with Input Queues," *IEEE INFOCOM 2000.*

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

[7] S. Nojima, E. Tsutsui, H. Fukuda, and M. Hashimmoto, "Integrated Packet Network Using Bus Matrix," *IEEE Journal on Selected Areas in Communications, Vol. 5, No. 8, pp.1284-1291, Oct. 1987.*

[8] A.K. Gupta, L. O. Barbosa, and N. D. Gorganas, "16x16 Limited Intermediate Buffer Switch Module for ATM Networks for B-ISDN," *GLOBECOM'91, pp.939-943, Dec. 1991.*

[9] A.K. Gupta, L. O. Barbosa, and N. D. Gorganas, "16x16 Limited Intermediate Buffer Switch Modules and Their Interconnection Networks for B-ISDN," *ICC'92, pp.1646-1650, June 1992.*

[10] M. Nabeshima, "Performance Evaluation of Combined Input-and Crosspoint-Queued Switch," *IEICE Trans. Commun., Vol. E83-B, No.3 March 2000.*

[11] R. Rojas-Cessa, E. Oki, and H. J. Chao, "CIXB-1: Combined Input One-Cell-Crosspoint Buffered Switch," *Proceedings of the 2001 IEEE Workshop on High Performance Switching and Routing, 2001, pp. 271-275.*

[12] T. Javadi, R. Magill, and T. Hrabik, "A high-Throughput Algorithm for Buffered Crossbar Switch Fabric," *Proceedings IEEE ICC, pp. 1581-1591, June 2001.*