Providing Flow Based Performance Guarantees for Buffered Crossbar Switches

Deng Pan
Dept. of Electrical & Computer Engineering
Florida International University
Miami, Florida 33174, USA
pand@fiu.edu

Yuanyuan Yang
Dept. of Electrical & Computer Engineering
State University of New York
Stony Brook, NY 11794, USA
yang@ece.sunysb.edu

Abstract

Buffered crossbar switches are a special type of combined input-output queued switches with each crosspoint of the crossbar having small on-chip buffers. The introduction of crosspoint buffers greatly simplifies the scheduling process of buffered crossbar switches, and furthermore enables buffered crossbar switches with speedup of two to easily provide port based performance guarantees. However, recent research results have indicated that, in order to provide flow based performance guarantees, buffered crossbar switches have to either increase the speedup of the crossbar to three or greatly increase the total number of crosspoint buffers, both adding significant hardware complexity. In this paper, we present scheduling algorithms for buffered crossbar switches to achieve flow based performance guarantees with speedup of two and with only one or two buffers at each crosspoint. When there is no crosspoint blocking in a specific time slot, only the simple and distributed input scheduling and output scheduling are necessary. Otherwise, the special urgent matching is introduced to guarantee the on-time delivery of crosspoint blocked cells. With the proposed algorithms, buffered crossbar switches can provide flow based performance guarantees by emulating push-in-first-out output queued switches, and we use the counting method to formally prove the perfect emulation. For the special urgent matching, we present sequential and parallel matching algorithms. Both algorithms converge with N iterations in the worst case, and the latter needs less iterations in the average case. Finally, we discuss an alternative backup-buffer implementation scheme to the bypass path, and compare our algorithms with existing algorithms in the literature.

Keywords: Buffered crossbar switches, cell scheduling, performance guarantees, stable marriage.

1 Introduction

With the rapid development of broadband networks in recent years, a variety of novel Internet based multimedia applications, such as Voice over IP, Video on Demand, and Distance Education, have been developed, which usually have different quality of service (QOS) requirements. The capability to provide QOS support has become an important issue for the design of modern switches and routers [10]. Switches and routers control the departure order of packets from different flows, and the scheduling algorithms adopted largely determine the quality of service that can be provided by the networks.

Due to the unique advantages of crossbar switches, they have long been used as high speed interconnects in various computing environments, such as PC clusters, Internet routers, and system-on-chip networks. The crossbar provides non-blocking capability, and also overcomes the bandwidth limitation of bus based switching fabrics.

Packets in crossbar switches may be buffered at either output ports, input ports, or crosspoints of the crossbar. Based on the locations to store packets, crossbar switches can be divided into several different categories: output queued (OQ) switches, input queued (IQ) switches, combined input-output queued switches (CIOQ) switches, and buffered crossbar switches.

Output queued (OQ) switches only have buffer space at the output side. There may be more than one logical queues at each output port to differentiate packets from different input ports or different flows. Since there is no buffer space at the input side, if multiple input ports have packets arriving at the same time and destined for the same output port, all the packets have to be simultaneously transmitted through the crossbar and stored in the output buffers. Thus, in order to achieve 100% throughput, a crossbar with speedup of N is necessary for an $N \times N$ OQ switch, or in other words, the crossbar needs N times bandwidth as that of the input port and output port. Several fair scheduling algorithms, such as WFQ [5] and DRR [18], have been proposed

for OQ switches to emulate the ideal Generalized Processor Sharing (GPS) fairness model. They provide different levels of performance guarantees using different scheduling approaches.

Input queued (IQ) switches only have buffer space at the input side, and thus eliminate the speedup requirement. For input buffers, virtual output queued (VOQ) buffering [12] is usually used, because traditional first-in-first-out (FIFO) buffering suffers from the head of line (HOL) blocking, which limits the maximum throughput of the switch. Unfortunately, until now IQ switches are found to be able to achieve 100% throughput only with maximum matching algorithms or their variants [12], which have high time complexity. Fair scheduling algorithms for IQ switches, such as iFS [15] and iDRR [22], usually work in an iterative mode, and provide performance guarantees by emulating the corresponding fair scheduling algorithms for OQ switches.

In order to combine the advantages of both OQ switches and IQ switches, combined input-output queued (CIOQ) switches make a trade-off between the crossbar speedup and the complexity of the scheduling algorithm. They usually have small fixed speedup of two, and thus need buffer space at both the input side and output side. CIOQ switches with speedup of two are shown to achieve 100% throughput with any maximal scheduling algorithm [4]. In addition, CIOQ switches are proved to be able to emulate push-in-first-out (PIFO) OQ switches [2]. Therefore, special scheduling algorithms (with high computational complexity) can be designed for CIOQ switches to duplicate the packet departure order and time of existing fair scheduling algorithms for OQ switches, and provide desired performance guarantees.

With the development of modern VLSI technology, it has been feasible to integrate small on-chip memory to the crossbar switching fabric. Buffered crossbar switches, or combined input-crosspoint-output queued (CICOQ) switches are a special type of CIOQ switches, where each crosspoint of the crossbar is equipped with small exclusive buffers. Due to the introduction of crosspoint buffers, the scheduling process is greatly simplified [7] [11] [21].

Performance guarantees provided by switches can be at different granularity levels. We say that a switch provides port based performance guarantees, if packets from different input ports are treated differently by the output port of the switch. For example, such a switch can ensure the bandwidth allocated to a specific input port at each output port. However, with port based performance guarantees, there is no way to differentiate the packets from the same input port but from different flows. Hence, it is possible for one of the flows to inject a larger amount of traffic into the shared buffer and use up all the bandwidth allocated to its input port, causing packet loss to other flows of the same input port. On the contrary, if a switch provides flow based performance guarantees, resources are allocated on a per flow basis, and each flow can have its guaranteed bandwidth, delay, or jitter performance.

In [3], Chuang et al. analyzed the capability of buffered crossbar switches to provide performance guarantees. They showed that speedup of two is sufficient for buffered crossbar switches to emulate restricted PIFO OQ switches (where restricted PIFO OQ switches are PIFQ OQ switches with the restriction that cells of an input-output pair depart from the switch in the same order as they arrive), or in other words, to achieve port based performance guarantees. However, in order for buffered crossbar switches with speedup of two to provide flow based performance guarantees, either a separate crosspoint buffer must be available for each flow, or the switch structure must first be modified with a more complicated buffering scheme (similar to that of OQ switches) and then a total of N^3 crosspoint buffers must be provided for an $N \times N$ switch. Unfortunately, both schemes greatly increase the total number of crosspoint buffers and are not scalable. Alternatively, the speedup of the crossbar may be increased to three, which will drop the maximum throughput of the switch by one third. The additional speedup of one is used to eliminate the crosspoint blocking, which refers to the situation that a cell in the input buffer with earlier departure time is blocked by another cell already in the crosspoint buffer from a different flow and with later departure time. The crosspoint blocking may happen when a new cell arrives at the input buffer. In such a case, the locations of the blocked cell and the blocking cell are exchanged using the additional speedup. Because at most one cell may arrive at each input port in a single time slot, the additional speedup of one is guaranteed to completely remove the crosspoint blocking.

Although buffered crossbar switches are able to directly schedule variable length packets without segmentation-and-reassembly (SAR) [21] [19], following the work in [3], we consider only fixed length packet scheduling, or cell scheduling, and make the switch work in a synchronous time slot mode. Since a crosspoint buffer must be at least large enough to store a single packet, cell scheduling has less hardware requirements than packet scheduling. For example, the maximum packet length in Ethernet networks is 1500 bytes, while the fixed cell length in ATM networks is 53 bytes. Considering that on-chip memories are expensive resources, this feature reduces the hardware cost for buffered crossbar switches.

In this paper, we present scheduling algorithms to provide flow based performance guarantees for buffered crossbar switches with speedup of two and with one or two buffers at each crosspoint of the crossbar. We add a simple bypass path to each crosspoint, which can directly transmit a cell from the input buffer to the output buffer. When there are no crosspoint blocked packets, the switch uses simple

and distributed scheduling algorithms. Otherwise, an extra urgent matching sub-phase is conducted to transmit the crosspoint blocked cells using the bypass paths, so as to guarantee their on-time delivery to the output buffers. Using the counting method, we formally prove that the buffered crossbar switch using our approach can emulate any PIFO OQ switch to achieve flow based performance guarantees. For the urgent matching process, we present sequential and parallel matching algorithms. Both algorithms converge within N iterations for an $N \times N$ switch in the worst case, and the latter can achieve a smaller number of convergence iterations for the average case. Finally, we give an alternative implementation scheme to remove the bypass path, which adds only one more buffer to each crosspoint, and we compare our algorithms with existing algorithms in the literature.

The rest of the paper is organized as follows. In Section 2, we provide an overview of the scheduling algorithms proposed in the literature for buffered crossbar switches. In Section 3, we introduce the structure of the buffered crossbar switches considered in this paper, and describe the counting proof method. In Section 4, we present the corresponding scheduling algorithms, and formally prove the flow based performance guarantees. In Section 5, we give the sequential and parallel urgent matching algorithms. In Section 6, we discuss how to remove the bypass path by adding one more buffer to each crosspoint, and make comparison between our approaches and other existing approaches. Finally, in Section 7, we conclude the paper.

2 Related Work

Scheduling algorithms for buffered crossbar switches in the literature can be broadly classified into two categories: those to achieve high throughput and those to emulate scheduling algorithms for OQ switches.

A buffered crossbar switch architecture called CIXB-1 was proposed in [16], where each crosspoint has a one-cell buffer. CIXB-1 offers several advantages for feasible implementation such as scalability and timing relaxation. It is shown that, in conjunction with round robin arbitration, CIXB-1 can provide 100% throughput under uniform traffic. CIXOB-k [17] is the extended version of CIXB-1 with a k-cell buffer at each crosspoint and small speedup for the crossbar. CIXOB-k is shown to be able to achieve 100% throughput under uniform traffic as well as non-uniform traffic. A cell scheduling scheme for buffered crossbar switches called Most Critical Buffer First (MCBF) was proposed in [13]. It conducts scheduling based on the crosspoint buffer information and has low hardware complexity. MCBF exhibits good performance and shows optimal stability in simulations. Shortest Crosspoint Buffer First (SCBF) [23] is another cell scheduling scheme, which finds a matching with minimum weight in each time slot. It is proved that SCBF achieves 100% throughput for any admissible traffic without speedup requirement. In order to facilitate hardware implementation, a maximal solution of SCBF was also proposed in [23], which achieves low $O(\log N)$ time complexity and is shown to have almost identical performance. The algorithms discussed in the above are cell scheduling algorithms targeting high throughput.

The emulation of OQ switches by buffered crossbar switches was studied in [11]. It is proved that buffered crossbar switches with speedup of two satisfying non-negative slackness (NNS) insertion and lowest time to live (LTTL) blocking, and LTTL fabric scheduling can exactly emulate OQ switches. In particular, it is shown that the GB-VOQ_OCF scheduling algorithm can exactly emulate FIFO OQ switches, and the GBFG_SP scheduling algorithm can exactly emulate strict priority OQ switches. In [14], the MCAF-LTF cell scheduling scheme for one-cell buffered crossbar switches was proposed. MCAF-LTF does not require costly time stamping mechanism, and is proved to be able to emulate OQ switches with speedup of two. [3] studied practical scheduling algorithms for buffered crossbar switches. It is shown that with speedup of two, buffered crossbar switches can mimic restricted PIFO OQ switches, regardless of the incoming traffic pattern, and that with speedup of three, buffered crossbar switches can mimic arbitrary PIFO OQ switches and hence provide delay guarantees. It is also shown that buffered crossbar switches can achieve 100% throughput with speedup of two for any Bernoulli i.i.d. admissible traffic. The above algorithms also consider cell scheduling, but are mainly designed to emulate scheduling algorithms for OQ switches.

A buffered crossbar switch architecture supporting packet scheduling was proposed in [8]. The chip layout was presented and the hardware cost was analyzed. The simulation results demonstrate that the proposed architecture outperforms unbuffered crossbar switches. A segmentation and reassembly (SAR) scheme was proposed in [9]. It uses variable size segments while merging multiple packets into each segment. The proposed scheme eliminates padding overhead, reduces header overhead and crosspoint buffer size, and is suitable for use with external, modern DRAM buffer memory in ingress line cards. The simulation results show that it outperforms existing segmentation schemes in buffered as well as unbuffered crossbar switches. The performance guarantees of packet scheduling for asynchronous buffer crossbar switches were discussed in [21], and two algorithms were designed based on existing cell scheduling algorithms. It is theoretically proved that, with speedup of two, the Packet GVOQ (Group by Virtual Output Queue) scheduling algorithm provides work-conserving guarantees with $2L$ crosspoint buffer space and can emulate a PIFO scheduling algorithm for OQ switches

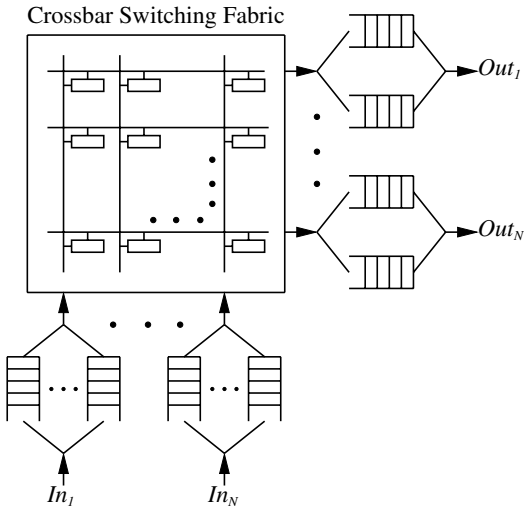

Figure 1. The structure of buffered crossbar switches.

with $5L$ crosspoint buffer space, where L is the maximum packet length. The Packet LOOFA (Lowest Output Occupancy First Algorithm) scheduling algorithm provides work-conserving guarantees with $16L/3$ crosspoint buffer space, and can emulate a PIFO scheduling algorithm for OQ switches with $22L/3$ crosspoint buffer space. [19] proposed the Distributed Packet Fair Queueing (DPFQ) architecture for physically dispersed line cards to emulate an OQ switch with fair queueing, and the simulation results demonstrate that the resulting system provides service that closely approximates an output buffered switch employing fair queueing with modest speedup. The above schemes use variable length packets as the scheduling and switching units.

3 Preliminaries

In this section, we give the switch structure considered in this paper, and describe the counting method that will be used by the proof in Section 4.

3.1 Buffered Crossbar Switch Structure

The switch structure considered in this paper is illustrated in Figure 1. N input ports and N output ports are connected by a crossbar switching fabric, which has speedup of two. Packets are buffered at input ports, output ports, and crosspoints. At input buffers, packets are stored on a per-flow basis to avoid the HOL blocking. In other words, the number of logical virtual queues at any input buffer is equal to the number of flows coming from the input port, and each virtual queue stores the packets of a single flow

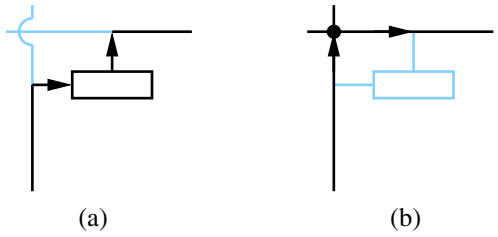

Figure 2. Two possible data transmission paths. (a) The normal path. (b) The bypass path.

in their arrival order. Similarly, output buffers are also organized on a per-flow basis, so as to differentiate cells of different flows and provide flow based performance guarantees. Each crosspoint has a small exclusive buffer, whose size is equal to the length of a single cell.

Similar to [3], we make the assumption that in each time slot, at most one cell can be injected into an input buffer, and one cell will be removed from an output buffer if it is not empty. Note that the crossbar has speedup of two. In other words, it is able to retrieve two cells from each input buffer and deliver two cells to each output buffer in a single time slot.

The considered switch has two possible paths to transmit a cell from the input buffer to the output buffer, which we call the normal path and the bypass path, respectively. For the normal path, as shown in Figure 2(a), a cell is first sent from the input buffer to the crosspoint buffer, from where it will be retrieved to the output buffer. For the bypass path, as illustrated in Figure 2(b), a cell is directly transmitted from the input buffer to the output buffer without being temporarily stored at the crosspoint. It should be noted that, at any specific time, only one of the two paths is available.

The bypass path slightly increases the hardware complexity of the switching fabric, but it is practically available with state-of-the-art IC technology. As will be seen in Section 6, the bypass path can be easily removed by equipping each crosspoint with one additional buffer.

The operation of a buffered crossbar switch can be divided into several phases. In this paper, we consider the following order of phases, which does not affect the generality of the results but simplifies the analysis.

- *Arrival phase.* A new cell arrives at each input buffer and is put into one of the virtual queues in the input buffer.
- *First scheduling phase.* There are three sub-phases: input scheduling, urgent matching, and output scheduling, where the urgent matching sub-phase is not necessary if there are no crosspoint blocked packets. In

input scheduling, an input port selects one of the virtual queues with empty crosspoint buffers, and sends its first cell to the corresponding crosspoint buffer. In urgent matching, some crosspoint blocked cells are arranged to transmit directly from the input buffers to the output buffers using the bypath paths. In output scheduling, an output port chooses one of the occupied crosspoint buffers, and retrieves the buffered cell to the output buffer.

- *Second scheduling phase.* Because the crossbar has speedup of two, a second scheduling phase is also conducted.

- *Departure phase.* One cell is removed from each output buffer and is sent to either the output line or a special buffer to reassemble the segmented cells back to the original packet.

3.2 The Counting Method

The counting method is widely used to prove the emulation of PIFO OQ switches by CIOQ switches [2] [11] or buffered crossbar switches [3]. For PIFO OQ switches, there is a single PIFO queue at each output port. With such a PIFO queue, a newly arrived cell may be inserted at any position in the queue, and the cell that departs next is always the first cell of the queue. Since after a cell has been put into the queue, its position cannot be moved any more, the departure sequence of the cells in the queue is predictable. In order words, the relative transmission order of the awaiting cells is not affected by future arrivals. Many fair scheduling algorithms for OQ switches, such as WFQ [5] and DRR [18], use such a PIFO queueing policy. If a buffered crossbar switch is proved to be able to emulate PIFO OQ switches, it can mimic the scheduling behavior of the fair scheduling algorithms for OQ switches, and therefore provide desired bandwidth, delay, and jitter performance guarantees.

In the following, we give some definitions for the counting method.

- *Shadow OQ Switch.* A shadow OQ switch is the switch that the switch considered intends to emulate. A successful emulation of the shadow OQ switch means that, for any incoming traffic, the departure time of each cell in both switches is exactly the same.

- *Output Priority.* Output priority determines the departure order of different cells from the output buffer. The output priority of a cell is defined by its departure time in the shadow OQ switch. For two cells destined for the same output port, the one with higher output priority leaves the output buffer first.

- *Input Priority.* Input priority determines the departure order of different cells from the input buffer. The input priority of a cell is defined by the Group by VOQ (GB-VOQ) policy [2], which we will describe in more detail in Section 4. For two cells in the same input buffer whose crosspoint buffers are both empty, the one with higher input priority leaves first.

- *Output Cushion.* The output cushion of a cell c is the number of cells that are in the destination output buffer of c and have higher output priority. We use $OC(c,t,A)$, $OC(c,t,F)$, $OC(c,t,S)$ and $OC(c,t,D)$ to denote the output cushion of c at time slot t after the arrival phase, the first scheduling phase, the second scheduling phase and the departure phase, respectively.

- *Input Thread.* The input thread of a cell c is the number of cells that are in the same input buffer as c and have higher input priority. If c is in the crosspoint buffer, we define its input thread to be zero. We use $IT(c,t,A)$, $IT(c,t,F)$, $IT(c,t,S)$ and $IT(c,t,D)$ to denote the input thread of c at time slot t after the arrival phase, the first scheduling phase, the second scheduling phase and the departure phase, respectively.

- *Slackness.* The slackness of a cell c is equal to its output cushion minus its input thread. We use $L(c,t,A)$, $L(c,t,F)$, $L(c,t,S)$ and $L(c,t,D)$ to denote its slackness at time slot t after the arrival phase, the first scheduling phase, the second scheduling phase and the departure phase, respectively, and we have $L(c,t,*) = OC(c,t,*) - IT(c,t,*)$, where $*$ can be either A, F, S, or D.

Intuitively, the output cushion of a cell indicates the number of time slots that the cell can wait before departs from the output buffer, and the input thread indicates the number of time slots that the cell has to wait to enter the output buffer. As their difference, the slackness reflects the urgency level to transmit a cell to its output buffer. The basic idea of the counting method is to maintain non-negative slackness for each cell, so that the cell can be guaranteed to depart from its output buffer on time.

4 Achieve Flow Based Performance Guarantees with Speedup of Two

In this section, we describe the scheduling algorithms for each operation phase, and use the counting method to formally prove the emulation of PIFO OQ switches by buffered crossbar switches with speedup of two.

4.1 Arrival Phase

At the arrival phase of time slot t, a new cell c may arrive at each input buffer. It is put at the end of the corresponding virtual queue, and its input priority is determined by the GBVOQ policy as follows. If its virtual queue is empty, it is inserted at the head of the input priority list with zero input thread, i.e., $IT(c, t, A) = 0$. Otherwise, if its virtual queue is not empty, it is inserted in the input priority list immediately after the last cell d of its virtual queue. In the latter case, the input thread of c is equal to that of d plus one, i.e., $IT(c, t, A) = IT(d, t, A) + 1 = IT(d, t - 1, D) + 1$.

The purpose to insert a cell at the head of the input priority list when its virtual queue is empty is to ensure that it can be delivered on time to the output buffer. When a new cell arrives at the input buffer, it is possible that the cell departs from the output buffer in the shadow OQ switch at the same time slot. Assigning the cell the highest input priority guarantees that it can be immediately transmitted to the output buffer and depart from the output buffer at the same time slot. On the other hand, a cell with a non-empty virtual queue does not need the highest input priority, since it has to wait for the cells ahead of it in the same virtual queue to depart first.

4.2 Scheduling Phase

Since cells of the same flow are stored in the same virtual queue and the head cell has the highest output priority, the head cell should always be delivered first to the output buffer. Thus, we only need to consider such head cells in the scheduling phase, and we divide them into two categories: blocked cells and normal cells. A blocked cell is a head cell whose crosspoint is occupied by another cell with lower output priority. The rest of the head cells are called normal cells.

The input scheduling sub-phase is conducted as follows. Each input port independently selects the normal cell with an empty crosspoint buffer and the highest input priority. Note that this is a distributed operation, and there is no information exchange between different input ports. The cell selected is called a scheduled cell. It may happen that there is no scheduled cell in an input buffer, because all the crosspoint buffers have already been occupied.

If there is no blocked cell, the switch can now skip the urgent matching sub-phase and directly proceed to the output scheduling sub-phase. Otherwise, the urgent matching sub-phase is conducted for blocked cells. In such a case, the scheduled cell will not be immediately sent to the crosspoint buffer, but wait in the input buffer for the scheduling decisions for blocked cells.

We define a special type of blocked cells called urgent cells. A blocked cell is an urgent cell if

1. its slackness is -1 for the first scheduling phase or 0 for the second scheduling phase,
2. it has the highest input priority among all such cells in the same input buffer destined for the same output port,
3. it has higher input priority than the scheduled cell, and
4. it has higher output priority than all the crosspoint buffered packets to its destination output port.

Such a cell is urgent because if the scheduled cell of its input buffer is sent to the crosspoint buffer, neither its input thread is going to decrease nor its output cushion to increase, and it will not be able to catch the departure time in the shadow OQ switch.

From the definition, it is easy to see the following property.

Property 1. *There is at most one urgent cell for each output port in any input buffer.*

Based on the urgent cells and the scheduled cells with zero input thread, the urgent matching is conducted, in which some urgent cells will be arranged to directly transmit from the input buffers to the output buffers. The purpose of including the scheduled cells with zero input thread in the urgent matching is to ensure that, if such a cell has higher output priority than all the urgent cells to the same output port, this cell instead of any urgent cell will be delivered to the output buffer at the end of the scheduling phase. We define a matched cell to be a cell that is chosen to transmit in the urgent matching. The urgent matching uses the stable matching algorithms, and has the following property.

Property 2. *For any cell that participates in the urgent matching sub-phase but is not matched, there must be a matched cell either from the same input buffer with higher input priority or to the same output buffer with higher output priority.*

For ease of reading, we will directly use Property 2 in the rest of this section, but present sequential and parallel algorithms for the urgent matching in Section 5.

Now we have obtained the input scheduling results and the urgent matching results. The cells are arranged to leave the input buffers as follows. If a matched cell is an urgent cell, which we call a matched urgent cell, it will be directly transmitted from its input buffer to the output buffer using the bypass path. On the other hand, if a matched cell is a scheduled cell, which we call a matched scheduled cell, it will be sent to the crosspoint buffer. In the third case, if an input has a scheduled cell but no matched cell, the scheduled cell is also sent to the crosspoint buffer. It should be noted that all the cells leave their input buffers at the

same time, because the switch is working in a synchronous time slot mode. The only difference is that the matched urgent cells take the bypass paths, while the other cells take the normal paths.

Next, the output scheduling sub-phase is conducted. Each output port independently retrieves the cell with the highest output priority from one of its crosspoint buffers. If there was an urgent match sub-phase, some output ports may have received matched urgent cells from the bypass paths, and they do not need to retrieve packets from the crosspoint buffers anymore.

From the above description for the output scheduling sub-phase, we can obtain the following lemma.

Lemma 1. *For any cell that participates in the urgent matching sub-phase but is not matched, its slackness is increased by at least one after the current scheduling phase.*

Proof. From Property 2, we know that for a cell not matched in the urgent matching sub-phase, there is a matched cell either from the same input with higher input priority or to the same output buffer with higher output priority. We analyze each of the two cases below.

In the first case, since there is a matched cell from the same input with higher input priority, no matter it is a matched urgent cell or a matched scheduled cell, it will be removed from the input buffer. Thus, the input thread of the cell that is not matched decreases by one. Because the output cushion of any cell will not decrease during the scheduling phase, its slackness is guaranteed to increase by at least one.

In the second case, there is a matched cell to the same output buffer with higher output priority. If the matched cell is a matched urgent cell, it will be delivered to the output buffer by the bypass path and it has higher output priority. Otherwise, if the matched cell is a matched scheduled cell, it will be sent to the crosspoint buffer. Note that there will be no matched urgent cell to the same output port. Thus, the cell delivered to the output buffer must come from one of the crosspoint buffers, and according to the output scheduling policy, the delivered cell must have higher output priority than or the same output priority as that of the matched scheduled cell, and therefore have higher output priority than that of the cell not matched. As a result, no matter the matched cell to the same output port is a matched urgent cell or a matched scheduled cell, the output cushion of the cell that is not matched increases by one. Similarly, considering that the input thread of any cell will not increase during the scheduling phase, its slackness is guaranteed to increase by at least one. □

4.3 Departure Phase

In the departure phase, each output port independently finds the cell in its output buffer that has the highest output priority and removes it from the output buffer.

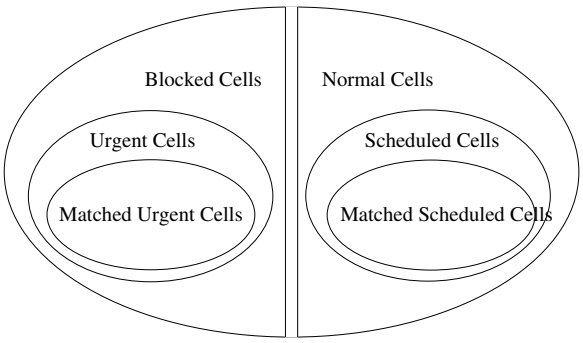

Figure 3. The relationship of different types of cells.

4.4 Flow Based Performance Guarantees

With the above scheduling algorithms, we are ready to prove the flow based performance guarantees using the counting method.

We need the following lemma regarding the first scheduling phase.

Lemma 2. *For any cell c not in the output buffer, if its slackness after the arrival phase is larger than or equal to -1, then its slackness after the first scheduling phase is larger than or equal to 0, i.e.,*

$$L(c,t,A) \geq -1 \Rightarrow L(c,t,F) \geq 0$$

Proof. We first assume that c is in the crosspoint buffer after the arrival phase. After the first output scheduling sub-phase, c is either in the output buffer or still in the crosspoint buffer. If c is in the output buffer, it can depart at any time, and we do not need to consider its slackness. Otherwise, if c is still in the crosspoint buffer, then another cell d is delivered to the output buffer, which might be a matched urgent cell or a cell from another crosspoint buffer. In either case, d must have higher output priority than c, and as a result, $OC(c,t,F) = OC(c,t,A) + 1$. Since $IT(c,t,F) = IT(c,t,A) = 0$, we have that $L(c,t,F) \geq L(c,t,A) + 1 \geq -1 + 1 = 0$.

Next, we assume that c is in the input buffer after the arrival phase. We have defined several different types of cells for the scheduling phase. The relationship between these cell types is illustrated in Figure 3. We prove that the lemma holds for each type of cells.

Matched urgent cells. If c is a matched urgent cell, it has been transmitted to the output buffer. As a result, it can depart at anytime when necessary.

Urgent cells. If c is an urgent cell that is not matched in the urgent matching, based on Lemma 1, we know that $L(c,t,F) \geq L(c,t,A)+1 \geq -1+1=0$.

Blocked cells. By definition, if c is a blocked cell but not an urgent cell, there are four possible reasons. In the following, we analyze each of them.

1. The slackness of the blocked cell c is not equal to -1. Since $L(c,t,A) \geq -1$ and $L(c,t,A) \neq -1$, we have $L(c,t,A) \geq 0$. Because after the first scheduling phase $IT(c,t,F) \leq IT(c,t,A)$ and $OC(c,t,F) \geq OC(c,t,A)$, we have $L(c,t,F) \geq L(c,t,A) \geq 0$.

2. The slackness of the blocked cell c is -1, i.e., $L(c,t,A) = -1$, but there is another blocked cell in its input buffer that has the same slackness and higher input priority, such as the urgent cell d. Since $L(d,t,A) = L(c,t,A)$, and by the definition of urgent cells, $IT(d,t,A) < IT(c,t,A)$, it is easy to see that $OC(d,t,A) < OC(c,t,A)$ as well. In other words, d has both higher input priority and output priority than c. Thus, $OC(c,t,F) - OC(c,t,A) \geq OC(d,t,F) - OC(d,t,A)$ and $IT(c,t,F) - IT(c,t,A) \leq IT(d,t,F) - IT(d,t,A)$, and we can obtain $L(c,t,F) - L(c,t,A) \geq L(d,t,F) - L(d,t,A)$. Since we have proved in the above that for urgent cells, $L(d,t,F) \geq L(d,t,A)+1$, it follows that $L(c,t,F) \geq L(c,t,A)+1 \geq -1+1=0$.

3. The blocked cell c has lower input priority than the scheduled cell d of the input port, i.e., $IT(d,t,A) < IT(c,t,A)$. If d is removed from the input buffer, then $IT(c,t,F) = IT(c,t,A)-1$. Otherwise, a matched urgent cell e instead of the scheduled cell d is removed from the input buffer. Since e is an urgent cell, it has higher input priority than the scheduled cell d, i.e., $IT(e,t,A) < IT(d,t,A)$. As a result, we can obtain $IT(e,t,A) < IT(c,t,A)$, and therefore $IT(c,t,F) = IT(c,t,A) - 1$ due to the removal of e. Thus, in both cases, we have $IT(c,t,F) = IT(c,t,A) - 1$. Again, since $OC(c,t,F) \geq OC(c,t,A)$, it follows that $L(c,t,F) \geq L(c,t,A)+1 \geq -1+1=0$.

4. The blocked cell c has lower output priority than a crosspoint buffered cell d to the same output port. If a cell e in the crosspoint buffer is delivered to the output buffer, we know that e has higher output priority than or the same output priority as that of d. As a result, e has higher output priority than c, and $OC(c,t,F) = OC(c,t,A)+1$ due to the delivery of e. Otherwise, a matched urgent cell f is transmitted to the output buffer. Since f is an urgent cell, it has higher output priority than any crosspoint buffered cell, including d. As a result, f has higher output priority than c, and $OC(c,t,F) = OC(c,t,A)+1$ due to the delivery of f. Thus, in both cases, $OC(c,t,F) = OC(c,t,A)+1$. Similarly, considering that $IT(c,t,F) \leq IT(c,t,A)$, we have $L(c,t,F) \geq L(c,t,A)+1 \geq -1+1=0$.

To sum up, for the first reason, nothing needs to be done to ensure $L(c,t,F) \geq 1$, and for other three reasons, the slackness of the cell is increased by one by the end of the first scheduling phase.

Matched scheduled cells. If a scheduled cell c is matched in the urgent matching process, it will be sent from the input buffer to the crosspoint buffer, resulting in zero input thread after the input scheduling sub-phase. Since there is only one matched cell to each output port, there will be no matched urgent cell to the same output port. As a result, the cell delivered to the output buffer must be from one of the crosspoint buffers. If the delivered cell is c, we do not need to consider its slackness any more. Otherwise, if a cell d from another crosspoint buffer is delivered, d must have higher output priority due to the output scheduling policy, and therefore $OC(c,t,F) = OC(c,t,A)+1$. Considering $IT(c,t,F) = IT(c,t,A) = 0$, we have $L(c,t,F) \geq L(c,t,A)+1 \geq -1+1=0$.

Scheduled cells. If a scheduled cell c does not leave the input buffer during the input scheduling due to a matched urgent cell e, because e is an urgent cell and has higher input priority, i.e., $IT(e,t,A) < IT(c,t,A)$, we can have $IT(c,t,F) = IT(c,t,A)-1$ due to the removal of e. Since $OC(c,t,F) \geq OC(c,t,A)$, it follows that $L(c,t,F) \geq L(c,t,A)+1 \geq -1+1=0$. Otherwise, If a scheduled cell c is not a matched scheduled cell and is sent to the crosspoint buffer, there are two possible cases.

1. The input thread of cell c after the arrival phase is not zero, i.e., $IT(c,t,A) \geq 1$, and thus did not participate in the urgent matching. After c is sent to the crosspoint buffer, it has zero input thread, i.e., $IT(c,t,F) = 0$. Since $OC(c,t,F) \geq OC(c,t,A)$, $L(c,t,F) \geq L(c,t,A)+1 \geq -1+1=0$.

2. The input thread of cell c is zero, and thus participated in the urgent matching but it is not matched. By Lemma 1, $L(c,t,F) \geq L(c,t,A)+1 \geq -1+1=0$.

Normal cells. We consider three possible situations depending on whether there is a cell removed from the input buffer and whether a scheduled or a matched cell is removed.

1. No cell leaves the input buffer, which means that there is no scheduled cell as well. According to the input scheduling policy, the crosspoint buffer of each normal cell c is occupied by another cell d with higher output priority. For the cell e delivered to the output buffer of c in the first scheduling phase, which may

be a matched urgent cell or a cell from one of the crosspoint buffers, e must have higher output priority than or the same output priority as that of d. Thus, e has higher output priority than c, and $OC(c,t,F) = OC(c,t,A) + 1$. Since $IT(c,t,F) \leq IT(c,t,A)$, $L(c,t,F) \geq L(c,t,A) + 1 \geq -1 + 1 = 0$.

2. A scheduled cell, including a matched scheduled cell, is removed from the input buffer. For a normal cell c with higher input priority than the scheduled cell, it must have a crosspoint buffered cell d with higher output priority. Based on the analysis for the first situation, we know that $L(c,t,F) \geq L(c,t,A) + 1 \geq -1 + 1 = 0$. On the other hand, for a normal cell c with lower input priority than the scheduled cell, $IT(c,t,F) = IT(c,t,A) - 1$ due to the removal of the scheduled cell. Since $OC(c,t,F) \geq OC(c,t,A)$, it follows that $L(c,t,F) \geq L(c,t,A) + 1 \geq -1 + 1 = 0$.

3. A matched urgent cell leaves the input buffer. Recall that an urgent cell has higher input priority than the scheduled cell. With a similar analysis to that for the second situation, we can obtain that $L(c,t,F) \geq L(c,t,A) + 1 \geq -1 + 1 = 0$.

To sum up, for all the three possible situations, the slackness of the cell increases by one after the first scheduling phase. □

We can have a similar lemma for the second scheduling phase.

Lemma 3. *For any cell c not in the output buffer, if its slackness after the first scheduling phase is larger than or equal to 0, then its slackness after the second scheduling phase is larger than or equal to 1, i.e.,*

$$L(c,t,F) \geq 0 \Rightarrow L(c,t,S) \geq 1$$

The proof is similar to that of Lemma 2 and omitted.

The following lemma gives the slackness of a cell after the arrival phase.

Lemma 4. *For a cell c not in the output buffer, its slackness after the arrival phase is always larger than or equal to -1, i.e.,*

$$L(c,t,A) \geq -1$$

Proof. We prove the lemma by induction.

Base case. After the arrival phase of the first time slot, i.e., slot 1, there is at most one cell arrived at each input buffer. For such a cell c, $IT(c,1,A) = 0$ and $OC(c,1,A) = 0$, resulting in $L(c,1,A) = 0 \geq -1$.

Inductive case. Suppose that after the arrival phase of time slot $t-1$, for any cell c in the input buffer or crosspoint buffer, $L(c,t-1,A) \geq -1$. We will prove that the slackness of the cell c after the arrival phase of time slot t is still larger than or equal to zero, i.e, $L(c,t,A) \geq -1$.

Given $L(c,t-1,A) \geq -1$, we know from Lemma 2 that $L(c,t-1,F) \geq 0$. Furthermore, from Lemma 3, we can obtain $L(c,t-1,S) \geq 1$. In the departure phase of time slot $t-1$, a cell is removed from each output buffer. The output cushion of any cell destined for the output port is decreased by one, in particular, $OC(c,t-1,D) = OC(c,t-1,S) - 1$. Since $IT(c,t-1,D) = IT(c,t-1,S)$, we have $L(c,t-1,D) = L(c,t-1,S) - 1 \geq 1 - 1 = 0$.

During the arrival phase of time slot t, there may be a new cell arriving at each input buffer. In the following, we consider three types of cells that are not in the output buffer after the arrival phase of slot t.

1. c is the newly arrived cell, and its virtual queue is empty. According to the GBVOQ policy, c is inserted at the head of the input priority list, and therefore $IT(c,t,A) = 0$. Since $OC(c,t,A) \geq 0$, we have $L(c,t,A) \geq 0 \geq -1$.

2. c is the newly arrived cell, and its virtual queue is not empty. c is inserted in the input priority list immediately after the last cell d of its virtual queue. Since d is already in the input buffer before the arrival phase of time slot t, based on the above analysis we can obtain that $L(d,t-1,D) \geq 0$. In addition, since $IT(d,t,A) = IT(d,t-1,D)$ and $OC(d,t,A) = OC(d,t-1,D)$, it follows that $L(d,t,A) \geq 0$. For the new cell c, we know that $IT(c,t,A) = IT(d,t,A) + 1$ and $OC(c,t,A) \geq OC(d,t,A)$, and therefore $L(c,t,A) \geq L(d,t,A) - 1 \geq 0 - 1 = -1$.

3. c is an existing cell in the input buffer or crosspoint buffer. In this case, we know that $L(c,t-1,D) \geq 0$. Since $IT(c,t,A) \leq IT(c,t-1,D) + 1$ and $OC(c,t,A) = OC(c,t-1,D)$, we have $L(c,t,A) \geq L(c,t-1,D) - 1 \geq 0 - 1 = -1$.

Thus, for all the three cases, a cell c not in the output buffer after the arrival phase of time slot t has $L(c,t,A) \geq -1$. □

The following lemma describes the slackness of a cell before the departure phase.

Lemma 5. *For a cell c not in the output buffer, its slackness after the second scheduling phase is always larger than or equal to 1, i.e., $L(c,t,S) \geq 1$.*

Proof. Combine Lemma 4, Lemma 2 and Lemma 3. □

We are ready to present the main result of the paper.

Theorem 1. *A buffered crossbar switch with speedup of two can exactly emulate a PIFO OQ switch to provide flow based performance guarantees.*

Proof. Assume that the buffered crossbar switch has successfully emulated the shadow PIFO OQ switch up to time slot $t-1$, and that a cell c departs from an output buffer in the shadow switch at time slot t. We analyze the slackness of c before the departure phase at time slot t in the buffered crossbar switch. Since c departs in the current time slot, there is no cell in the output buffer with higher output priority than c. Thus, $OC(c,t,S) = 0$ and therefore $L(c,t,S) \leq 0$. From Lemma 5, we know that for any cell d in the input buffer or the crosspoint buffer, $L(d,t,S) \geq 1$. This indicates that c must be in the output buffer, and therefore will not be blocked for leaving in the departure phase. In this way, the buffered crossbar switch successfully emulates the shadow switch for time slot t, and the emulation process can continue. As a result, each cell has the same departure time in both switches, and the buffered crossbar switch successfully emulates the shadow PIFO OQ switch.

Recall that PIFO OQ switches with different fair scheduling algorithms can provide different flow based performance guarantees. Thus, buffered crossbar switches can provide the desired flow based performance guarantees by emulating the corresponding PIFO OQ switches.

□

5 Urgent Matching

In this section, we describe the matching algorithms used for the urgent matching sub-phase. As discussed in the previous section, due to the crosspoint blocking, there are some blocked cells in the input buffers that cannot get their slackness increased in the input scheduling and output scheduling sub-phases. The urgent matching sub-phase is therefore introduced to guarantee the on-time delivery of these blocked cells to the output buffers.

From Property 1, we know that there is at most one urgent cell in each input buffer that is destined for a specific output port. On the other hand, for a scheduled cell with zero input thread, it must be the only cell in its input buffer participating in the urgent matching, because it has the highest input priority and there could be no urgent cell from the same input buffer. As a result, we can draw the conclusion that there are at most N^2 cells in the urgent matching.

In order to guarantee the increase of slackness, it is required that for each blocked cell, there must be a matched cell with either higher input priority or higher output priority. The urgent matching problem can be reduced to the stable marriage problem, which can be described as follows.

Given N men and N women, where each person has ranked all members of the opposite sex with a unique number between 1 and N in order of preference, marry the men and women off such that there are no two people of opposite sex who would both rather have each other than their current partners. If there are no such people, all the marriages are stable.

Gale et al. showed that stable marriages exist for any choice of rankings [6], and proposed a solution algorithm that works in an iterative manner and is guaranteed to converge in N^2 iterations.

The urgent matching problem can be converted to the stable marriage problem in the following way. The input ports and output ports are the sets of men and women, respectively, and the input priority of the cells in the same input buffer represents the preference of this input port over the set of output ports, and the output priority of the cells to the same output port represents the preference of this output port over the set of input ports. After such conversion, we can easily obtain Property 2 for the urgent matching.

Due to the special property of the cells participating in the urgent matching, the urgent matching problem can be solved using only N instead of N^2 iterations by a strategy similar to Delay Till Critical (DTC) in [2]. Next, we present sequential and parallel matching algorithms for urgent matching. The sequential matching algorithm is a centralized algorithm, and can be implemented at low cost. The parallel matching algorithm runs faster in the average case, but needs more sophisticated hardware support. For the comparison in the following algorithms, ties are broken randomly.

5.1 Sequential Urgent Matching Algorithm

The sequential urgent matching algorithm can be described as follows. First, select the output port corresponding to the smallest output cushion, and compare the output priority of all the cells to this output port and find the input port corresponding to the highest output priority. It is clear that the cell of the above input-output pair has the smallest output cushion and the highest output priority among all the cells to the same output port, and it also has the highest input priority based on the following reasoning. If the cell is a scheduled cell with zero input thread, there is no cell before it in the input priority list and it must have the highest input priority. Otherwise, it is an urgent cell, which means that there is no scheduled cell from the same input buffer participating in the urgent matching. Because all the urgent cells have the same slackness (-1 for the first scheduling phase and 0 for the second scheduling phase), the one with the smallest output cushion also has the smallest input thread, which means the high input priority. As a result, such a cell must be included in the urgent matching results. Then, we mark the above input-output pair as matched and continue the next iteration of the algorithm. Since in each iteration, there must be one output port marked as matched, the algorithm is guaranteed to converge in N iterations.

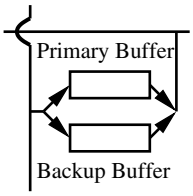

Figure 4. The bypass path can be removed by adding a backup buffer to each crosspoint of the crossbar.

5.2 Parallel Urgent Matching Algorithm

The above algorithm works in a sequential manner, and finishes in N iterations in the best case (given N^2 participating cells) and in the worst case. Next, we present a parallel matching algorithm similar to PIM [1], which requires fewer iterations to converge in the average case. Each iteration of the parallel algorithm has the following three steps:

Request step. Each input port sends a request to every output port for which it has a cell in the urgent matching.

Grant step. Each output port selects the request with the highest output priority to grant.

Accept step. Each input port accepts the grant if the corresponding cell has the highest input priority among all the cells in the input buffer whose output ports have not been matched. The input-output pair is then marked as matched.

Although in the worst case, the parallel urgent matching algorithm still needs N iterations to converge, it is very likely that more than one matched input-output pairs can be generated in a single iteration, and the total number of iterations is greatly reduced. Unfortunately, we have not been able to prove an $O(\log N)$ average number of convergence iterations similar to that in [1].

6 Discussions and Comparisons

In this section, we discuss how to remove the bypass path by adding one more buffer to each crosspoint, and compare our algorithms with existing algorithms in the literature.

6.1 Removing the Bypass Path

The structure of a crosspoint after removing the bypass path and adding the additional buffer is shown in Figure 4. We call the existing crosspoint buffer the primary buffer and the additional one the backup buffer. The backup buffer is only used to temporarily store the matched urgent cell.

For such a switch, there are also two scheduling phases. However, a blocked cell is now defined as a cell whose primary crosspoint buffer is occupied by another cell with lower output priority. In a similar way, the input scheduling sub-phase is conducted for the normal cells independently by each input port, and the urgent matching sub-phase is conducted for the urgent cells and the scheduled cells with zero input thread. The urgent matching results overwrite the input scheduling results, i.e., if there is a matched urgent cell at a specific input port, the scheduled cell of the same input port will not leave the input buffer. Then the matched urgent cells are sent to the backup buffers, and simultaneously the scheduled cells, including the matched scheduled cells, are sent to the primary buffers.

In the output scheduling sub-phase, each matched urgent cell in the backup crosspoint buffer is directly sent to its output buffer, and each of the remaining output ports independently retrieves the cell with the highest output priority from one of the primary crosspoint buffers. In this way, a matched urgent cell is guaranteed to be delivered to the output buffer in the same scheduling phase, and the backup buffer must be empty at the end of each scheduling phase.

6.2 Comparisons with Other Algorithms

We have presented the scheduling algorithms for buffered crossbar switches to provide flow based performance guarantees. There are also some existing works in the literature discussing flow based performance guarantees. In particular, a stable marriage based approach is given in [2] for CIOQ switches, and additional speedup or crosspoint buffers are used in [3] for buffered crossbar switches.

Comparing our approach with that in [2], when there is no crosspoint blocking, the switch only runs the simple and distributed input scheduling and output scheduling algorithms. When there are crosspoint blocked cells, the switch in our case needs to apply the additional urgent matching algorithm, which has the same computational complexity as the scheduling algorithm running in each time slot in [2]. Considering that crosspoint blocking usually only occurs when the first cell of a new flow arrives, the average scheduling time of our approach should be much shorter. The trade-off is that each crosspoint of the crossbar should be equipped with one or two small buffers.

Comparing with the approach that uses additional speedup [3], the switch in our case can achieve 50% higher throughput. Comparing with the approach that uses additional crosspoint buffers [3], the advantage of our approach is the significantly reduced hardware complexity. On the other hand, the trade-off is that when there are crosspoint blocked packets, the switch in our case needs to run the urgent matching algorithm, which has relatively high computational complexity.

7 Conclusions

In this paper, we have studied how to achieve flow based performance guarantees for buffered crossbar switches with

speedup of two. We have added simple bypass paths to the buffered crossbar switches, and presented scheduling algorithms for the switches to emulate PIFO OQ switches. When there are crosspoint blocked cells, the urgent matching is introduced to overcome the crosspoint blocking. We have used the counting method to formally prove the flow based performance guarantees achieved by the buffered crossbar switches. We have also presented the sequential and parallel urgent matching algorithms. Both algorithms converge in N iterations in the worst case, and the latter needs fewer iterations to converge in the average case. Finally, we have designed an alternative implementation scheme for the bypass path by adding one additional buffer at each crosspoint, and compared our algorithms with existing algorithms in the literature.

References

[1] T. Anderson, S. Owicki. J. Saxe and C. Thacker, "High-speed switch scheduling for local-area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319-352, Nov. 1993.

[2] S. Chuang, A. Goel, N. McKeown and B. Prabhkar, "Matching output queueing with a combined input output queued switch," *IEEE INFOCOM'99*, pp. 1169-1178, New York, 1999.

[3] S. Chuang, S. Iyer and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," *Proc. of IEEE INFOCOM 2005*, Miami, FL, March 2005.

[4] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," *IEEE INFOCOM '00*, vol. 2, pp. 556-564, Tel Aviv, Israel, Mar. 2000.

[5] A. Demers, S. Keshav and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM '89*, vol. 19, no. 4, pp. 3-12, Austin, TX, Sept. 1989.

[6] D. Gale, and L.S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly*, vol. 69, pp. 9-15, 1962.

[7] S. He, S. Sun, W. Zhao, Y. Zheng, and W. Gao, "Smooth switching problem in buffered crossbar switches," *ACM SIGMETRICS '05*, pp. 386-387, Banff, Alberta, Canada, Jun. 2005.

[8] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou and N. Chrysos, "Variable packet size buffered crossbar (CICQ) switches," *Proc. IEEE International Conference on Communications (ICC 2004)*, vol. 2, pp. 1090-1096, Paris, France, June 2004.

[9] M. Katevenis and G. Passas, "Variable-size multipacket segments in buffered crossbar (CICQ) architectures," *Proc. IEEE International Conference on Communications (ICC 2005)*, Seoul, Korea, May 2005.

[10] S. Keshav and R. Sharma, "Issues and trends in router design," *IEEE Communications Magazine*, vol. 36, no. 5, pp. 144-151, May 1998.

[11] B. Magill, C. Rohrs and R. Stevenson, "Output-queued switch emulation by fabrics with limited memory," *IEEE Journal on Selected Areas in Communications*, vol 21, no. 4, pp. 606-615, May 2003.

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

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

[14] L. Mhamdi and M. Hamdi, "Output queued switch emulation by a one-cell-internally buffered crossbar switch," *IEEE Globecom 2003*, vol. 7, pp. 3688-3693, San Francisco, CA, Dec. 2003.

[15] N. Ni and L. Bhuyan, "Fair scheduling for input buffered switches," *Cluster Computing*, vol. 6, no. 2, pp. 105-114, Hingham, MA, Apr. 2003.

[16] Rojas-Cessa, E. Oki, Z. Jing and H. J. Chao, "CIXB-1: Combined input-once-cell-crosspoint buffered switch," *IEEE Workshop on High Performance Switching and RoutinG*, Dallas, TX, July 2001.

[17] Rojas-Cessa, E. Oki and H. J. Chao, "CIXOB-k: Combined input-crosspoint-output buffered packet switch," *IEEE Globecom 2001*, San Antonio, Texas, Nov. 2001.

[18] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 375-385, Jun. 1996.

[19] D. Stephens and H. Zhang, "Implementing distributed packet fair queueing in a scalable switch architecture," *IEEE INFOCOM '98*, pp. 282-290, San Francisco, CA, March 1998.

[20] I. Stoica and H. Zhang, "Exact emulation of an output queueing switch by a combined input output queueing switch," *IEEE IWQoS'98*, pp. 218-224, Napa, California, 1998.

[21] J. Turner, "Strong performance guarantees for asynchronous crossbar schedulers," Proc. of IEEE INFOCOM 2006, Barcelona, Spain, Apr. 2006.

[22] X. Zhang and L. Bhuyan, "Deficit round-robin scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, no. 4, pp. 584-594, May 2003.

[23] X. Zhang and L. Bhuyan, "An efficient scheduling algorithm for combined-input-crosspoint-queued (CICQ) switches," *IEEE Globecom 2004*, Dallas, TX, Nov. 2004.