Flow Based Performance Guarantee Scheduling in Buffered Crossbar Switches

Deng Pan and Yuanyuan Yang

***Abstract*—Buffered crossbar switches are a special type of crossbar switches with a small buffer at each crosspoint of the crossbar. Existing research results indicate that they can provide port based performance guarantees with speedup of two, but require significant hardware complexity to provide flow based performance guarantees. In this paper, we present scheduling algorithms for buffered crossbar switches to achieve flow based performance guarantees with speedup of two and one buffer per crosspoint. When there is no crosspoint blocking, only simple and distributed input scheduling and output scheduling are needed. Otherwise, a special urgent matching procedure is necessary to guarantee on-time delivery of crosspoint blocked cells. For urgent matching, we present both sequential and parallel matching algorithms. The parallel version significantly reduces the average number of iterations for convergence, which is verified by simulation. 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 prove the perfect emulation. Finally, we discuss an alternative backup-buffer implementation design to the bypass path, and compare our scheme with existing solutions.**

***Index Terms*—Buffered crossbar switches, cell scheduling, performance guarantees, stable marriage problem.**

I. Introduction

TRADITIONALLY, in switch scheduling a flow is defined as the sequence of packets from an input port to an output port, and port based performance guarantees ensure the desired bandwidth and packet delay of such a flow. However, emerging protocols, such as OpenFlow [19], have made it feasible for switches to process traffic at much finer granularity. For example, an OpenFlow supported switch can flexibly define a flow by any combination of the twelve packet header fields [19]. A flow can thus be the packets generated by a specific application or from a specific virtual machine (VM) of a shared server. Since there may be multiple fine granularity flows for a single input-output pair, port based performance guarantees are no longer sufficient, and it is necessary to provide performance guarantees at the individual flow level. Flow based performance guarantees are particularly important for virtualization based computing environments, such as data centers or GENI-like [14] shared experimental infrastructure. In such an environment, multiple VMs reside in a single physical server, and their traffic shares the same physical network adapter and is correspondingly fed into the same switch port. Flow based performance guarantees are able to isolate traffic of different VMs, and make the shared underlying network infrastructure transparent to the VMs.

Buffered crossbar switches [1], [2], [3] are a special type of crossbar switches, where each crosspoint of the crossbar is equipped with small exclusive buffers. The crosspoint buffers decouple input ports and output ports, and greatly simplify the scheduling process [4], [5], [6]. Thus, buffered crossbar switches are a promising candidate for next generation high speed interconnects. Buffered crossbar switches can provide performance guarantees by emulating push-in-first-output (PIFO) output-queued (OQ) switches [6], [10], [13]. OQ switches have buffers at only output ports, and therefore need speedup of N to achieve 100% throughput [5], which make them not scalable. Specifically, for PIFO OQ switches, the buffer at each output port is organized as a PIFO queue, allowing a packet to be inserted at any position in the queue but removed from only the head of the queue. Many fair queueing algorithms, such as WFQ [15] and DRR [16], can work for PIFO OQ switches to provide performance guarantees. The emulation approach enables buffered crossbar switches to provide the same performance guarantees as OQ switches with reduced hardware complexity due to less speedup.

Chuang, et al. analyzed the capability of buffered crossbar switches to provide performance guarantees in [13]. They showed that speedup of two is sufficient 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 architecture must first be modified with a more complex 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 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. Crosspoint blocking may occur when a new cell arrives at the input buffer. In such a case, 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 one time slot, the additional speedup of

Paper approved by A. Pattavina, the Editor for Switching Architecture Performance of the IEEE Communications Society. Manuscript received November 3, 2010; revised February 26, 2012 and May 18, 2012.

D. Pan is with School of Computing and Information Sciences, Florida International University, Miami, FL 33199, USA (e-mail: pand@cis.fiu.edu).

Y. Yang is with Dept. of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA (e-mail: yang@ece.sunysb.edu).

Digital Object Identifier 10.1109/TCOMM.2012.09.100625

one is guaranteed to completely remove crosspoint blocking.

In this paper, we present scheduling algorithms to provide flow based performance guarantees for buffered crossbar switches with speedup of two and with only one buffer at each crosspoint. As in [13], we consider only fixed length cell scheduling, and make the switch work in a time slot mode. Our main contributions in this paper are summarized as follows. First, we present a buffered crossbar switch structure with a normal path and a bypass path at each crosspoint. The additional bypass path enables the switch to combine the advantages of both a buffered crossbar switch and a combined-input-output-queued switch. Second, we describe a hybrid scheduling algorithm that conducts low-complexity distributed scheduling for normal cases and high-complexity centralized scheduling only for crosspoint blocking cases. Third, we propose a parallel urgent matching algorithm to resolve crosspoint blocking, and show that its average number of iterations for convergence grows logarithmically with the switch size. Fourth, we use the counting method to prove that our scheme can perfectly emulate any PIFO OQ switch, and achieve flow based performance guarantees. Finally, we give an alternative implementation design to remove the bypass path, and compare our scheme with existing solutions.

The rest of the paper is organized as follows. In Section II, we review existing cell scheduling algorithms for buffered crossbar switches. In Section III, we present the switch scheduling algorithms, and in Section IV, we describe the urgent matching algorithms in detail. In Section V, we use the counting method to prove the emulation of PIFO OQ switches. In Section VI, we discuss how to remove the bypass path, and compare our scheme with existing solutions. Finally, in Section VII, we conclude the paper.

II. Related Work

In this section, we give a brief overview of existing cell scheduling algorithms for buffered crossbar switches, which can be divided into two broad categories.

The first group of algorithms target *high throughput*. Among them, [20], [21], [22] propose buffered crossbar based switch architectures and corresponding scheduling algorithms, and demonstrate high throughput by simulation. Further, [23], [24] propose scheduling algorithms for buffered crossbar switches, and prove that they achieve 100% throughput under uniform traffic. Recently, [25] proposes the Stable QUeue Input-output Scheduler with Hamiltonian walk (SQUISH) and Stable QUeuing Implementable Design (SQUID) algorithms, and theoretically proves that they achieve 100% throughput with a finite crosspoint buffer and without crossbar speedup for any admissible traffic. [26] further extends the design with distributed implementation, and proposes the DIStributed QUeue input-Output (DISQUO) algorithm, which also guarantees 100% throughput for any admissible traffic. The schemes in [27] and [28] schedule mixed multicast and unicast traffic, and show that they achieve high throughput by simulation and analysis, respectively. Compared with the above algorithms, our proposed scheme can provide strong performance guarantees, such as guaranteed bandwidth or packet delay, which are important for applications with QoS requirements.

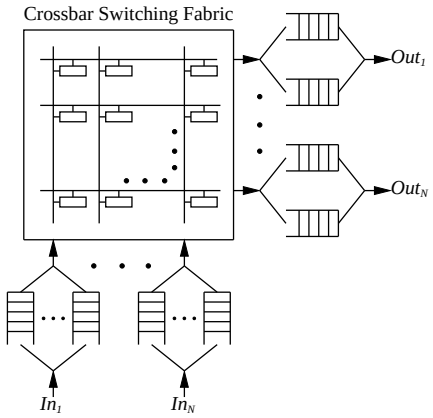

Fig. 1. Structure of buffered crossbar switch.

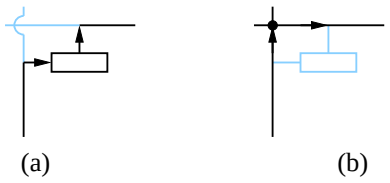

Fig. 2. Two possible transmission paths. (a) Normal path. (b) Bypass path.

The second group of algorithms intend to provide *performance guarantees*. The algorithms in [11], [12] are proved to emulate PIFO OQ switches at the port level. The smoothed multiplexer (sMUX) algorithm in [4] uses a TDMA approach to provide port based performance guarantees. Compared with those designs, our proposed scheme provides performance guarantees at finer granularity, for individual flows of the same input-output pair. [13] further proposes algorithms to emulate PIFO OQ switches at the flow level, at the cost of speedup of three or N^3 crosspoint buffers. Compared with the algorithms in [13], our scheme achieves the same flow based performance guarantees with reduced hardware complexity, i.e., speedup of two and N^2 crosspoint buffers.

III. Switch Scheduling

In this section, we present our scheduling algorithms to emulate PIFO OQ switches at the flow level.

A. Switch Structure

The switch structure considered in this paper is illustrated in Fig. 1. N input ports and N output ports are connected by a crossbar switching fabric with speedup of two. Packets are buffered at input ports, output ports, and crosspoints. To provide performance guarantees for individual flows, input and output buffers are organized on a per-flow basis, i.e., a logical virtual queue to store packets of a single flow in their arrival order. Each crosspoint has a small exclusive buffer, which can store a single cell. Following the convention, assume that in each time slot, at most one cell can enter an input buffer or leave an output buffer. Since the crossbar has speedup of two, it can retrieve two cells from each input buffer and deliver two cells to each output buffer in a single time slot.

The crossbar has two possible paths to transmit a cell from the input buffer to the output buffer, which we call the normal path and bypass path, respectively. For the normal path, as shown in Fig. 2(a), a cell is first sent from the input buffer to the crosspoint buffer, from where it will be retrieved to the

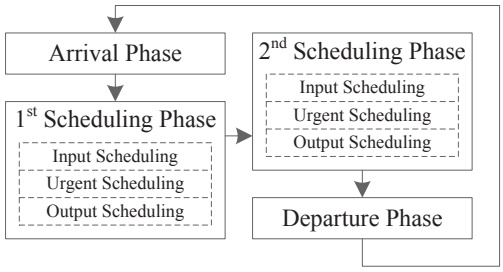

Fig. 3. Operation phases of buffered crossbar switch.

output buffer. For the bypass path, as illustrated in Fig. 2(b), a cell is directly transmitted from the input buffer to the output buffer without being stored at the crosspoint. In a single time slot, only one of the two paths is available.

B. Emulation of PIFO OQ Switches

Our approach to achieving performance guarantees is to emulate the scheduling algorithms of PIFO OQ switches. We call an OQ switch that we intend to emulate the *shadow switch*. A successful emulation means that, for any incoming traffic, the departure time of each cell in our considered switch is the same as that in the shadow switch. Define the output priority of a cell to be its departure time in the shadow OQ switch. For two cells to the same output port, the one with higher output priority leaves the output buffer first. Define the input priority of a cell by the Group by VOQ (GBVOQ) policy [10], which we will describe in more detail in Section III-C1. For two cells of the same input port whose crosspoint buffers are both empty, the one with higher input priority leaves the input buffer first.

We will use the counting method [7], [10], [11], [12], [13] to analyze the scheduling algorithms, and thus need the following notations. The *output cushion* of a cell is the number of cells that are in its destination output buffer and have higher output priority. The *input thread* of a cell is the number of cells that are in the same input buffer and have higher input priority. If the cell is in the crosspoint buffer, we define its input thread to be zero. The *slackness* of a cell equals its output cushion minus its input thread. Intuitively, slackness indicates the urgency level to transmit a cell to its output buffer, and a cell with non-negative slackness is guaranteed to depart from its output buffer on time.

C. Scheduling Phases

For easy description, we divide the operation of a buffered crossbar switch into four phases: arrival, first scheduling, second scheduling, and departure. In particular, each of the two scheduling phases consists of three sub-phases: input scheduling, urgent matching, and output scheduling, in which urgent matching is not necessary if there is no crosspoint blocking. Fig. 3 illustrates the operation phases and sub-phases in a time slot. Next, we describe each phase in detail.

1) Arrival Phase: In the arrival phase, a new cell may arrive at each input buffer, and it will be put at the end of the corresponding virtual queue. 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. Otherwise, it is inserted in the input priority list immediately after the last cell of its virtual queue.

2) Scheduling Phase: In each of the two scheduling phases, the crossbar makes a scheduling decision to retrieve up to one packet from every input buffer and deliver up to one packet to every output buffer. We only consider the head cell of a virtual queue in the scheduling phase, because cells of the same flow are in a single queue and the head cell always leaves first. For a head cell, if its crosspoint is occupied by another cell with lower output priority, we call it a *blocked cell*, and otherwise a *normal cell*.

In the *input scheduling sub-phase*, 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. The cell selected is called a *scheduled cell*. It may occur that there is no scheduled cell in an input buffer, because all the crosspoint buffers have been occupied. If there are no blocked cells, the switch can now proceed to the output scheduling sub-phase, otherwise the urgent matching sub-phase is necessary.

In the *urgent matching sub-phase*, the crossbar arranges the transmission of blocked cells. Urgent matching considers a special type of blocked cells called *urgent cells*. A blocked cell is an urgent cell if: *a)* its slackness is -1 for the first scheduling phase or 0 for the second scheduling phase, *b)* it has the highest input priority among all such cells in the same input buffer destined for the same output port, *c)* it has higher input priority than the scheduled cell, and *d)* it has higher output priority than all the crosspoint buffered packets to its destination output port. Besides urgent cells, urgent matching also involves the scheduled cells with zero input thread. We define a *matched cell* to be a cell selected to transmit in urgent matching. For easy reading, we defer the description of urgent matching algorithms to Section IV.

Now we have obtained the input scheduling and urgent matching results, and the cells in the input buffers are arranged to leave 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 buffer has a scheduled cell but no matched cell, the scheduled cell is also sent to the crosspoint buffer. All the cells leave their input buffers at the same time, with matched urgent cells taking bypass paths and other cells taking normal paths.

Next, in the *output scheduling sub-phase*, each output port independently retrieves the cell with the highest output priority from one of its crosspoint buffers. If there was an urgent matching 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.

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 for departure.

IV. Urgent Matching

In this section, we describe the algorithms used for the urgent matching sub-phase, which schedules the transmission of blocked cells. By the definition of urgent cells, 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 urgent matching, because it has the highest input priority and there could be no urgent cell from the same input buffer. Thus, we can draw the conclusion that there are at most N^2 cells in urgent matching.

We reduce the urgent matching problem 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 the 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. The reduction process is as follows. 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. By the reduction process, we have Property 1 and further Lemma 1 for urgent matching.

Property 1: 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.

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: We analyze each of the two cases of Property 1. In the first case, since there is a matched cell from the same input with higher input priority, regardless 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, regardless 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. ■

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 [17] iterations by a strategy similar to Delay Till Critical (DTC) in [10]. Next, we present sequential and parallel matching algorithms for urgent matching. For the comparison in the following algorithms, ties are broken randomly.

A. Sequential Urgent Matching Algorithm

The sequential urgent matching algorithm works as follows. First, select the output port corresponding to the smallest output cushion, 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 implies 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.

B. Parallel Urgent Matching Algorithm

The above algorithm works in a sequential manner, and completes in N iterations in both the best case (given N^2 participating cells) and the worst case. Next, we present a parallel matching algorithm similar to Parallel Iterative Matching (PIM) [18], 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.

C. Simulation Results

We now present simulation results to compare the sequential and parallel urgent matching algorithms. In the simulations, we considered the switch size N from 2^1 to 2^{10}. We assumed that the number of cells participating in the urgent matching is N^2, and assigned random values as the input priority and output priority of the participating cells. For each switch size,

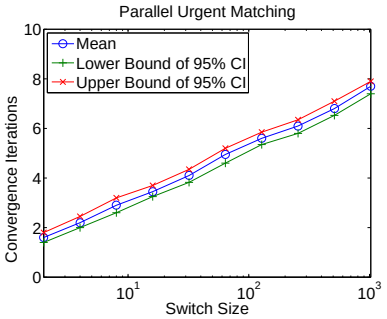

Fig. 4. Number of iterations for parallel urgent matching to converge.

we conducted 20 simulation runs, and then calculated the average number of iterations to converge and its lower and upper bounds of 95% confidence interval. Fig. 4 plots the simulation results. As can be seen, the number of iterations for the parallel urgent matching algorithm to converge grows logarithmically with the switch size. Even for a switch size of 1024, it has 95% of possibility to converge with 7.4 to 7.9 iterations. Compared with the linear average convergence iterations of the sequential algorithm, we can draw the conclusion that the parallel algorithm is effective in reducing the running time.

V. Achieving Flow Based Performance Guarantees

In this section, we show that a buffered crossbar switch running the proposed scheduling algorithms achieves flow based performance guarantees, in the sense that it can perfectly emulate any PIFO OQ switch. We first analyze each scheduling phase and then present the main result. For easy representation, we use $OC(c,t,*)$, $IT(c,t,*)$, and $L(c,t,*)$ to denote the output cushion, input thread, and slackness, respectively, of a cell c at time slot t after the $*$ phase, where $*$ may be either A, F, S, or D representing the Arrival, First scheduling, Second scheduling, or Departure phase accordingly.

We have 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 greater than or equal to -1, then its slackness after the first scheduling phase is greater 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 subphase, 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 phases. The relationship between these cell types is illustrated in Fig. 5. We prove that the lemma holds for each type of cells.

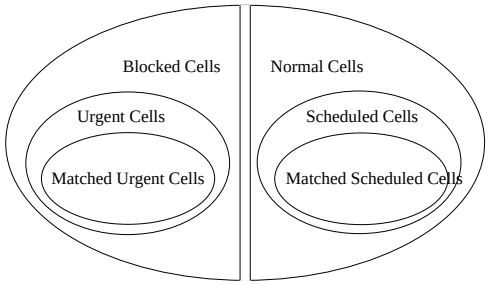

Fig. 5. Relationship of different types of cells.

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 cases. 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 an 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, in the first case, nothing needs to be done to ensure $L(c,t,F) \geq 1$, and in other three cases, the slackness of the cell increases 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 no longer need to consider its slackness. 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 non-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 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 case, 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. By a similar analysis to that for the second case, we can obtain that $L(c,t,F) \geq L(c,t,A)+ 1 \geq -1 + 1 = 0$.

To sum up, in all the three possible cases, 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 greater than or equal to 0, its slackness after the second scheduling phase is greater 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 thus 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 greater than or equal to -1, i.e., $L(c,t,A) \geq -1$.

Proof: We prove the lemma by induction on the number of time slots.

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. Next 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 non-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, in 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 greater than or equal to 1, i.e., $L(c, t, S) \geq 1$.

Proof: Combine Lemma 4, Lemma 2, and Lemma 3. ■

We are now ready to present the main result.

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. ■

VI. 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 ones in the literature.

A. Removing the Bypass Path

To apply our algorithms to a wider range of switches, we present an alternative implementation to remove the bypass path by adding one more buffer at each crosspoint. Although the bypass path in our switch structure is unique, many existing buffered crossbar switch designs contain more than one buffer at each crosspoint, and they may run the proposed algorithms with the alternative implementation. The structure of a crosspoint after removing the bypass path and adding the additional buffer is shown in Fig. 6. 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.

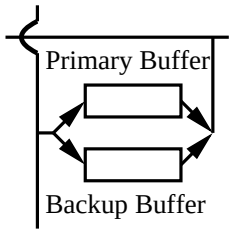

Fig. 6. Removing bypass path by adding a backup buffer to each crosspoint.

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. 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.

B. Comparisons with Other Algorithms

We compare our scheme with existing scheduling algorithms for buffered crossbar switches, and summarize the results in Table 1, where M is the maximum number of flows for an input-output pair.

First, we look at SQUISH [25], SQUID [25], and DISQUO [26]. All the three algorithms have low hardware complexity and time complexity. However, since they are designed to achieve high throughput, they cannot provide strong performance guarantees. Note that since our scheme and all emulation based algorithms can emulate arbitrary PIFO OQ switches, they may also achieve 100% throughput by emulating a specific PIFO OQ switch with 100% throughput.

Next, compared with the designs in [13] with additional speedup or crosspoint buffers, the advantage of our scheme is the reduced hardware complexity with only speedup of two and one buffer per crosspoint. The trade-off is that, when there is crosspoint blocking, our scheme needs to run the urgent matching algorithm with high time complexity. Fortunately, the parallel urgent matching algorithm reduces the average number of iterations for convergence to $O(\log N)$.

TABLE 1
COMPARISON WITH EXISTING ALGORITHMS.

	SQUISH [25]	SQUID [25]	DISQUO [26]	Additional hardware [13]	Stable marriage [10]	Our scheme
Throughput	100%	100%	100%	100%	100%	100%
Flow based perf. guarantees	No	No	No	Yes	Yes	Yes
Crossbar speedup	1	1	1	3/2	2	2
# of crosspoint buffers	N^2	N^2	N^2	N^2/N^3	0	N^2
Time complexity	$O(\log N)$	$O(\log N)$	$O(1)$	$O(\log M + \log N)$	$O(\log M + N \log N)$	$O(\log M + N \log N)$
Distributed scheduling	No	No	Yes	Yes	No	Hybrid
Avg. convergence #	N/A	N/A	N/A	N/A	O(N)	$O(\log N)$

Finally, compared with the stable marriage based algorithm in [10], our scheme has two advantages. First, when there is no crosspoint blocking, our scheme runs simple and distributed scheduling with low time complexity. Second, when there is crosspoint blocking, our parallel urgent matching algorithm ensures fast convergence. On the other hand, the advantages of the algorithm in [10] include small speedup of two and no need of crosspoint buffers. The algorithm in [10] is designed for combined-input-output-queued switches, which have no integrated buffers at the crosspoints of the crossbar.

VII. CONCLUSIONS

In this paper, we have studied how to achieve flow based performance guarantees for buffered crossbar switches with speedup of two and one buffer per crosspoint. We have added simple bypass paths to buffered crossbar switches, and presented scheduling algorithms for the switches to emulate PIFO OQ switches. When there is no crosspoint blocking, only distributed scheduling is necessary; otherwise the urgent matching sub-phase is introduced to transmit crosspoint blocked cells. We have also presented the sequential and parallel urgent matching algorithms. The latter needs much fewer iterations to converge in the average case, and its effectiveness is verified by simulations. We have used the counting method to formally prove the flow based performance guarantees achieved by the proposed scheme. Finally, we have designed an alternative implementation for the bypass path by adding one additional buffer at each crosspoint, and compared our scheme with existing algorithms in the literature.

ACKNOWLEDGMENTS

This work was partially supported by the US National Science Foundation under grant numbers CNS-1117016 and CCF-0915823.

REFERENCES

[1] G. Kornaros, "BCB: a buffered crossbar switch fabric utilizing shared memory," *EUROMICRO 2006*, Aug. 2006.
[2] L. Mhamdi, C. Kachris and S. Vassiliadis, "A reconfigurable hardware based embedded scheduler for buffered crossbar switches," *14th ACM/SIGDA International Symp. Field Programmable Gate Arrays*, pp. 143-149, Monterey, CA, Feb. 2006.
[3] I. Papaefstathiou, G. Kornaros and N. ChrysosUsing, "Buffered crossbars for chip interconnection," *17th Great Lakes Symp. VLSI*, pp. 90-95, Stresa-Lago Maggiore, Italy, Mar. 2007.
[4] S. He, et al., "On guaranteed smooth switching for buffered crossbar switches," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 718-731, June 2008.
[5] D. Pan and Y. Yang, "Localized independent packet scheduling for buffered crossbar switches," *IEEE Trans. Comput.*, vol. 58, no. 2, pp. 260-274, Feb. 2009.
[6] J. Turner, "Strong performance guarantees for asynchronous crossbar schedulers," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1017-1028, Aug. 2009.
[7] 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, CA, 1998.
[8] D. Pan and Y. Yang, "FIFO-based multicast scheduling algorithm for virtual output queued packet switches," *IEEE Trans. Comput.*, vol. 54, no. 10, Oct. 2005.
[9] D. Pan and Y. Yang, "Credit based fair scheduling for packet switched networks," *IEEE INFOCOM 2005*, vol. 2, pp. 843-854, Mar. 2005.
[10] S. Chuang, A. Goel, N. McKeown and B. Prabhkar, "Matching output queueing with a combined input output queued switch," *IEEE INFOCOM 1999*, pp. 1169-1178, NY, 1999.
[11] 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.
[12] B. Magill, C. Rohrs and R. Stevenson, "Output-queued switch emulation by fabrics with limited memory," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 606-615, May 2003.
[13] S. Chuang, S. Iyer and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," in *Proc. IEEE INFOCOM 2005*, Miami, FL, Mar. 2005.
[14] Global Environment for Network Innovations, [Online]. Available: http://www.geni.net
[15] A. Demers, S. Keshav and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM 1989*, vol. 19, no. 4, pp. 3-12, Austin, TX, Sept. 1989.
[16] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 375-385, June 1996.
[17] D. Gale, and L.S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly*, vol. 69, pp. 9-15, 1962.
[18] 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.
[19] N. McKeown, et al., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69-74, Apr. 2008.
[20] Rojas-Cessa, E. Oki, Z. Jing and H. J. Chao, "CIXB-1: Combined input-once-cell-crosspoint buffered switch," *IEEE Workshop High Performance Switching Routing*, Dallas, TX, July 2001.
[21] Rojas-Cessa, E. Oki and H. J. Chao, "CIXOB-k: Combined input-crosspoint-output buffered packet switch," *IEEE Globecom 2001*, San Antonio, TX, Nov. 2001.
[22] L. Mhamdi and M. Hamdi, "MCBF: A high-performance scheduling algorithm for buffered crossbar switches," *IEEE HPSR*, Torino, Italy, June 2003.
[23] M. Berger, "Delivering 100% throughput in a buffered crossbar with round robin scheduling," *IEEE HPSR*, Poznan, Poland, June 2006.
[24] T. Javidi, R. Magill and T. Hrabik, "A high-throughput scheduling algorithm for a buffered crossbar switch fabric," *IEEE ICC* Helsinki, Finland, June 2001.
[25] Y. Shen, S. Panwar and H.J. Chao, "SQUID: A Practical 100% Throughput Scheduler for Crosspoint Buffered Switches," *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, pp. 1119-1131, Aug. 2010.
[26] S. Ye, Y. Shen and S. Panwar, "DISQUO: A distributed 100% throughput algorithm for a buffered crossbar switch," *IEEE HPSR*, Richardson, TX, June 2010.
[27] 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, June 2009.
[28] C. Chang, Y. Hsu, J. Cheng and D. Lee, "A dynamic frame sizing algorithm for CICQ switches with 100% throughput," *IEEE INFOCOM 2009*, Rio de Janeiro, Brazil, Apr. 2009.

Deng Pan received the BS and MS degrees in computer science from Xi'an Jiaotong University, China, in 1999 and 2002, respectively, and the PhD degree in computer science from the State University of New York at Stony Brook, in 2007. He is currently an assistant professor in the School of Computing and Information Sciences, Florida International University. His research interests include high performance switch architecture and high speed networking. He is a member of the IEEE.

Yuanyuan Yang is a Professor of Electrical & Computer Engineering and Computer Science at Stony Brook University (SUNY at Stony Brook), and the Director of Communications and Devices Division at New York State Center of Excellence in Wireless and Information Technology (CEWIT). She received her PhD degree in computer science from Johns Hopkins University, Baltimore, Maryland, in 1992. Dr. Yang's research interests include interconnection networks, date center networks, wireless/mobile networks, optical networks, and high-speed networks. She has authored or coauthored more than 250 research articles in leading refereed journals and conferences on these topics. She is also an inventor/co-inventor of seven U.S. patents in the area of interconnection networks. She is a Fellow of the IEEE.