Localized Asynchronous Packet Scheduling for Buffered Crossbar Switches *

Deng Pan
Dept. of Computer Science
State University of New York
Stony Brook, NY 11794, USA
pandeng@cs.sunysb.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 crossbar switches. In such a switch, besides normal input queues and output queues, a small buffer is associated with each crosspoint. Due to the introduction of crosspoint buffers, output and input contention is eliminated, and the scheduling process for buffered crossbar switches is greatly simplified. Moreover, crosspoint buffers enable the switch to work in an asynchronous mode and easily schedule and transmit variable length packets. Compared with fixed length packet scheduling or cell scheduling, variable length packet scheduling, or packet scheduling for short, has some unique advantages: higher throughput, shorter packet latency and lower hardware cost. In this paper, we present a fast and practical scheduling scheme for buffered crossbar switches called Localized Asynchronous Packet Scheduling (LAPS). With LAPS, an input port or output port makes scheduling decisions solely based on the state information of its local crosspoint buffers, i.e., the crosspoint buffers where the input port sends packets to or the output port retrieves packets from. The localization property makes LAPS suitable for a distributed implementation and thus highly scalable. Since no comparison operation is required in LAPS, scheduling arbiters can be efficiently implemented using priority encoders, which can make arbitration decisions quickly in hardware. Another advantage of LAPS is that each crosspoint needs only L (the maximum packet length) buffer space, which minimizes the hardware cost of the switches. We also theoretically analyze the performance of LAPS, and in particular we prove that LAPS achieves 100% throughput for any admissible traffic with speedup of two. Finally, simulations are conducted to verify the analytical results and measure the performance of LAPS.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Packet-switching networks

General Terms

Algorithms, Design.

Keywords

Buffered crossbar switches, packet scheduling, 100% throughput, priority encoders.

*This research was supported in part by the U.S. National Science Foundation under grant numbers CCR-0073085 and CCR-0207999.

ANCS'06, December 3–5, 2006, San Jose, California, USA.

1. INTRODUCTION

Crossbar switches provide non-blocking capability and overcome the bandwidth limitation of bus based switches. They have long been the preferred structures for high speed switches and routers. With the ever increasing demand for more bandwidth and higher throughput, it has become a more and more challenging task to design high performance crossbar switches and efficient scheduling algorithms.

For crossbar switches, packets can be buffered at either output ports, input ports, or crosspoints of the crossbar. Output queued (OQ) switches only have buffer space at the output side, and new incoming packets must be immediately transferred through the crossbar and stored in the output queues. OQ switches achieve 100% throughput, and can provide different levels of performance guarantees by running various fair scheduling algorithms, such as WFQ [1], DRR [2], and FMCF [3], at each output port. However, in order for an $N \times N$ switch to achieve 100% throughput, speedup of N is required. In other words, the crossbar must have N times bandwidth as that of an input port or output port. Thus, OQ switches are difficult to scale. On the contrary, input queued (IQ) switches only have buffer space at the input side, and need no speedup. In order to make fast scheduling decisions, iterative maximal matching algorithms, such as PIM [4], iSLIP [5], and DRRM [6], were proposed for IQ switches, which can quickly converge on a maximal matching in multiple iterations. IQ switches are popular on the market due to their economical hardware architectures and efficient scheduling algorithms. Unfortunately, until now IQ switches are found to be able to achieve 100% throughput only when work with maximum matching algorithms or their variants, such as MWM [7], which have high time complexity [27] [28] and are not feasible for high speed scheduling. In order to combine the advantages of both OQ switches and IQ switches, combined input-output queued (CIOQ) switches make a tradeoff between the crossbar speedup and the complexity of the scheduling algorithms. They usually have fixed small speedup of two, and thus need buffer space at both the input side and output side. It has been shown that CIOQ switches with speedup of two can achieve 100% throughput with any maximal scheduling algorithms [8] [9], and can also emulate OQ switches with more complex algorithms [15] [16].

For buffering at the input side of a switch, usually the virtual

output queued (VOQ) buffering technique [7] is used, since the traditional single FIFO queue suffers from the head of line (HOL) blocking, i.e., even though the destination output ports of the packets behind the HOL packet may be free, these packets cannot be scheduled to transmit because the HOL packet is blocked. It was proved in [10] that the HOL blocking limits the maximum throughput of the switch to about 58.6%. On the contrary, VOQ buffering maintains a logically separate queue or virtual queue for each output port at every input port, so that a packet will no longer be held up by another packet ahead of it that goes to a different output port.

With the development of modern VLSI technology, it has been feasible to integrate on-chip memory to the crossbar switching fabric. Buffered crossbar switches, or combined input-crosspoint-output queued (CICOQ) switches [17] [18] [25], are a special type of CIOQ switches, where each crosspoint of the crossbar is equipped with a small buffer. Due to the introduction of crosspoint buffers, the scheduling process is greatly simplified. First, output contention is eliminated. Assume that multiple input ports have packets destined to the same output port. In a traditional unbuffered crossbar switch, since packets are directly sent from input ports to output ports, the transmission has to be carefully scheduled so that no two input ports will simultaneously send packets to the same output port. On the other hand, in a buffered crossbar switch, these packets can be first sent to the crosspoint buffers, and then the output port can retrieve the packets from the crosspoint buffers one by one. Furthermore, since output contention has been eliminated, different input ports no longer need to cooperate with each other, and their scheduling can be conducted independently. As a result, the complexity of the scheduling algorithm is greatly reduced.

Previous research on scheduling algorithms for crossbar switches mainly focused on fixed length packet scheduling or cell scheduling [11]. Without crosspoint buffers, packets have to be directly sent from input ports to output ports. In order to maximize throughput and make fast scheduling decisions, all the scheduling and transmission units must have the same length, and all the input ports and output ports have to work in a synchronized mode, i.e., all input ports send cells at the same time, and all output ports receive cells at the same time. When variable length packets arrive, they must be segmented into fixed length cells at input ports. The cells are then used as the scheduling units and transmitted to output ports, where they are reassembled into original packets and sent to the output lines. In contrast, buffered crossbar switches have removed the necessity of synchronization due to crosspoint buffers. They can work in an asynchronous mode, and directly handle variable length packets. In other words, each input port or output port periodically chooses a packet with an arbitrary length to send to or receive from the crosspoint buffer, and does not need to wait for other input ports or output ports.

Compared with cell scheduling, variable length packet scheduling, or packet scheduling for short, has some unique advantages. First, packet scheduling can better utilize available bandwidth and achieve higher throughput. For cell scheduling, when a packet is segmented into cells, its length may not be a multiple of the cell length, and the last segment has to be padded with empty bits to reach the cell length. The padding bits do not contain useful information, and waste transmission capacity of the switch. In the worst case, if all packets happen to have a slightly longer length than the cell length, each packet has to be segmented into two cells, and the switch can only achieve about a half of the maximum throughput. Second, since there is no segmentation and reassembly in packet scheduling, newly arrived packets at input ports can be immediately transferred through the crossbar, and similarly, transmitted packets at the output ports can be immediately sent to the output lines. Especially when the load is light, packets have short queueing delay, and the time for segmentation and reassembly can hardly be overlapped with other waiting time. Therefore, packet scheduling reduces the latency that a packet experiences in the switch. Third, no extra buffer space is needed at the input or output side to segment and reassemble the packets, which lowers hardware cost. Finally, cell scheduling can be regarded as a special case of packet scheduling, or in other words, packet scheduling can also handle fixed length cells.

Two packet scheduling algorithms for asynchronous buffered crossbar switches, Packet GVOQ (PGV) and Packet LOOFA (PLF), were proposed in [12], and their performance guarantees were analyzed. It was proved that, with speedup of two and $2L$ or more buffer space at each crosspoint, where L is the maximum packet length, PGV and PLF can provide work-conserving guarantees or emulate push-in-first-out (PIFO) scheduling algorithms for OQ switches. In order to be work-conserving, the algorithms proposed in [12] usually impose an order on buffered packets, and make scheduling decisions by sorting the packets. With slightly more buffer space at each crosspoint, they can emulate any PIFO fair scheduling algorithm for OQ switches by ordering the packets based on the departure sequence of the packets in the reference algorithm, and thus provide bandwidth and delay guarantees. Scheduling algorithms delivering strong performance guarantees are certainly important, especially considering that nowadays many broadband based multimedia applications have quality of service (QoS) requirements. On the other hand, it is also worth studying scheduling algorithms that are simple and easy to implement.

The objective of this paper is to design packet scheduling algorithms for buffered crossbar switches with low time complexity and easy hardware implementation. We present a packet scheduling scheme called Localized Asynchronous Packet Scheduling (LAPS). LAPS conducts scheduling in an asynchronous and distributed mode. An input port or output port make scheduling decisions solely based on the state information of its local crosspoint buffers, i.e., the crosspoint buffers where the input port sends packets to or the output port retrieves packets from. Since no comparison operation is required, scheduling arbiters can be efficiently implemented using priority encoders, which can make decisions quickly in hardware. LAPS requires only L buffer space at each crosspoint. Considering that on-chip buffers are expensive resources, LAPS minimizes the hardware cost of switches. We also theoretically analyze the performance of LAPS, and in particular prove that LAPS achieves 100% throughput for any admissible traffic with speedup of two. Finally, simulations are conducted to verify the analytical results and to measure the performance of LAPS.

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 present the LAPS scheduling scheme, analyze its performance and discuss its hardware implementation. In Section 4, we conduct simulations to verify the analytical results obtained in Section 3, and test the performance of LAPS. In Section 5, we conclude the paper.

2. RELATED WORK

Scheduling algorithms for buffered crossbar switches are generally designed with two possible objectives: to achieve high throughput or to emulate scheduling algorithms for OQ switches. The latter is a stronger requirement than the former, i.e., an algorithm that emulates an OQ switch based algorithm usually delivers 100% throughput, but the reverse is not always true. On the other hand, if 100% throughput is the only objective, an algorithm can be simpler or have less time complexity.

A buffered crossbar switch architecture called CIXB-1 was proposed in [18], 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 [19] 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 [20]. 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) [21] 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 [21], 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 [24]. It is proved that a buffered crossbar switch with speedup of two satisfying non-negative slackness (NNS) insertion and lowest time to live (LTTL) blocking, and LTTL fabric scheduling can exactly emulate an OQ switch with PIFO scheduling policies. In particular, it is shown that the GBVOQ_OCF scheduling algorithm can exactly emulate an FIFO OQ switch, and the GBFG_SP scheduling algorithm can exactly emulate a strict priority OQ switch. In [22], 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 an OQ switch with speedup of two. [23] studied practical scheduling algorithms for buffered crossbar switches. It is shown that with speedup of two, a buffered crossbar switch can mimic the restricted PIFO-OQ switch (a PIFO-OQ switch with the restriction that the cells of an input-output pair depart the switch in the same order as they arrive), regardless of the incoming traffic pattern, and that with speedup of three, a buffered crossbar switch can mimic an arbitrary PIFO-OQ switch 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 consider also cell scheduling, but are designed to emulate scheduling algorithms for OQ switches.

A buffered crossbar switch architecture supporting packet scheduling was proposed in [25]. 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 [26]. 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 [12], and two algorithms were designed based on existing cell scheduling algorithms. It is theoretically proved that, with speedup of two, the Packet GVOQ scheduling algorithm provides work-conserving guarantees with $2L$ crosspoint buffer space and can emulate a PIFO scheduling algorithm for OQ switches with $5L$ crosspoint buffer space. The Packet LOOFA 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. [17] 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.

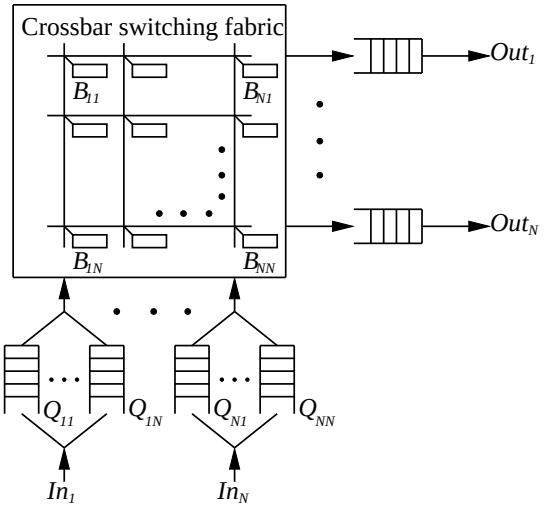


Figure 1: The structure of buffered crossbar switches.

3. LOCALIZED ASYNCHRONOUS PACKET SCHEDULING

In this section, we present our new Localized Asynchronous Packet Scheduling (LAPS) scheme, analyze its performance, and discuss its hardware implementation.

The switch structure that we consider is illustrated in Figure 1. N input ports and N output ports are connected by a crossbar switching fabric, which has speedup of two. We denote the bandwidth of an input port or output port by R, and the crossbar has bandwidth $2R$. An input port has N virtual queues to store the packets destined to different output ports. Each crosspoint has an exclusive buffer of size L. Depending on the granularity level of performance guarantees, an output port may have a single queue, or multiple logical queues. For example, if bandwidth is to be fairly allocated among input ports, packets in output ports should be buffered on a per input port basis, and if different flows require different packet delay guarantees, an output port needs to set up as many queues as the number of flows. Different input ports and output ports work independently and asynchronously. After a packet arrives at the switch, it is first stored in the input queue. The packet is sent from the input queue to the crosspoint buffer, and then from the crosspoint buffer to the output queue and finally delivered to the output line.

Based on the locations of the packets to be scheduled, there are three types of scheduling involved in such a buffered crossbar switch, which we call input scheduling, crossbar scheduling, and output scheduling, respectively. In input scheduling, an input

port selects one of the virtual queues, and sends its head packet to the crosspoint buffer. In crossbar scheduling, an output port selects one of the crosspoint buffers, and retrieves the buffered packet to the output queue. In output scheduling, an output port selects a buffered packet from the output queue, and sends the packet to the output line.

Over the last decade, output scheduling has been well studied, and a lot of output scheduling algorithms have been proposed, such as WFQ [1] and DRR [2]. By using different approaches, the algorithms provide different performance guarantees and have different time complexity. It is reported in [13] that a fundamental tradeoff exists between the performance guarantees that an algorithm can provide and its time complexity. It should be noted that output scheduling algorithms usually do not affect the throughput performance as long as they are work-conserving. In other words, if an output scheduling algorithm sends packets to the output line whenever there is a packet in the output queue, 100% throughput can be achieved given that the input scheduling algorithm and crossbar scheduling algorithm deliver packets to the output queue in time. In the rest of the paper, we will mainly consider input scheduling and crossbar scheduling, and adopt a simple FIFO algorithm for output scheduling, which is work-conserving.

3.1 Algorithm Description

Input scheduling and crossbar scheduling of LAPS are conducted in an asynchronous and distributed manner, and they only rely on the state information of local crosspoint buffers. Local crosspoint buffers of an input port or output port are the crosspoint buffers that the input port sends packets to or the output port receives packets from.

In input scheduling, when the transmission channel of an input port to the crosspoint buffers is idle, the input port selects one of its backlogged virtual queues whose corresponding crosspoint buffer is empty, and sends the head packet to the crosspoint buffer. When there are multiple eligible virtual queues, different arbitration rules can be used, such as fixed priority, random priority, or round robin. In particular, the round robin rule is able to avoid starvation by alternatively giving each virtual queue the highest priority. We will see later in this section that any work-conserving rule is able to achieve 100% throughput. It should be noted that since the crossbar has speedup of two, the packet is transferred from the virtual queue to crosspoint buffer with bandwidth of $2R$. After the last bit of the packet is sent to the crosspoint buffer, the scheduling and transmission process is repeated again. Crossbar scheduling is similar to input scheduling. When the transmission channel of an output port for receiving packets from the crosspoint buffers is idle, the output port selects a crosspoint buffered packet and saves it in its output queue. The transmission rate from the crosspoint buffer to the output queue is also $2R$, and different arbitration rules can be used as well.

In order to reduce the packet latency and increase the switch throughput, we use cut-through switching in the crossbar. In other words, when a packet is being sent from the virtual queue to the crosspoint buffer, if the transmission channel to its output is idle, the packet can be directly sent to the output queue without waiting for the whole packet to be buffered at the crosspoint buffer. Similarly, if the output port has cut-through switching capability, a packet can also be immediately sent to the output line as soon as its first bit arrives at the output queue. Thus, with the cut-through technique, it is possible that a packet is directly sent from the input queue to the output line without being fully buffered anywhere on the way. Since the bandwidth from the input queue to the crosspoint buffer and from the crosspoint buffer to the output queue is $2R$, and the bandwidth from the output queue to the output line is R, a packet can be safely delivered to the output line without being blocked in the middle. However, it should be noted that, because the bandwidth from the input line to the virtual queues is R, cut-throughput switching cannot be used at the input port. In other words, only after all the bits of a new incoming packet have been saved in the virtual queue, the packet can begin to be sent to the crosspoint buffer.

Table 1: Localized Asynchronous Packet Scheduling

```
Input Scheduling:
each input port independently does {
    while true do {
        select a backlogged virtual queue
            whose crosspoint buffer is empty;
        transfer the virtual queue head packet
            to the crosspoint buffer;
        if the channel to the output port is idle {
            transfer the packet to the output queue;
        }
    }
}

Crossbar Scheduling:
each output port independently does {
    while true do {
        select a crosspoint buffered packet;
        transfer the packet to the output queue;
        if the channel to the output line is idle {
            transfer the packet to the output line;
        }
    }
}
```

For easy understanding, the pseudo code description for the input scheduling and crossbar scheduling of LAPS is presented in Table 1. Note that, in input scheduling, the scheduling candidates of an input port are only the virtual queues whose crosspoint buffers are empty. This restriction seems to be unnecessary, because a crosspoint buffer may be able to contain more than one packets with shorter length. However, instead of calculating the remaining free space of the buffer, testing only whether it is empty or occupied greatly simplifies the operation. In this way, only one bit of information needs to be tested for each crosspoint buffer during input scheduling, and the testing of all the crosspoint buffers of the same input port can be conducted in parallel to minimize the time cost. As a consequence, the round-trip-time (RTT) time between input scheduling and crossbar scheduling of LAPS can be neglected. Similarly, in crossbar scheduling, an output port only needs to test whether a crosspoint buffer is occupied or empty due to the cut-through switching capability of the crossbar.

3.2 Performance Analysis

In this subsection, we analyze the performance of LAPS. First, we use the fluid theory in [8] to prove that LAPS is able to achieve 100% throughput for any admissible traffic with speedup of two, regardless of the arbitration rules used by input scheduling and crossbar scheduling.

Before starting the analysis, we define some notations and variables. In_i denotes the i^{th} input port, and Out_j denotes the j^{th}

output port. Q_{ij} denotes the virtual queue of In_i to buffer the packets destined to Out_j, and B_{ij} denotes the crosspoint buffer connecting In_i and Out_j. The following variables are used to represent the status of the switch. Their initial values (at time 0) are conventionally assumed to be zero.
$Q_{ij}(t)$: the number of bits buffered in Q_{ij} at time t
$B_{ij}(t)$: the number of bits buffered in B_{ij} at time t
$A_{ij}(t)$: the number of bits arrived at Q_{ij} up to time t
$D_{ij}(t)$: the number of bits left from Q_{ij} up to time t
$E_{ij}(t)$: the number of bits left from B_{ij} up to time t

$A_{ij}(t)$ satisfies a strong law of large numbers (SLLN), i.e.,

$$\lim_{t\to\infty}\frac{A_{ij}(t)}{t}=\lambda_{ij}$$

where λ_{ij} is called the arrival rate of Q_{ij}.

A traffic is said to be admissible if it has no oversubscription at any input port or output port, i.e.,

$$\forall i,\sum_k \lambda_{ik}\le R$$
$$\forall j,\sum_k \lambda_{kj}\le R$$

Following the definition in [8], we say that a scheduling scheme achieves 100% throughput if the following equation holds for any admissible traffic.

$$\forall i,\forall j,\lim_{t\to\infty}\frac{E_{ij}(t)}{t}=\lambda_{ij}$$

$\lim_{t\to\infty}\frac{\sum_j E_{ij}(t)}{t}$ is the average rate that packets are transmitted to the j^{th} output port. $\lim_{t\to\infty}\frac{\sum_j E_{ij}(t)}{t}=\sum_j\lambda_{ij}$ means that all traffic to the j^{th} output port is delivered to the output queue. Thus, 100% throughput can be achieved, as long as the output scheduling algorithm is work-conserving. Intuitively, traffic arrives at Q_{ij} and leaves from B_{ij} with the same speed, and thus packets will not infinitely accumulate at either Q_{ij} or B_{ij}.

A packet is saved to the virtual queue only after it has been fully received by the input port, and thus the value of $A_{ij}(t)$ changes only at some specific time points $\{t_0,t_1,\dots,t_n,\dots\}$, where $t_0=0$ and $t_n(n>0)$ is the time that the n^{th} packet is saved into the virtual queue. For $t_n<t<t_{n+1}$, we have $A_{ij}(t)=A_{ij}(t_n)$. Similar as in [8], $A_{ij}(t)$ is right continuous and have left limits in $[0,\infty)$.

In reality, $D_{ij}(t)$ and $E_{ij}(t)$ are discrete functions, because a packet is removed from the buffer only after the whole packet has been fully transmitted. In order to make $D_{ij}(t)$ and $E_{ij}(t)$ continuous as in [8], in the following analysis, we assume that a bit is immediately released from the buffer after it has been transmitted. In other words, suppose that s_n $(s_0=0)$ and r_n $(r_0=0)$ are the time that the n^{th} packet is removed from Q_{ij} and B_{ij} respectively. For $s_n\le s<s_{n+1}$,

$$D_{ij}(s)=D_{ij}(s_n)+\frac{s-s_n}{s_{n+1}-s_n}\left(D_{ij}(s_{n+1})-D_{ij}(s_n)\right)$$

and for $r_n\le r<r_{n+1}$,

$$E_{ij}(r)=E_{ij}(r_n)+\frac{r-r_n}{r_{n+1}-r_n}\left(E_{ij}(r_{n+1})-E_{ij}(r_n)\right)$$

When $f(t)$ is differentiable at t, use $\dot{f}(t)$ to denote the derivative. Notice that $\dot{D}_{ij}(t)$ is equal to $2R$ when Q_{ij} is sending a packet to B_{ij}, and is zero otherwise. Similarly, $\dot{E}_{ij}(t)$ is equal to $2R$ when B_{ij} is sending a packet to the output queue, and is zero otherwise.

Regarding the relationship of the defined variables, we have the following fluid equations. It should be noted that, because $A_{ij}(t)$ satisfies a SLLN, we can use $\lambda_{ij}t$ to approximate $A_{ij}(t)$ when t is sufficiently large.

$$Q_{ij}(t)=\lambda_{ij}t-D_{ij}(t)$$
$$B_{ij}(t)=D_{ij}(t)-E_{ij}(t)$$
$$Q_{ij}(t)+B_{ij}(t)=\lambda_{ij}t-E_{ij}(t)$$

In order to prove 100% throughput of LAPS, we need the following lemma from [8].

LEMMA 1. *Let $f:[0,\infty)\to[0,\infty)$ be an absolutely continuous function with $f(0)=0$. Assume that $\dot{f}(t)\le 0$ for almost every t (wrt Lebesgue measure) such that $f(t)>0$ and f is differentiable at t. Then, $f(t)=0$ for almost every t.*

The basic idea to prove 100% throughput of LAPS is to first define the function $f(t)$ in Lemma 1 as follows

$$V(t)=\sum_{ij}Q_{ij}(t)\left(\sum_k Q_{ik}(t)\right)+\sum_{ij}Z_{ij}(t)\left(\sum_k Z_{kj}(t)\right)$$

where $Z_{ij}(t)$ is the sum of $Q_{ij}(t)$ and $B_{ij}(t)$, i.e.,

$$Z_{ij}(t)=Q_{ij}(t)+B_{ij}(t)$$

Then we prove that $V(t)$ has a negative or zero derivative when it is positive, and thus by Lemma 1, $V(t)=0$ for almost every t. Since $Z_{ij}(t)\le V(t)$, we know that $Z_{ij}(t)=0$ for almost every t as well, which means that the length of each virtual queue is always bounded, or in other words all incoming traffic is transmitted to the output queue.

Next, we introduce some supporting lemmas.

LEMMA 2. *If B_{ij} is not empty at time t, $\sum_k Z_{kj}(t)$ has a negative derivative, i.e.,*

$$B_{ij}(t)>0\Rightarrow\sum_k\dot{Z}_{kj}(t)<0$$

PROOF. The intuitive explanation for this lemma is that, if B_{ij} has buffered bits, Out_j must be receiving packets.

Since the crossbar scheduling of LAPS is work-conserving and $B_{ij}>0$, Out_j is either receiving bits from B_{ij} or another crosspoint buffer B_{kj}. Noticing that the bandwidth from the crosspoint buffer to the output queue is $2R$, we can obtain $\sum_k\dot{E}_{kj}(t)=2R$.

According to the fluid equation $Z_{ij}(t)=\lambda_{ij}t-E_{ij}(t)$, we have

$$\begin{aligned}\sum_k\dot{Z}_{kj}(t)&=\sum_k\left(\lambda_{kj}-\dot{E}_{kj}(t)\right)\\&=\sum_k\lambda_{kj}-\sum_k\dot{E}_{kj}(t)\\&\le R-2R\\&<0\end{aligned}$$

It should be noted that, due to the cut-through switching technique, even if B_{kj} has only received part of the packet but not the complete packet, B_{kj} can start to send the packet to the output queue of Out_j.

□

LEMMA 3. *If Q_{ij} is not empty at time t, $\sum_k Q_{ik}(t)+\sum_k Z_{kj}(t)$ has a negative or zero derivative, i.e.,*

$$Q_{ij}(t)>0\Rightarrow\sum_k\dot{Q}_{ik}(t)+\sum_k\dot{Z}_{kj}(t)\le 0$$

PROOF. The intuitive explanation for this lemma is that, when Q_{ij} has buffered bits, either a virtual queue of In_i is sending packets to its crosspoint buffer, or Out_j is receiving packets from one of the crosspoint buffers.

Based on the state of B_{ij}, we consider two possible cases.

Case 1: B_{ij} is empty. Since the input scheduling of LAPS is work-conserving and $Q_{ij}(t) > 0$, either Q_{ij} is sending packets to B_{ij}, or another virtual queue Q_{ik} of the i^{th} input port is sending packets to B_{ik}. For either case, $\sum_k \dot{D}_{ik} = 2R$.

Case 2: B_{ij} is occupied, including the case that B_{ij} has a fully buffered packet, and the case that B_{ij} is receiving a packet from Q_{ij} and simultaneously sending the packet to the output queue of Out_j and $B_{ij}(t) = 0$. Since the crossbar scheduling of LAPS is work-conserving and the crossbar switching fabric uses cut-through switching, either B_{ij} or another crosspoint buffer B_{kj} is sending packets to Out_j. For either case, $\sum_k \dot{E}_{kj} = 2R$.

Note that $\dot{D}_{ij}(t) \geq 0$ and $\dot{E}_{ij}(t) \geq 0$. Thus, for both of the above cases, we can obtain $\sum_k \dot{D}_{ik}(t) + \sum_k \dot{E}_{kj}(t) \geq 2R$.

According to the fluid equations $Q_{ij}(t) = \lambda_{ij}t - D_{ij}(t)$ and $Z_{ij}(t) = \lambda_{ij}t - E_{ij}(t)$, we have

$$\begin{aligned}
&\sum_k \dot{Q}_{ik}(t) + \sum_k \dot{Z}_{kj}(t) \\
=\ &\sum_k \left(\lambda_{ik} - \dot{D}_{ik}(t)\right) + \sum_k \left(\lambda_{kj} - \dot{E}_{kj}(t)\right) \\
=\ &\sum_k \lambda_{ik} + \sum_k \lambda_{kj} - \sum_k \dot{D}_{ik}(t) - \sum_k \dot{E}_{kj}(t) \\
\leq\ &R + R - 2R \\
\leq\ &0
\end{aligned}$$

□

THEOREM 1. *With speedup of two, LAPS achieves 100% throughput for any admissible traffic.*

PROOF. We define

$$\begin{aligned}
V(t) &= \sum_{ij} Q_{ij}(t) \left(\sum_k Q_{ik}(t)\right) + \sum_{ij} Z_{ij}(t) \left(\sum_k Z_{kj}(t)\right) \\
&= \sum_{ijk} \left(Q_{ij}(t)Q_{ik}(t) + Z_{ij}(t)Z_{kj}(t)\right)
\end{aligned}$$

It it clear that $V(0) = 0$. In addition,

$$\begin{aligned}
\dot{V}(t) = &\sum_{ijk} \Big(\dot{Q}_{ij}(t)Q_{ik}(t) + Q_{ij}(t)\dot{Q}_{ik}(t) + \\
&\dot{Z}_{ij}(t)Z_{kj}(t) + Z_{ij}(t)\dot{Z}_{kj}(t)\Big) \\
= &\ 2\sum_{ijk} \left(Q_{ij}(t)\dot{Q}_{ik}(t) + Z_{ij}(t)\dot{Z}_{kj}(t)\right) \\
= &\ 2\sum_{ijk} \left(Q_{ij}(t)\dot{Q}_{ik}(t) + \left(Q_{ij}(t) + B_{ij}(t)\right)\dot{Z}_{kj}(t)\right) \\
= &\ 2\sum_{ijk} Q_{ij}(t) \left(\dot{Q}_{ik}(t) + \dot{Z}_{kj}(t)\right) + 2\sum_{ijk} B_{ij}(t)\dot{Z}_{kj}(t) \\
= &\ 2\sum_{ij} Q_{ij}(t) \left(\sum_k \dot{Q}_{ik}(t) + \sum_k \dot{Z}_{kj}(t)\right) + \\
&\ 2\sum_{ij} B_{ij}(t) \left(\sum_k \dot{Z}_{kj}(t)\right)
\end{aligned}$$

By Lemma 3, we know

$$2\sum_{ij} Q_{ij}(t) \left(\sum_k \dot{Q}_{ik}(t) + \sum_k \dot{Z}_{kj}(t)\right) \leq 0$$

and by Lemma 2,

$$2\sum_{ij} B_{ij}(t) \left(\sum_k \dot{Z}_{kj}(t)\right) \leq 0$$

Thus, we can obtain

$$\dot{V}(t) \leq 0$$

By Lemma 1, we know that $V(t) = 0$ for almost every t. Since $Q_{ij}(t) \leq V(t)$, $Q_{ij}(t) = 0$ for almost every t. Noticing that $B_{ij}(t) \leq L$, we can obtain

$$\begin{aligned}
\lim_{t\to\infty} \frac{E_{ij}(t)}{t} &= \lim_{t\to\infty} \frac{A_{ij}(t) - Q_{ij}(t) - B_{ij}(t)}{t} \\
&= \lim_{t\to\infty} \frac{A_{ij}(t)}{t} - \lim_{t\to\infty} \frac{Q_{ij}(t) + B_{ij}(t)}{t} \\
&= \lambda_{ij} - 0 \\
&= \lambda_{ij}
\end{aligned}$$

The above equation holds for any admissible traffic. Thus, by the definition, LAPS achieves 100% throughput. □

We have studied the throughput of LAPS. Next we discuss the delay and queue length properties.

Before a packet is sent to the output line, it may be buffered at the input queue, the crosspoint buffer, and the output queue, and experience corresponding delay at each location. We define the input queueing delay of a packet to be the interval from the time that the last bit of the packet arrives at the virtual queue to the time that the last bit of the packet leaves the virtual queue. In a similar way, we can define the crossbar queueing delay and output queueing delay. It should be noted that due to the cut-through switching technique, the last bit of a packet may leave the crosspoint buffer as soon as it arrives, which makes the crossbar queueing delay of the packet be zero.

Since the traffic arrival rate at an input port $\sum_i \lambda_{ij}$ is less than or equal to R, and the bandwidth of the crossbar is $2R$, most packets are immediately transmitted through the crossbar after they arrive, but are buffered in output queues. This indicates short input and crossbar queueing delay and long output queueing delay. The observation is consistent with the simulation results obtained in Section 4. Assume that the traffic arrives according to a Poisson process and the packet length follows an exponential distribution with mean M. Then, In_i can be approximately modeled as an M/M/1 system, and accordingly

$$\text{Average input queueing delay} = \frac{M}{2R - \sum_i \lambda_{ij}}$$

Applying Little's Law, we can obtain

$$\text{Average input queue length} = \frac{M\sum_i \lambda_{ij}}{2R - \sum_i \lambda_{ij}}$$

3.3 Hardware Implementation

Practical scheduling algorithms are expected to be efficiently implemented in hardware to make fast decisions for high speed switching. In the following, we discuss the hardware implementation for LAPS.

One of the advantages of LAPS is that each input port or output port makes scheduling decisions solely based on the state information of its local crosspoint buffers. Since no comparison operation is required, the scheduling arbiters can be efficiently implemented using priority encoders [14]. The theoretical time complexity to make an arbitration is $O(\log N)$. In practice, priority encoders perform all the operations in hardware, and technically achieve constant time complexity for a moderate switch size [29]. Depending on the arbitration rules, different types of priority encoders may be used. For example, if arbitration candidates are assigned different priorities at different time, such as the situation in a round-robin arbiter, a programmable priority encoder can be used to implement the arbiter.

Moreover, LAPS allows the scheduling of different input ports and output ports to be conducted in an independent and asynchronous mode. Since different arbiters do not need to exchange scheduling information, LAPS can be implemented in a distributed manner, which makes it highly scalable.

The cost of crosspoint buffers may seem to be a problem for the implementation of buffered crossbar switches. Fortunately, with the recent development of VLSI technology, it has been feasible to integrate small on-chip memory to the crossbar switching fabric [25]. In addition, LAPS requires only L buffer space at each crosspoint buffer, and minimizes the switch hardware cost. For example, if the switch size N is 32, and the maximum packet length L is equal to 12K bits (1.5K bytes), the total size of all crosspoint buffers is 1.5M bytes.

4. SIMULATION RESULTS

We have conducted simulations to verify the 100% throughput of LAPS and to measure its delay and buffer performance.

For the input or crossbar scheduling of LAPS, when there are more than one eligible virtual queues or crosspoint buffers, different scheduling decisions can be made depending on the rules to make the arbitration. In the simulations, we consider five different LAPS implementation versions with different arbitration rules: (1) FP (fixed priority) assigns a fixed priority order to all the virtual queues of the same input port or all the crosspoint buffers to the same output port, and always picks the candidate with the highest priority. In our implementation, higher priorities are assigned to virtual queues to output ports with smaller indexes (e.g., Q_{ij} has higher priority than Q_{ij+1}), or to crosspoint buffers from input ports with smaller indexes (e.g., B_{ij} has higher priority than B_{i+1j}). (2) RD (random) does not favor any particular candidate, but makes arbitration on a random basis. (3) RR (round robin) sets up a round robin pointer for the virtual queues of the same input port or the crosspoint buffers to the same output port, and grants to the first candidate that is equal to or larger than the robin robin pointer (in a modular manner). After making an arbitration, the round robin pointer is updated to the next candidate of the current assigned one (in a modular manner). (4) OPF (oldest packet first) uses the packet arrival time as the arbitration criterion. In input scheduling, the eligible virtual queue whose head packet arrives earliest at the input port is selected. In crossbar scheduling, the eligible crosspoint buffer whose packet arrives earliest at the crosspoint is selected. (5) LQF (longest queue first) uses the queue length as the arbitration criterion. In input or crossbar scheduling, the eligible packet whose virtual queue or crosspoint buffer has the longest queue is selected.

FP, RD and RR rely only on the state information, i.e., for input scheduling, whether a virtual queue has packets and its crosspoint buffer is available, and for crossbar scheduling, whether a crosspoint buffer has a buffered packet. As discussed earlier, since there is no comparison operation involved, these algorithms can be efficiently implemented using priority encoders. In particular, RR is also able to avoid starvation in the scheduling, by giving each candidate the chance to obtain the highest priority. On the other hand, LQF and OPF need to compare either the packet arrival time or the queue length when making arbitrations and require more sophisticated hardware support. Our purpose to include the two algorithms is that LQF and OPF demonstrate advantages in the scheduling for VOQ switches [7], and we want to study whether they are also superior in the scheduling for buffered crossbar switches.

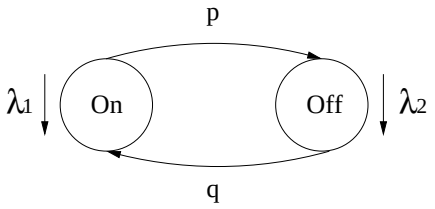


Figure 2: In a Markov modulated Poisson process, the intensity of the Poisson process depends on the state of the Markov chain.

In order to reflect the burst nature of real network traffic, we emulate the incoming traffic by a Markov modulated Poisson process, as illustrated in Fig. 2. The intensity of the Poisson process is defined by the state of a Markov chain. The Markov chain has two states: on and off. In the on state, the intensity of the Poisson process is λ_1, and in the off state the intensity is λ_2. The probability to switch from the on state to the off state is p, and the probability to switch from the off state to the on state is q. In the simulations, we set $p = q = 0.2$ and $\lambda_2 = 0$, and change the value of λ_1 to adjust the load.

For the destination of the packets, we consider both uniform traffic and non-uniform traffic. For uniform traffic, the destination of a new incoming packet is uniformly distributed among all the output ports, i.e., $\lambda_{ij} = lR/N$, where l is the effective load. For non-uniform traffic, we use the same model as that in [18] and [20]. The traffic arrival rate λ_{ij} is defined by i, j and an unbalanced probability w as follows.

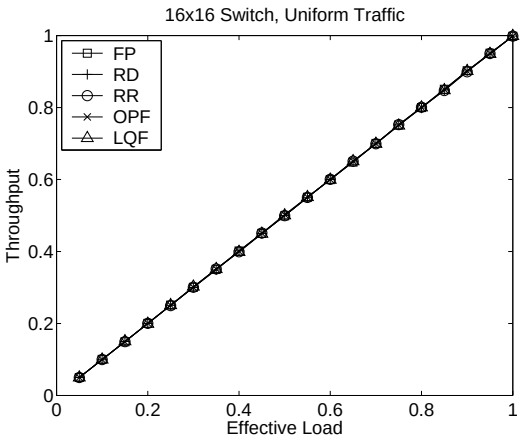
$$\lambda_{ij} = \begin{cases} lR\left(w + \frac{1-w}{N}\right), & \text{if } i = j \\ lR\frac{1-w}{N}, & \text{if } i \neq j \end{cases}$$

The packet length in the simulation is uniformly distributed between [50, 1500] bytes. We consider a 16×16 switch, and each input port or output port has bandwidth of 1G bps. All packets to an output port are buffered in the same queue, and FIFO is used as the output scheduling policy for all the algorithms.

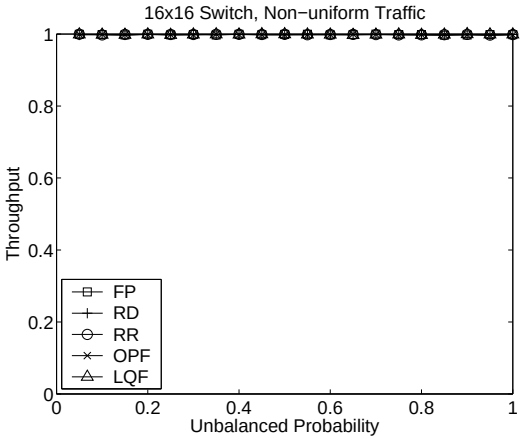
4.1 Throughput

In Section 3, we have theoretically proved that with speedup of two LAPS achieves 100% throughput for any admissible traffic. Now we verify the analytical results by simulation.

Figure 3(a) depicts the relationship between the throughput of different algorithms and the effective load under uniform traffic. As can be seen, all algorithms have similar curves and achieve 100% throughput. Figure 3(b) shows the results under non-uniform traffic. We fix the load of the switch to one, and adjust the unbalanced probability. Again, all the five algorithms achieve 100% throughput. As we have seen, both the simulation data under uniform traffic and non-uniform traffic support the previous analytical results well. It also can be noticed that, in terms of throughput performance, the five algorithms have no significant difference.

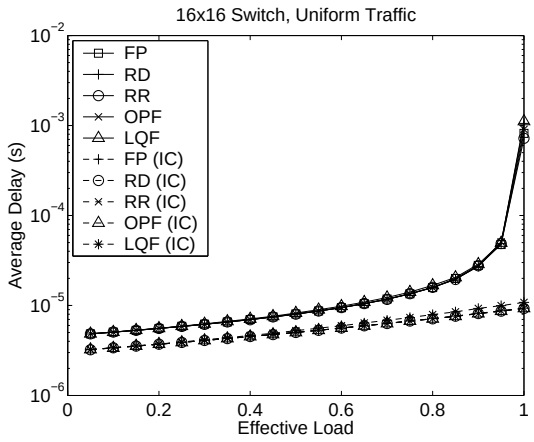


(a) Uniform traffic.

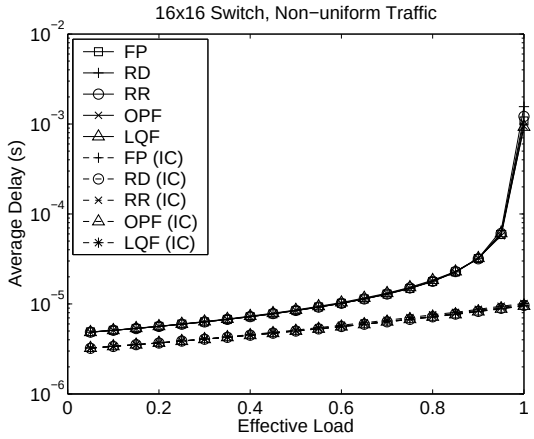


(b) Non-uniform traffic.

Figure 3: Throughput of different algorithms.



(a) Uniform traffic.



(b) Non-uniform traffic.

Figure 4: Average delay of different algorithms.

In the rest simulations, we fix the unbalanced probability of the non-uniform traffic to 0.5.

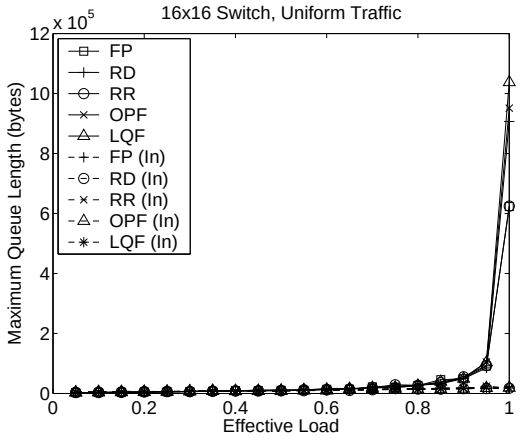
4.2 Average Delay

In this subsection, we study the delay performance of different algorithms. Two types of delay measures are considered. The first one is called transmission delay, which is the interval from the time that the last bit of a packet arrives at its input port to the time that the last bit of the packet is sent to the output line. Transmission delay is the total time that a packet stays in the switch, and is an important performance criterion. The other measure is called input and crossbar (IC) queueing delay, which is the interval from the time that the last bit of a packet arrives at its input port to the time that the last bit of the packet leaves the crosspoint buffer. In this paper, we mainly discuss the input scheduling and crossbar scheduling of buffered crossbar switches, and all the simulation algorithms use the same output scheduling principle. Thus, IC queueing delay is a good measure to compare the different arbitration rules used in different algorithms. On the other hand, the transmission delay of a packet is equal to the sum of its IC queueing delay and its output queueing delay. With the above two measures, it is possible to determine the proportion of time that packets spend at different buffering locations in the switch.

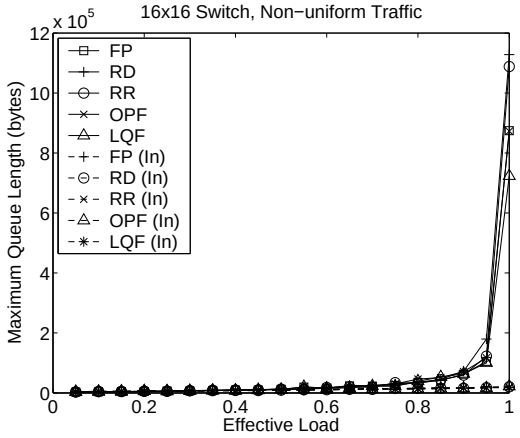
The average transmission delay and IC queueing delay of the five algorithms under uniform traffic and non-uniform traffic are shown in Figure 4(a) and (b) respectively. The solid lines represent the transmission delay, and the dashed lines represent the IC queueing delay. First, we can notice that, for any algorithm, compared with the transmission delay (10^{-3} second), the IC queueing delay (10^{-5} second) is small enough to be neglected. Second, we can also see that the IC queueing delay of different algorithms does not have significant difference. Combining the above two observations, we can make the conclusion that, for buffered crossbar switches with speedup of two, input scheduling and crossbar scheduling do not significantly affect the transmission delay of the packet. Thus, when considering implementation cost, the simplest algorithms, such as RR and FP, are the preferred choices. On the other hand, since the IC queueing delay is very small, most packets are immediately transmitted to their output buffers after they arrive at input ports. Thus, we can expect LAPS to exhibit similar performance as OQ switch based scheduling algorithms. Moreover, in conjunction with LAPS, existing fair scheduling algorithms, such as WFQ, DRR and FMCF, can be used as the output scheduling principles to provide deterministic performance guarantees.

4.3 Maximum Queue Length

In order to achieve 100% throughput for admissible traffic, input ports and output ports must have enough buffer space to avoid



(a) Uniform traffic.



(b) Non-uniform traffic.

Figure 5: Maximum queue length of different algorithms.

packet overflow. We also collect the maximum queue length at both the output side and input side during the simulations to reflect the buffer requirement of the algorithms. The maximum output queue length is defined to be the maximum number of bytes buffered at any output queue during the whole simulation run. The maximum input queue length is defined to be the maximum number of bytes buffered at all the virtual queues of any input port.

Figure 5(a) and (b) show the maximum queue length of the algorithms under uniform traffic and non-uniform traffic respectively. The solid lines represent the maximum output queue length, and the dashed lines represent the maximum input queue length. It can be seen that all the algorithms exhibit similar buffer requirement at both the input side and the output side. On the other hand, the input ports have much shorter maximum queue length than the output ports. This indicates that, with speedup of two, packets can be quickly transferred through the crossbar, and more packets are buffered at output ports than at input ports.

5. CONCLUSIONS

In this paper, we have studied packet scheduling for buffered crossbar switches. Buffered crossbar switches are a special type of CIOQ switches, whose crosspoints are associated with small on-chip buffers. The introduction of crosspoint buffers eliminates output and input contention, and greatly simplifies the scheduling process. Furthermore, the scheduling of different input ports or output ports are conducted in an independent and asynchronous mode, and variable length packets can be directly scheduled and transmitted without segmentation or reassembly. Compared with cell scheduling, packet scheduling has some unique advantages: higher throughput, shorter packet delay and cheaper hardware cost. We have presented a packet scheduling scheme called Localized Asynchronous Packet Scheduling (LAPS) for buffered crossbar switches. With LAPS, each crosspoint needs as little as L buffer space, which minimizes the hardware cost for switches. Another advantage of LAPS is that the scheduling of an input port or output port relies only on the state information of its local crosspoint buffers. The localization property makes LAPS suitable for a distributed implementation and thus highly scalable. Since there is no comparison needed, priority encoders can be used to quickly make scheduling arbitrations in hardware. We also theoretically proved that LAPS with speedup of two achieves 100% throughput for any admissible traffic, and conducted simulations to verify the analytical results and evaluate the performance of LAPS.

6. REFERENCES

[1] 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.

[2] 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.

[3] D. Pan and Y. Yang, "Credit based fair scheduling for packet switched networks," *Proc. of IEEE INFOCOM 2005*, pp. 843-854, Miami, FL, March 2005.

[4] 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.

[5] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 188-201, 1999.

[6] H. J. Chao, "Saturn: A terabit packet switch using dual round-robin," *IEEE Communications Magazine*, vol. 8, no. 12, pp. 78-84, Dec. 2000

[7] 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.

[8] J. G. 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.

[9] D. Pan and Y. Yang, "Pipelined two step iterative matching algorithms for CIOQ crossbar switches," *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, Oct. 2005, Princeton, NJ.

[10] M. J. Karol, M. J. Hluchyj and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. 35, pp. 1347-1356, 1987.

[11] N. McKeown, "A fast switched backplane for a gigabit switched router," *Business Communications Review*, vol. 27, no. 12, 1997.

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

[13] J. Xu and R. Lipton, "On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms", *ACM SIGCOMM'2002*, Pittsburgh, PA, Aug. 2002.

[14] M. Morris Mano, *Digital Design*, 3rd edition, Prentice Hall, Aug. 2001.

[15] 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.

[16] S.-T. 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.

[17] 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.

[18] 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.

[19] Rojas-Cessa, E. Oki and H. J. Chao, "CIXOB-k: Combined input-crosspoint-output buffered packet switch," *IEEE Globecom'01*, San Antonio, Texas, Nov. 2001.

[20] 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

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

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] J. Hopcroft and R. Karp, "An $N^{5/2}$ algorithm for maximum matching in bipartite graphs," *SIAM Journal of Computing*, vol. 2, no. 4, pp. 225-231, Dec. 1973.

[28] R. Tarjan, "Data structures and network algorithms," *CBMS-NSF Regional Conference Series in Applied Mathematics*, Dec. 1983.

[29] S. Ramabhadran, J. Pasquale, "Stratified round robin: a low complexity packet scheduler with bandwidth fairness and bounded delay," *ACM SIGCOMM '03*, pp. 239-250, Karlsruhe, Germany, Aug. 2003.