Packet-mode Asynchronous Scheduling Algorithm for Partially Buffered Crossbar Switches

Masoumeh Karimi, Zhuo Sun, Deng Pan, and Zesheng Chen
Florida International University, Miami, FL, USA
Email: {mkari001, zsun003, pand, zchen}@fiu.edu

***Abstract*—Traditional crossbar switches use centralized scheduling algorithms with high time complexity. In contrast, buffered crossbar switches are capable of distributed scheduling due to crosspoint buffers, which decouple the dependency between inputs and outputs. However, crosspoint buffers are expensive on-chip memories. To reduce the hardware cost of buffered crossbar switches and make them scalable, we consider partially-buffered crossbar switches, whose crosspoint buffers can be of an arbitrarily small size and store only part of a packet instead of the entire packet. In this paper, we propose the Packet-mode Asynchronous Scheduling Algorithm (PASA) for partially buffered crossbar switches. PASA combines the features of both distributed and centralized scheduling algorithms. It works in an asynchronous mode and can directly handle variable length packets without Segmentation And Reassembly (SAR). We theoretically prove that, with a speedup of two, PASA achieves 100% throughput for any admissible traffic. We also show that outputs in PASA have a large probability to avoid the more time-consuming centralized scheduling process, and thus make fast scheduling decisions. Finally, we present simulation data to verify the analytical results and evaluate the performance of PASA.**

I. Introduction

Crossbar switches have received significant attention over the past two decades. They offer non-blocking capability and large bandwidth utilization in comparison with bus based switches [3]. According to whether the crossbar switching fabric has integrated buffers or not, crossbar switches can be generally divided into two types: non-buffered and buffered.

In non-buffered crossbar switches, buffers may exist at inputs or outputs but not on the crossbar. They usually work with centralized cell-mode scheduling algorithms, such as Parallel Iterative Matching [9], iSLIP [19], and maximum size or weight bipartite matching [15], which have high time complexity. To be specific, non-buffered crossbar switches operate in a synchronized time slot mode and carry fixed length cells. In each time slot, the centralized scheduling algorithm collects the statuses of all inputs and outputs, and generates matched input-output pairs. All the matched pairs will then simultaneously transmit a cell. When variable length packets arrive, they are segmented into fixed length cells at inputs. After the cells arrive at outputs, they are reassembled to the original packets before being sent to output lines. This process is called Segmentation And Reassembly (SAR) [11].

To simplify packet reassembly in the SAR process, packet-mode scheduling is proposed in [18]. In this scheme, all cells belonging to the same packet are consecutively transmitted from the input to the output without interruption. Therefore, a connection between an input-output pair has to be maintained until all cells of a packet finish transmission. Packet-mode scheduling has unique advantages over cell-mode scheduling [2], [5], such as high throughput, low delay, and reduced hardware cost. However, synchronized packet-mode scheduling of non-buffered crossbar switches has some drawbacks. First, it may cause temporary flow starvation [16], [21]. When one input-output pair finishes transmitting the last cell of a packet, all the other pairs may still be in the middle of transmitting their packets and not available. As a result, the pair has to transmit one more packet from the same flow. This process may repeat for an arbitrary long time, and results in starvation for other flows. Furthermore, it does not eliminate port contention [12]. Output contention happens when different inputs simultaneously have cells destined to the same output, and input contention happens when several outputs concurrently demand cells from the same input.

Regardless of the applied scheduling scheme, non-buffered crossbar switches still suffer from centralized scheduling with high time complexity. Advances in modern VLSI technology have made it feasible to integrate a miniaturized on-chip memory to the crossbar [1], so that each crosspoint of the crossbar can have an exclusive buffer. Crosspoint buffers provide temporary buffering during packet transmission, and thus decouple the dependency between inputs and outputs. Crossbar switches with such crosspoint buffers are called buffered crossbar switches. They can work with distributed packet-mode scheduling algorithms [2], [8]. To be specific, they operate in an asynchronous mode, and each input or output independently selects a packet to send to or retrieve from one of its crosspoint buffers. As a result, they can directly handle variable length packets without SAR. Buffered crossbar switches usually have a small fixed speedup of two [4], which means that the bandwidth of the crossbar is twice as that of the input and output.

However, on-chip memories are expensive resources. Existing buffered crossbar switches require storing at least an entire packet at each crosspoint. Furthermore, the total crosspoint buffer size grows by the square of the switch size. To reduce the hardware cost of buffered crossbar switches and make them scalable, we consider partially buffered crossbar switches, whose crosspoint buffer size can be an arbitrarily small length, even smaller than the maximum packet length. They make a cost performance tradeoff, but improve significantly over non-buffered crossbar switches with the small crosspoint buffers. Note that partially buffered crossbar switches are defined in

[7] to be a switch maintaining a small number of buffers per output, which cannot directly deal with variable length packets.

In this paper, we propose the Packet-mode Asynchronous Scheduling Algorithm (PASA) for partially buffered crossbar switches. PASA combines the features of distributed and centralized scheduling algorithms. Inputs and outputs in PASA can conduct distributed scheduling for most of the time, but occasionally need to participate in a centralized scheduling. PASA works in an asynchronous mode, and directly handles variable length packets without SAR. We theoretically prove that, with a speedup of two, PASA achieves 100% throughput for any admissible traffic. We also show that outputs in PASA have a large probability to avoid the more time-consuming centralized scheduling process and thus make fast scheduling decisions. Finally, we conduct simulations to verify the analytical results and evaluate the performance of PASA.

II. Packet-mode Asynchronous Scheduling Algorithm

In this section, we describe the Packet-mode Asynchronous Scheduling Algorithm (PASA) for partially buffered crossbar switches.

The switch structure that we consider is depicted in Figure 1. N inputs and N outputs are connected by a buffered crossbar, which has a speedup of two. Let R denote the bandwidth of the input and output, and the crossbar has a bandwidth of $2R$. We use Virtual Output Queue (VOQ) buffering to avoid the Head Of Line (HOL) blocking, which limits the maximum throughput of the switch [15]. VOQ buffering maintains a virtual queue for each output at every input. Each crosspoint has a small exclusive buffer. Depending on the granularity level of performance guarantees, an output may have a single or multiple logical queues. In order to reduce the packet latency and increase the switch throughput, we consider cut-through switching for the crossbar. To be specific, a packet does not need to be fully buffered at the crosspoint, during the transmission from the input to the output. When a packet is being sent from its virtual queue to the crosspoint buffer, if the transmission channel to its output is idle, bits of the same packet can be simultaneously sent further from the crosspoint buffer to the output.

For such a switch, there are three types of scheduling involved, which we call input scheduling, output scheduling, and departure scheduling. In input scheduling, an input selects a packet from one of its virtual queues, and sends it to the crosspoint buffer. In output scheduling, an output selects a packet from one of its crosspoint buffers, and retrieves it to the output buffer. In departure scheduling, an output selects a packet from its output buffer, and delivers it to the output line. Departure scheduling is a well studied area, and there are many algorithms available in the literature, such as First-In-First-Out [3], Weighted Fair Queueing [17], and Deficit Round Robin [10]. In this paper, we focus on input scheduling and output scheduling. In the scheduling processes, when an input or output needs to make a selection among multiple candidates, different arbitration rules may be used, such as oldest packet first or round robin.

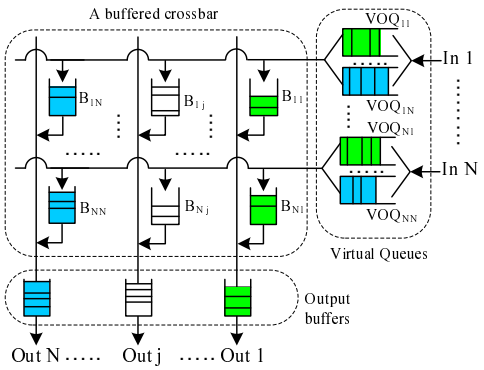


Fig. 1. Structure of partially buffered crossbar switches.

Next, we describe the output scheduling of PASA. The first step of output scheduling is to identify eligible packets, and there are two types of them. For the first type, if a packet is entirely buffered in the crosspoint buffer, which is called a fully buffered packet, it is eligible. It is also possible that only part of a packet is in the crosspoint buffer, because either the packet length is larger than the crosspoint buffer size or the remaining part has not been delivered yet. We call such a packet a partially buffered packet, for which we need to look at the input of the crosspoint buffer. If the input is not currently sending another packet to a different output by cut-through switching, the packet is eligible. Otherwise, if the input is sending a different packet to the crosspoint buffer and then further to the output by cut-through switching, the original packet is not eligible. The reason is that an input cannot sustain simultaneous transmission to two outputs. The restriction can be implemented by using a lock similar to that in [5]. If an input is sending a packet to a crosspoint buffer, which is sending the same packet to the output using cut-through switching, the input is locked. An input with a lock indicates that its crosspoint buffers are no longer available to other outputs. After the input has sent the last bit of the current packet to the crosspoint buffer, the lock is released. Note that the lock may be released before an entire packet is delivered to the output, because part of the packet was initially in the crosspoint buffer. Because of the lock, conflicts may exist when multiple outputs need to make scheduling decisions at the same time. If an input has multiple crosspoint buffers with partially buffered packets, the first output retrieving a packet from one of the crosspoint buffers will lock out other outputs.

After an output identifies the eligible packets, it tries to make the scheduling decision by selecting one of the fully buffered packets. This selecting process for fully buffered packets is independent for different output ports, similar to the distributed scheduling process of existing buffered crossbar switches. Otherwise, if the output has no fully buffered packets but only partially buffered packets, it selects a partially buffered packet, and sends a grant to the corresponding input. An input may receive multiple grants, and it selects one grant to accept. If the grant of the output is accepted, the output can immediately start retrieving the packet. Otherwise, if its grant is not accepted, it needs to repeat the iteration, i.e., sending a

TABLE 1
PSEUDO CODE DESCRIPTION OF PASA

```
Output Scheduling:
While (true) {
   Identifies eligible packets;
    If (eligible fully buffered packets exist) {
        Selects a fully buffered packet and retrieves it to output buffer;
     }
    Else {
       While (output is not matched and eligible partially buffered
          packets exist) {
          Selects a partially buffered packet and grants its input;
          If (the grant is accepted) {
            Output is matched and retrieves the packet to the output buffer;
          }
       }
   }
}

Input Scheduling:
While (true) {
   If (receives grants) {
      Selects a grant to accept;
      Adds a lock;
      Sends the remaining part of a packet to the crosspoint buffer;
      Releases the lock;
   }
   Else {
      Identifies the crosspoint buffers with free space;
      Selects such a crosspoint buffer, giving higher priorities to those
           with idle outputs;
      Sends a packet to the crosspoint buffer,
           until the buffer is full or the transmission is interrupted;
   }
}
```

second grant to the input of a different partially buffered packet and waiting for the acceptation. As can be seen, the iterative matching process for partially buffered packets is similar to the centralized scheduling process of non-buffered crossbar switches. Although an output may need to participate in the iterative matching process, the average scheduling time can be kept small due to the reasons discussed in detail in Section III-B.

Now we describe the input scheduling of PASA. If an input has no lock, it can select a crosspoint buffer with free space and start sending a packet to it. Again, because the crosspoint buffer size is small, a packet may only be able to have its first part sent to the crosspoint buffer, and the second part will remain in the virtual queue. To maximize the throughput, the input scheduling gives higher priorities to the crosspoint buffers with idle outputs, i.e., the outputs currently not retrieving packets. On the other hand, if the crosspoint buffer of an idle output starts receiving a packet, it will notify the output to start retrieving the packet by cut-through switching. It should be noted that input scheduling may be interrupted by output scheduling. To be specific, if an input is currently sending a packet to a crosspoint buffer but not further to an output, and now a second output is matched to retrieve a partially buffered packet from a crosspoint buffer of this input by cut-through switching, the input will stop current transmission and switch to send the remaining part of the partially buffered packet to the matched output. For easy reading, the pseudo code description of the output scheduling and input scheduling of PASA is given in Table 1.

III. PERFORMANCE ANALYSIS

In this section, we analyze the performance of PASA. We use the fluid model in [6] to prove that with a speedup of two, PASA achieves 100% throughput for any admissible traffic, regardless of the applied arbitration rule. We also show that outputs in PASA have a large probability to avoid the more time-consuming iterative matching process.

A. Throughput

We first define some notations. In_i denotes the i^{th} input, and Out_j indicates the j^{th} output. Q_{ij} denotes the virtual queue of In_i to buffer packets destined to Out_j, and B_{ij} represents the crosspoint buffer connecting In_i to Out_j. The bandwidth of an input or output is equal to R, and therefore the bandwidth of the crossbar is $2R$ with a speedup of two. The following variables are used to represent the status of the crossbar switch. Their initial values are assumed to be zero at time $t = 0$.

$B_{ij}(t)$: the number of bits buffered in B_{ij} at time t
$Q_{ij}(t)$: the number of bits buffered in Q_{ij} at time t
$A_{ij}(t)$: the number of bits arriving at Q_{ij} up to time t
$D_{ij}(t)$: the number of bits departing from Q_{ij} up to time t
$E_{ij}(t)$: the number of bits departing from B_{ij} up to time t

Assume that the number of arriving bits $A_{ij}(t)$ satisfies the strong Law of Large Numbers (LLN), i.e.

$$\lim_{n\to\infty}[A_{ij}(t)/t] = \lambda_{ij} \tag{1}$$

where λ_{ij} is the arrival rate of Q_{ij}. A traffic is said to be admissible when equation (1) holds and the non-oversubscription principle is satisfied at all inputs and outputs, i.e.

$$\forall i, \sum_j \lambda_{ij} \le R, \quad and \quad \forall j, \sum_i \lambda_{ij} \le R \tag{2}$$

A crossbar switch operating under a scheduling algorithm is said to be rate stable [14] if for any admissible traffic, the following equation holds.

$$\forall i, \forall j, \lim_{t\to\infty}[E_{ij}(t)/t] = \lambda_{ij} \tag{3}$$

A scheduling algorithm delivers 100% throughput, if the crossbar switch holds equations (1) to (3).

We use the fluid model in [6] to describe the dynamic characteristic of a crossbar switch. We have the following equations for our partially buffered crossbar switches

$$Q_{ij}(t) = A_{ij}(t) - D_{ij}(t) = \lambda_{ij}t - D_{ij}(t) \tag{4}$$

$$B_{ij}(t) = D_{ij}(t) - E_{ij}(t) \tag{5}$$

The sum of the left-hand sides of equations (4) and (5) gives us the total number of accumulated bits of the flow from In_i to Out_j up to time t, which we define to be $Z_{ij}(t)$, i.e.

$$Z_{ij}(t) = Q_{ij}(t) + B_{ij}(t) = \lambda_{ij}t - E_{ij}(t) \tag{6}$$

When $f(t)$ is differentiable at time t, use $\dot{f}(t)$ to denote the derivative. Now, we introduce a lemma from [14].

Lemma 1: Let f be a non-negative, absolutely continuous function defined on $[0, \infty)$ with $f(0) = 0$. Assume that for almost every t such that $f(t) > 0$, $\dot{f}(t) \le 0$. Then $f(t) = 0$ for almost every $t \ge 0$.

In order to prove 100% throughput of PASA, we first define a variable $C_{ij}(t)$ as follows

$$C_{ij}(t) = \sum_{j'} Z_{ij'}(t) + \sum_{i'} Z_{i'j}(t) \tag{7}$$

$C_{ij}(t)$ denotes the sum of the bits buffered at the virtual queues and crosspoint buffers from In_i and to Out_j. Use $C(t)$ to represent the matrix formed by $C_{ij}(t)$. Now, we define $f(t)$ as a non-negative function as follows

$$f(t) =< Z(t), C(t) >= \sum_{ij} Z_{ij}(t)C_{ij}(t) \quad (8)$$

Then, we want to show that $f(t)$ has a negative or zero derivative, and thus by Lemma 1, $f(t) = 0$ for almost every t. Since $Z_{ij}(t) \leq C_{ij}(t) \leq f(t)$, we know that $Z_{ij}(t) = 0$ for almost every t as well, which means that $Q_{ij}(t)$ and $B_{ij}(t)$ are always bounded. In other words, all packets waiting at the virtual queues and crosspoint buffers are getting evacuated to outputs. Next, we present a supporting lemma.

Lemma 2: If either Q_{ij} or B_{ij} is not empty at time t, $C_{ij}(t)$ has a negative or zero derivative, i.e.

$$Z_{ij}(t) = Q_{ij}(t) + B_{ij}(t) > 0 \quad \Rightarrow \quad \dot{C}_{ij}(t) \leq 0$$

Proof: We look at two possible cases regarding whether B_{ij} is empty at time t.

Case 1. $B_{ij}(t) > 0$. This indicates that B_{ij} has a fully or partially buffered packet. Three situations may occur in this case as follows. Note that the crossbar has a bandwidth of $2R$.

1.1 Out_j is retrieving a packet from B_{ij}. Thus, $\sum_{i'} \dot{E}_{i'j}(t) = 2R$

1.2 Out_j is retrieving a packet from another crosspoint buffer B_{kj} where $k \neq i$. We have $\sum_{i'} \dot{E}_{i'j}(t) = 2R$

1.3 Out_j cannot retrieve a packet from B_{ij}, because another output Out_k is retrieving a packet from B_{ik} where $k \neq j$ and In_i is locked. We can obtain $\sum_{j'} \dot{E}_{ij'}(t) = 2R$

Since $\dot{E}_{ij}(t) \geq 0$, we obtain the following equation for cases 1.1 to 1.3

$$\sum_{j'} \dot{E}_{ij'}(t) + \sum_{i'} \dot{E}_{i'j}(t) \geq 2R \quad (9)$$

Case 2. $Q_{ij}(t) > 0$ and $B_{ij}(t) = 0$. This indicates that B_{ij} is empty but Q_{ij} has buffered bits destined to Out_j. Two situations may occur in this case as follows.

2.1 Out_j is idle. However, by $B_{ij}(t) = 0$ we know that In_i is not sending a packet to B_{ij} in input scheduling. This means that, there was another idle output Out_k where $k \neq j$, and In_i gave a higher priority to B_{ik} than B_{ij} in input scheduling. Because Out_k was idle, after B_{ik} started receiving the packet from In_i, it has immediately sent the same packet to Out_k by cut-through switching. As a result, In_i is locked, and thus even though Q_{ij} has buffered bits, they cannot be sent to B_{ij}. In this situation, we have $\sum_{j'} \dot{E}_{ij'}(t) = 2R$

2.2 Out_j is not idle but retrieving a packet from B_{kj} where $k \neq i$. Thus $\sum_{i'} \dot{E}_{i'j}(t) = 2R$

Since $\dot{E}_{ij}(t) \geq 0$, we obtain for both cases 2.1 and 2.2

$$\sum_{j'} \dot{E}_{ij'}(t) + \sum_{i'} \dot{E}_{i'j}(t) \geq 2R \quad (10)$$

Now, we return to find the derivative of $C_{ij}(t)$. By (6), (7), (9), and (10) we obtain

$$\begin{aligned} \dot{C}_{ij}(t) &= \sum_{j'} [\lambda_{ij'} - \dot{E}_{ij'}(t)] + \sum_{i'} [\lambda_{i'j} - \dot{E}_{i'j}(t)] \\ &\leq (R + R) - 2R \leq 0 \end{aligned}$$

■

Theorem 1: For a partially buffered crossbar switch with a speedup of two, the Packet-mode Asynchronous Scheduling Algorithm (PASA) can provide 100% throughput for any admissible traffic.

Proof: We have defined $f(t)$ in (8), thus

$$f(t) = \sum_{ijk} [Z_{ij}(t)Z_{ik}(t) + Z_{ij}(t)Z_{kj}(t)]$$

It is obvious that $f(0) = 0$. We now calculate the derivative of $f(t)$.

$$\dot{f}(t) = \sum_{ijk} \dot{Z}_{ij}(t)Z_{ik}(t) + \sum_{ijk} Z_{ij}(t)\dot{Z}_{ik}(t) + \sum_{ijk} \dot{Z}_{ij}(t)Z_{kj}(t) + \sum_{ijk} Z_{ij}(t)\dot{Z}_{kj}(t) = 2\sum_{ij} Z_{ij}(t)\dot{C}_{ij}(t)$$

We know by Lemma 2 that when $Z_{ij}(t) > 0$, then $\dot{C}_{ij}(t) \leq 0$. Therefore, $2\sum_{ij} Z_{ij}(t)\dot{C}_{ij}(t) \leq 0$, which indicates $\dot{f}(t) \leq 0$.

By applying Lemma 1, we find that $f(t) = 0$ for almost every t. Because $Q_{ij}(t) \leq f(t)$ and $B_{ij}(t) \leq f(t)$, we know that $Q_{ij}(t)$ and $B_{ij}(t)$ are bounded as well. Equation (3) can now be validated as follows

$$\lim_{t\to\infty} [E_{ij}(t)/t] = \lim_{t\to\infty} [A_{ij}(t) - Q_{ij}(t) - B_{ij}(t)]/t =$$
$$\lim_{t\to\infty} [A_{ij}(t)/t] - \lim_{t\to\infty} [Q_{ij}(t) + B_{ij}(t)/t] = \lambda_{ij} - 0 = \lambda_{ij}$$

Since the above equation holds for any admissible traffic, PASA achieves 100% throughput by the definition. ■

B. Conflict-free Probability of Outputs

As seen in Section II, an output may need to participate in the iterative matching process, which increases the scheduling time. In this subsection, we show that an output has a large probability to avoid the iterative matching process, which is also supported by the simulation data in Section IV-B.

Define the conflict-free probability of an output to be the probability that when the output makes a scheduling decision, no other outputs need to make scheduling decisions at the same time and hence no conflicts from other outputs.

For easy analysis, we assume that the packet length l follows the geometric distribution with parameter p, i.e.

$$Pr(l = k) = p.(1 - p)^{k-1}, \ k = 1, 2, 3, ...$$

Theorem 2: When the packet length follows the geometric distribution with parameter p, the conflict-free probability of an output in PASA is larger than or equal to $(1 - p/2)^{N-1}$.

Proof: To simplify the proof, we assume that all outputs work in a synchronized time slot mode, and one slot is the time to transmit one byte. To be specific, an output can only start retrieving a packet at the beginning of a time slot. Because the packet length is always a multiple of bytes, the output will finish retrieving the packet at the end of a time slot. As can be seen, the time slot mode leads to a lower conflict-free probability, because the operations of all the outputs are synchronized. In PASA, different outputs operate asynchronously, and may start and finish packet retrieval at any time. Thus, the conflict-free probability in PASA should be higher than the analytical result obtained from the synchronized time slot mode.

First, we assume that a specific output Out_j finishes the current packet transmission at time slot s, and needs to make a scheduling decision to retrieve the next packet.

Next, we look at the probability that a different output Out_k needs to make a scheduling decision at the end of time slot s.

Use ϕ_k to denote the current load to Out_k, and note that the crossbar bandwidth is $2R$. Then the probability that Out_k is transmitting a packet at time slot s is $\phi_k/2R$, i.e.

$$Pr(Out_k \text{ is busy at } s) = \phi_k/2R$$

Furthermore, because the packet length follows the geometric distribution, which is memoryless [13], the probability that Out_k is retrieving the last byte of the current packet at time slot s is p. In other words, Out_k has the probability of p to finish transmitting the current packet at time slot s, i.e.

$$Pr(Out_k \text{ finishes transmission at } s \mid Out_k \text{ is busy at } s) = p$$

As a result, the probability that Out_k needs to make a scheduling decision at the end of time slot s is the multiplication of the above two probabilities $p\phi_k/2R$. Accordingly, the probability that Out_k does not need a scheduling decision at the end of time slot s is $1 - p\phi_k/2R$.

We can safely assume that the packet arrival for each output is independent. Thus, the probability that no output except Out_j will make a scheduling decision at the end of time slot s is $\prod_{k\neq j}^{N}(1 - p\phi_k/2R)$. Note that $\phi_k \leq R$, we have

$$Pr(Out_j \text{ is conflict-free at } s) \geq (1 - p/2)^{N-1}$$

Use a 16×16 switch for IP networks as an example. Since the length of IP packets typically varies from 40 to 1,500 bytes [5], we assume that the average packet length is 770 bytes, which is equal to $1/p$. Using the result in Theorem 2, we know that the conflict-free probability is higher than 99%.

It should be noted that in practice PASA has an even higher conflict-free probability for the following three reasons. First, outputs in PASA operate asynchronously, and they have fewer conflicts than in the synchronized time slot mode assumed in Theorem 2. Second, even though multiple outputs may need to simultaneously make scheduling decisions, some outputs may have fully buffered packets, in which case they do not participate in the iterative matching process. Third, even though multiple outputs have only partially buffered packets, the partially buffered packets may be in the crosspoint buffers of different inputs, and thus there is no conflict. ■

IV. Simulation Results

In this section, we conduct simulations to verify the analytical results in Section III, and to evaluate the performance of PASA. We consider a 16×16 switch. Each input and output has a bandwidth of $R = 1$ Gbps, and the crossbar has a speedup of two. Because the geometric packet length distribution used in the analysis of Section III-B may have an arbitrary long packet length, it is not practical. We set the packet length to be distributed between 40 and 1,500 bytes [5]. Since in a real network, packet arrival is typically bursty and packets are highly correlated, we use a two-state Markov Modulated Poisson Process (MMPP) to model the incoming traffic [2]. 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., $\lambda_{ij} = \eta R/N$, where η is the effective load. For non-uniform traffic, we use the same model as that in [20]. The traffic arrival rate λ_{ij} is defined by i, j and an unbalanced probability w as follows.

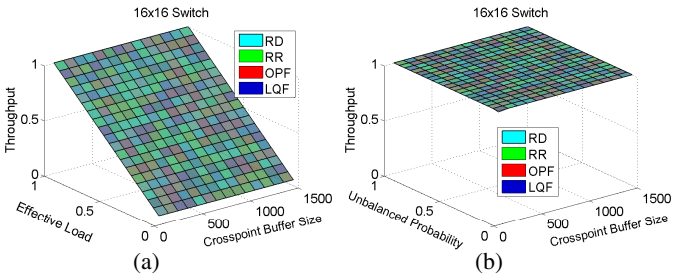


Fig. 2. Throughput of different arbitration rules. (a) Uniform Traffic. (b) Non-uniform Traffic.

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

In our simulations, we consider the following arbitration rules to see how they affect the performance: (1) Random (RD) makes the arbitration on a random basis; (2) Round Robin (RR) alternatively chooses eligible candidates in a round robin manner to avoid starvation; (3) Oldest Packet First (OPF) uses the packet arrival time as the arbitration criterion, i.e., the packet arriving earlier has a higher priority; (4) Longest Queue First (LQF) uses the queue length as the arbitration criterion, i.e., the packet with the longest queue in the virtual queue or crosspoint buffer is chosen. We use a simple First-In-First-Out policy as the departure scheduling algorithm, which is a work conserving algorithm [8].

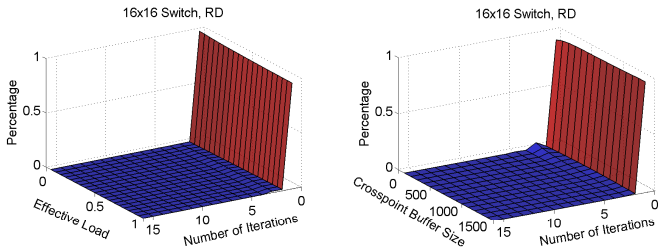
A. Throughput

Theorem 1 in Section III-A shows that for a partially buffered crossbar switch with a speedup of two, PASA achieves 100% throughput for any admissible traffic. We now look at the simulation throughput data. Figure 2(a) demonstrates the switch throughput under different effective loads and crosspoint buffer sizes. We can see that the throughput results of different arbitration rules are almost identical, and grow consistently with the effective load. Finally, all reach 100% throughput when the effective load becomes 1, no matter what the crosspoint buffer size is. Figure 2(b) shows the throughput under non-uniform traffic. We fix the effective load to 1, and adjust the unbalanced probability and the crosspoint buffer size. As can be seen, all the arbitration rules achieve 100% throughput. We can make the conclusion that, the four arbitration rules have no significant difference regarding the throughput performance.

B. Conflict-free Probability

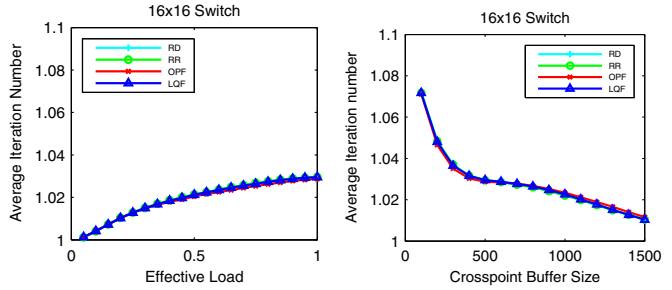
Theorem 2 in Section III-B proves that an output in PASA has a large conflict-free probability. In this subsection, we look at the number of iterations for an output to obtain its scheduling decision. It may happen that the output selects one of the fully buffered packets to retrieve, without participating in the iterative matching process. In such a case, the output is counted as using one iteration to obtain its scheduling decision, because it spends approximately the same scheduling time as in the case that its first grant is accepted in an iterative matching process.

Figure 3(a) depicts the percentage of the number of iterations for Out_0 to obtain its scheduling decisions in RD, with a fixed buffer size under uniform traffic. The Y-axis denotes



(a) (b)

Fig. 3. Percentage of different iteration numbers of Out_0 in RD. (a) With different effective loads. (b) With different crosspoint buffer sizes.



(a) (b)

Fig. 4. Average iteration number of different arbitration rules. (a) With different effective loads. (b) With different crosspoint buffer sizes.

the number of iterations that Out_0 may use. In the best case, Out_0 needs just one iteration, and in the worst case, it needs 16 iterations. As expected, the percentage for Out_0 to use one iteration to obtain its scheduling decisions is close to one, ranging from 97.3% to 99.9% depending on different effective loads. Note that some percentage values are less than the analytical result in Section III-B, because different packet length models are used. When the iteration number becomes larger than one, the percentage quickly drops to less than 1% and then further to zero. Figure 3(b) demonstrates the percentage of the number of iterations for Out_0 to obtain its scheduling decisions in RD, by varying the buffer size, when the effective load is fixed to 1. Similarly, the percentage that Out_0 needs only one iteration is close to one, and when the iteration number becomes greater than one, the percentage drops quickly. As can be seen, the number of iterations slightly increases when the crosspoint buffer size becomes smaller than 300 bytes, which demonstrates the cost performance tradeoff.

Figure 4(a) shows the average iteration number of all the outputs under uniform traffic, when the effective load changes. As can be seen, the average iteration number is very close to but slightly greater than one, and it increases slowly with the effective load. Thus, for most of the time, an output can start retrieving a packet after just one iteration. Among the four arbitration rules, OPF has a comparatively smaller average iteration number. Figure 4(b) shows the average iteration number of all the outputs when the crosspoint buffer size changes. We can obtain the similar conclusion that the average iteration number is very close to one. When the crosspoint buffer size increases, the average iteration number drops accordingly, which reflects the cost performance tradeoff.

V. Conclusion

In this paper, we have studied the partially buffered crossbar switches, which lower the hardware cost of existing buffered crossbar switches by reducing the crosspoint buffers to an arbitrary small size. They make a cost performance tradeoff, but improve significantly over traditional non-buffered crossbar switches with the small crosspoint buffers. Based on partially buffered crossbar switches, we propose the Packet-mode Asynchronous Scheduling Algorithm (PASA), which combines the features of both distributed and centralized scheduling algorithms. PASA enables different inputs and outputs to work asynchronously, and can directly deal with variable length packets without Segmentation And Reassembly (SAR). We prove that with a speedup of two, PASA achieves 100% throughput for any admissible traffic. We also show that outputs in PASA have a large probability close to one to avoid the more time-consuming centralized scheduling process and thus make fast scheduling decisions. Finally, we present simulation data to show that they are consistent with the analytical results.

References

[1] I. Papaefstathiou, G. Kornaros, and N. ChrysosUsing, "Buffered crossbars for chip interconnection," *17th Great Lakes Symposium on VLSI*, pp. 90-95, Stresa-Lago Maggiore, Italy, Mar. 2007.

[2] D. Pan and Y. Yang, "Localized Independent packet scheduling for buffered crossbar switches," *IEEE Trans. on Comp.*, vol. 58, no. 2, 2009.

[3] J. Kurose and K. Ross, "Computer networking: a top-down approach," *Addison Wesley*, 4th edition, 2007.

[4] S. Chuang, S. Iyer, and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," *IEEE Infocom*, FL, Mar. 2005.

[5] G. Passas and M. Katevenis, "Packet Mode Scheduling in Buffered Crossbar (CICQ) Switches," *IEEE Workshop on High Performance Switching and Routing (HPSR 2006)*, pp. 105-112, Poznan, Poland, June 2006.

[6] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," *IEEE Infocom*, vol. 2, pp. 556-564, Israel, Mar. 2000.

[7] L. Mhamdi, "A Partially Buffered Crossbar Packet Switching Architecture and its Scheduling," *IEEE Int. Symposium on Computers and Commun. (ISCC'08)*, pp. 942 - 948, Morocco, July 2008.

[8] J. Turner, "Strong performance guarantees for asynchronous crossbar schedulers," *Proc. of IEEE INFOCOM'06*, Barcelona, Spain, Apr. 2006.

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

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

[11] M. Katevenis and G. Passas, "Variable-Size Multipacket Segments in Buffered Crossbar (CICQ) Architectures," *Proc. IEEE International Conference on Comm. (ICC'05)*, Seoul, Korea, 16-20 May 2005.

[12] C. Hu, Wenjie Li, X. Chen, and Bin Liu, "Performance comparison between fixed length switching and variable length switching," *Int. Journal Comm. Syst.*, vol. 21, pp. 489-508, 2008

[13] Roy D. Yates and David J. Goodman, "Probability and stochastic processes," *Wiley*, Second Edition, 2005.

[14] T. Javidi, R Magill, and T. Hrabik, "A High Throughput Scheduling Algorithm for a Buffered Crossbar Switch Fabric," *Proc. IEEE, International Conference on Comm. (ICC'01)*, vol. 5, pp. 1586-1591, 2001.

[15] N. McKeown, V. Anantharan, and J. Walrand, "Achieving 100% throughput in an Input-Queued switch," *IEEE Transaction on Communications*, vol. 47, no. 8, pp. 1260-1267, 1999.

[16] G. Passas and M. katevinis, "Asynchronous Operation of Bufferless Crossbar," *Proc of HPSR'07*, Brooklyn, NY, USA, May 2007.

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

[18] M. A. Marsan, et al. "Packet-Mode Scheduling in Input-Queued Cell-Based Switches," *IEEE/ACM Trans. on Netw.*, vol. 10, no. 5, Oct. 2002.

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

[20] 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.

[21] W. Li and B. Liu, "SPF: to improve the performance of packet-mode scheduling," *ELSEVIER, Comp. Com.*, vol. 28, pp. 1380-1391, 2005.