Providing Performance Guarantees for Buffered Crossbar Switches without Speedup

Deng Pan, Zhenyu Yang, Kia Makki, and Niki Pissinou

Florida International University
Miami, FL 33199
{pand, yangz, makkik, pissinou}@fiu.edu

Abstract. Buffered crossbar switches are special crossbar switches with each crosspoint equipped with a small exclusive buffer. The crosspoint buffers decouple input ports and output ports, and simplify switch scheduling. In this paper, we propose a scheduling algorithm called Fair and Localized Asynchronous Packet Scheduling (FLAPS) for buffered crossbar switches, to provide tight performance guarantees. FLAPS needs no speedup for the crossbar and handles variable length packets without segmentation and reassembly (SAR). With FLAPS, each input port and output port independently make scheduling decisions and rely on only local queue statuses. We theoretically show that a crosspoint buffer size of $4L$ is sufficient for FLAPS to avoid buffer overflow, where L is the maximum packet length. In addition, we prove that FLAPS achieves strong stability, and provides bounded delay guarantees. Finally, we present simulation data to verify the analytical results.

Keywords: Buffered crossbar switches, performance guarantees, speedup, stability.

1 Introduction

Crossbar switches provide nonblocking capabilities, and overcome the bandwidth limitation of bus based switches [1]. They have long been used as high speed interconnects in various computing environments, such as Internet routers, computer clusters, and system-on-chip networks. Traditional crossbar switches have no buffers at the crosspoints of the crossbar switching fabric, and packets have to be directly transmitted from input ports to output ports. Such switches work in a synchronous time slot mode and handle only fixed length cells [2]. When variable length packets arrive, they need to 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. This process is called segmentation and reassembly (SAR) [3].

With the development of VLSI technology, it has been feasible to integrate on-chip memories to the crossbar [4] - [7]. Buffered crossbar switches are a special type of crossbar switches, which have a small exclusive buffer at each crosspoint,

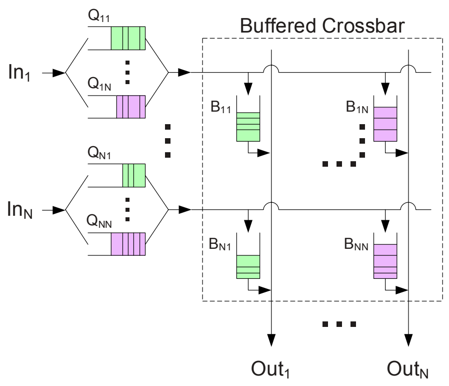

Fig. 1. Structure of buffered crossbar switches

as shown in Figure 1. Crosspoint buffers decouple input ports and output ports, and greatly simplify the scheduling process. Buffered crossbar switches can now directly handle variable length packets and work in an asynchronous mode. To be specific, input ports independently and periodically send packets of arbitrary length to their crosspoint buffers, from where output ports retrieve the packets one by one.

Compared with (fixed length) cell scheduling of unbuffered crossbar switches, (variable length) packet scheduling of buffered crossbar switches 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 padding bits have to be inserted to the last segment to reach the cell length. The padding bits do not contain useful information and waste bandwidth. In the worst case, if all packets 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 its maximum capacity [14]. Second, packet scheduling reduces packet latency, and helps achieve tight performance guarantees. Because there is no SAR, packets arriving at input ports can be immediately transmitted, and packets received at output ports can be immediately sent to the output lines. Third, no extra buffer space is necessary at input ports and output ports for SAR, which lowers hardware cost. In cell scheduling, an input port of an $N \times N$ switch may alternatively send segments of N packets to the N different output ports, and similarly an output port may receive segments from N input ports. Thus, NL buffer space is needed at each input port and output port for SAR, where L is the maximum packet length. Finally, cell scheduling is a special case of packet scheduling, or in other words, packet scheduling can also handle fixed length cells.

The speedup of a switch defines the ratio of the crossbar speed to the input or output port speed. To be specific, speedup of S means that the crossbar has S times bandwidth as that of the input port or output port. Obviously, with the same crossbar, the larger the speedup requirement is, the smaller the switch capacity is. The buffered crossbar switches considered in this paper do not need speedup. Because the crossbar runs at the same speed as the output port, no buffer space is necessary at the output port. When a packet is transmitted to the output port, it will be immediately sent to the next hop via the output line.

There are a number of scheduling algorithms for buffered crossbar switches in the literature. Those algorithms can be classified into two categories: to provide performance guarantees [8] - [14] and to achieve high throughput [15] - [20]. The former requirement is stronger than the latter. In other words, an algorithm with tight performance guarantees usually delivers 100% throughput, but the reverse is not always true. Among the scheduling algorithms providing performance guarantees, some consider cell scheduling [8] - [13]. As discussed in the above, cell scheduling may waste bandwidth due to the padding bits in SAR. Others require speedup of two or more [9] - [14], which reduces the maximum capacity of the switch by half. In particular, two packet scheduling algorithms, Packet GVOQ (PGV) and Packet LOOFA (PLF), were proposed for buffered crossbar switches in [14], and their performance guarantees were analyzed. There are two main differences between the algorithms in [14] and our algorithm. First, PGV and PLF work by emulating push-in-first-out (PIFO) scheduling algorithms for output queued (OQ) switches, which means that they need to maintain the operation of the reference algorithms. Second, PGV and PLF require speedup of two for the crossbar. To the best of our knowledge, there have been no existing packet scheduling algorithms for buffered crossbar switches without speedup.

In this paper, we propose the Fair and Localized Asynchronous Packet Scheduling (FLAPS) algorithm for buffered crossbar switches without speedup. FLAPS allows input ports and output ports to make independent scheduling decisions based on only local information without data exchange. More specifically, an input port needs only the statuses of its input queues, and an output port needs only the states of its crosspoint buffers. FLAPS uses a time stamp based approach to schedule packets. We show that FLAPS has a crosspoint buffer size bound of $4L$, independent of the switch size. Furthermore, we prove that FLAPS achieves strong stability, and provides bounded performance guarantees. Finally, simulations are conducted to verify the analytical results and evaluate the performance of FLAPS.

The rest of the paper is organized as follows. In Section 2, we discuss the ideal fairness model that will be used. In Section 3, we present the FLAPS algorithm. In Section 4, we theoretically analyze the performance of FLAPS, and in Section 5, we present simulation data to verify the analytical results. In Section 6, we conclude the paper.

2 Preliminaries

In this section, we discuss the ideal fairness model that will be used in this paper. To effectively evaluate the fairness performance of a scheduling algorithm, it is necessary to have an ideal fairness model as the comparison reference. A fairness model for packet scheduling can be regarded to have two roles. The first role is to calculate the allocated bandwidth for traffic flows based on their requested bandwidth. The second role is to schedule the packets of different flows to ensure that the actually received bandwidth of each flow is equal to its allocated bandwidth.

Generalized Processor Sharing (GPS) [21] is a widely used fairness model for packet scheduling. When GPS applies to a shared output link, it divides the link bandwidth into multiple logical transmission channels. Each flow has its own logical channel, and the channel bandwidth is proportional to the requested bandwidth of the flow. GPS views flows as fluids of continuous bits, and transmits the packets of a flow in its independent channel. As a result, each flow uses the same amount of bandwidth as that of its allocated bandwidth. To improve utilization, when a flow temporarily becomes empty, GPS will reallocate the leftover bandwidth of the empty flow to the remaining backlogged flows in proportion to their requested bandwidth.

As can be seen, when GPS serves a shared output link, it allocates available bandwidth, including leftover bandwidth, in the proportional manner. However, simple proportional bandwidth allocation is not proper for switch scheduling [22] [23]. The reason is that, while flows of a shared output link are constrained only by the link bandwidth, flows of a switch are subject to two bandwidth constraints: the available bandwidth at both the input port and output port of the flow. Naive bandwidth allocation at the output port may make the flows violate the bandwidth constraints at their input ports, and vice versa.

Fair bandwidth allocation for switches is an interesting problem, and there are algorithms [22] [23] in the literature to solve it. In this paper, we assume that bandwidth allocation has been calculated using such algorithms, and the scheduling algorithms just schedule packets to ensure the allocated bandwidth of each flow. Also, when a flow of the switch temporarily becomes empty, we do not assume that its allocated bandwidth is immediately reallocated. Instead, the bandwidth allocation algorithms will consider the leftover bandwidth in the next calculation. Bandwidth allocation is recalculated when requested bandwidth changes or existing backlogged flows become empty. Given the calculated bandwidth allocation, GPS can divide the bandwidth of a switch into logical channels, as shown in Figure 2. Each flow has an independent logical channel,

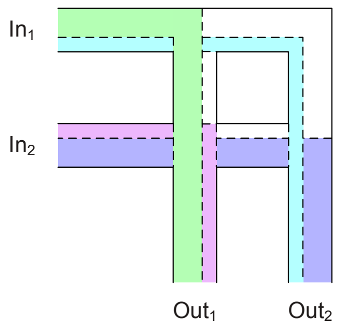

Fig. 2. GPS used for switch scheduling

and the channel bandwidth is equal to the allocated bandwidth of the flow. Thus, traffic of the flow can smoothly stream from the input port to the output port.

To sum up, we use GPS as the ideal fairness model only for its second role, i.e., given the allocated bandwidth, to compare the received bandwidth of a flow in our algorithm and in GPS.

3 Fair and Localized Asynchronous Packet Scheduling

In this section, we present the Fair and Localized Asynchronous Packet Scheduling (FLAPS) algorithm.

The switch structure that we consider is shown in Figure 1. N input ports and N output ports are connected by a buffered crossbar, which has no speedup. Denote the i^{th} input port as In_i and the j^{th} output port as Out_j. Use R to represent the available bandwidth of each input port and output port, and the crossbar also has bandwidth R. Each input port has a buffer organized as virtual output queues (VOQ) [24], i.e., there are N virtual queues to store the packets destined to the N different output ports. Denote the virtual queue at In_i for packets to Out_j as Q_{ij}. Each crosspoint has a small exclusive buffer. Denote the crosspoint buffer connecting In_i and Out_j as B_{ij}. Output ports have no buffers. After a packet arrives at the switch, it is first stored in the input queue, and waits to be sent to the crosspoint buffer. The packet will then be sent from the crosspoint buffer to the output port and immediately delivered to the output line. We say a packet arrives at or departs from a buffer when its last bit arrives at or departs from the buffer.

Define the traffic from In_i to Out_j to be a flow F_{ij}. Because we consider performance guarantees in this paper, each flow has explicit allocated bandwidth, and the objective of the scheduling algorithm is to ensure that each flow receives the same amount of bandwidth as that of its allocated bandwidth. Use $r_{ij}(t)$ to represent the allocated bandwidth of F_{ij}, which is a function of time t with discrete values in practice. Bandwidth allocation is calculated by the algorithms mentioned in Section 2. The calculated bandwidth allocation should be feasible, i.e., no over-subscription at any input port or output port

$$\forall i, \sum_{x=1}^{N} r_{ix}(t) \leq R, \text{ and } \forall j, \sum_{x=1}^{N} r_{xj}(t) \leq R \tag{1}$$

The feasibility requirement is only for bandwidth allocation. It is necessary because it is impossible to allocate more bandwidth than what is actually available. However, it does not mean that no temporary overload is allowed for an input port or output port. As will be seen in Section 4.2, we use the extended leaky bucket scheme for flow admission control. It allows any flow to be overloaded for an arbitrarily long period with an arbitrarily large but finite burst.

There are two types of scheduling in the switch, which we call input scheduling and output scheduling. In input scheduling, an input port selects a packet from one of its input queues, and sends it to the corresponding crosspoint buffer.

In output scheduling, an output port selects a packet from one of its crosspoint buffers, and sends it to the output line. When we need to differentiate the scheduling algorithms used for input scheduling and output scheduling, we use the notation "A-B". A is the scheduler for input scheduling, and B is the scheduler for output scheduling. A and B could be either FLAPS or GPS. If we do not care the scheduler for output scheduling, we use a * mark for B. For example, GPS-GPS means that GPS is used for both input scheduling and output scheduling. In such a scenario, each flow has an independent channel, and its traffic moves smoothly from the input output to the output port without buffering in the middle, as shown in Figure 2.

Input scheduling and output scheduling of FLAPS rely on only local information, and are conducted in an asynchronous and distributed manner. To be specific, an input port needs only the statuses of the queues in its input buffer, and does not exchange information with any crosspoint buffer or output port. Similarly, an output port needs only the statuses of its crosspoint buffers.

We first describe the input scheduling of FLAPS, which uses time stamps as scheduling criteria. There are two types of time stamps. For easy representation, we denote the k^{th} arrived packet of F_{ij} as P_{ij}^k. The first time stamp for P_{ij}^k is called virtual input start time, denoted as $\widehat{IS}_{ij}^k$, which is the service start time of P_{ij}^k at the input port in GPS-*. The second time stamp is virtual input finish time, denoted as $\widehat{IF}_{ij}^k$, which is the service finish time of P_{ij}^k at the input port in GPS-*. In other words, if GPS is the scheduler for input scheduling, $\widehat{IS}_{ij}^k$ and $\widehat{IF}_{ij}^k$ are the time that the first bit and last bit of P_{ij}^k leave Q_{ij}, respectively. $\widehat{IS}_{ij}^k$ can be calculated as follows

$$\widehat{IS}_{ij}^k = \max\left(IA_{ij}^k, \widehat{IF}_{ij}^{k-1}\right) \tag{2}$$

where IA_{ij}^k is the arrival time of P_{ij}^k at the input port. $\widehat{IF}_{ij}^k$ satisfies the following relationship

$$\int_{\widehat{IS}_{ij}^k}^{\widehat{IF}_{ij}^k} r_{ij}(x)dx = L_{ij}^k \tag{3}$$

where L_{ij}^k is the length of P_{ij}^k. Because $r_{ij}(t)$ has only discrete values in practice, $\widehat{IF}_{ij}^k$ can be easily calculated. For example, if $r_{ij}(t)$ is a constant r_{ij} during $\left[\widehat{IS}_{ij}^k, \widehat{IF}_{ij}^k\right]$, $\widehat{IF}_{ij}^k$ can be calculated as

$$\widehat{IF}_{ij}^k = \widehat{IS}_{ij}^k + \frac{L_{ij}^k}{r_{ij}} \tag{4}$$

In the first step of input scheduling of FLAPS, In_i identifies eligible packets. A packet P_{ij}^k is eligible for input scheduling if its virtual input start time $\widehat{IS}_{ij}^k$ is

Table 1. Input Scheduling of FLAPS

```
for $In_i$ do {
  while true do {
    if there are packets in local input queues with virtual input
      start time smaller than or equal to current system time {
      select among such packets the one with the smallest
        virtual input finish time, say $P_{ij}^k$;
      send $P_{ij}^k$ to crosspoint buffer $B_{ij}$;
      // system time progressing by $\frac{P_{ij}^k}{R}$
    }
    else {
      wait until the next earliest virtual input start time;
    }
  }
}
```

smaller than or equal to the current system time t. In other words, a packet that has started transmission in GPS-* is eligible in FLAPS-*. If there exist eligible packets in the input buffer, In_i will select among such packets the one P_{ij}^k with the smallest virtual input finish time $\widehat{IF}_{ij}^k$, and send it to B_{ij}. If there are no eligible packets, In_i will wait until the next earliest virtual input start time of a packet. Note that when In_i is waiting for an eligible packet, if an empty input queue has a new incoming packet, whose virtual input start time is equal to its arrival time, In_i should immediately start transmitting this new packet. For easy reading, the pseudo code description for input scheduling of FLAPS is given in Table 1.

We denote the actual input start time and finish time of P_{ij}^k in FLAPS-* as IS_{ij}^k and IF_{ij}^k, which are the time that the first bit and the last bit of P_{ij}^k leave Q_{ij} in FLAPS-*, respectively. Apparently

$$IF_{ij}^k = IS_{ij}^k + \frac{L_{ij}^k}{R} \tag{5}$$

Output scheduling of FLAPS is similar to input scheduling. There are also several time stamps for P_{ij}^k. The virtual output start time $\widehat{OS}_{ij}^k$ and virtual output finish time $\widehat{OF}_{ij}^k$ are the time that the first bit and last bit of P_{ij}^k leave B_{ij} in FLAPS-GPS, respectively. In other words, after P_{ij}^k is delivered to B_{ij} by FLAPS, if GPS is the scheduler for output scheduling, P_{ij}^k will start transmission at $\widehat{OS}_{ij}^k$ and finish at $\widehat{OF}_{ij}^k$. $\widehat{OS}_{ij}^k$ is calculated as

$$\widehat{OS}_{ij}^k = \max\left(OA_{ij}^k, \widehat{OF}_{ij}^{k-1}\right) \tag{6}$$

Table 2. Output Scheduling of FLAPS

```
for $Out_j$ do {
    while true do {
        if there are packets in local crosspoint buffers with virtual
            output start time smaller than or equal to current system time {
            select among such packets the one with the smallest
                virtual output finish time, say $P_{ij}^k$;
            send $P_{ij}^k$ to the output line;
            // system time progressing by $\frac{P_{ij}^k}{R}$
        }
        else {
            wait until the next earliest virtual output start time;
        }
    }
}
```

where OA_{ij}^k is the arrival time of P_{ij}^k at B_{ij} in FLAPS-*, and is equal to IF_{ij}^k by neglecting the propagation delay. $\widehat{OF}_{ij}^k$ satisfies the following relationship

$$\int_{\widehat{OS}_{ij}^k}^{\widehat{OF}_{ij}^k} r_{ij}(x)dx = L_{ij}^k \tag{7}$$

Similarly, in output scheduling of FLAPS, Out_j first identifies eligible packets, and a packet P_{ij}^k is eligible if its virtual output start time $\widehat{OS}_{ij}^k$ is smaller than or equal to the current system time t. If there are eligible packets in the crosspoint buffers, Out_j retrieves the one P_{ij}^k with the smallest virtual output finish time $\widehat{OF}_{ij}^k$, and sends it to the output line. Otherwise, it waits until there is an eligible packet. The pseudo code description for output scheduling of FLAPS is given in Table 2.

Correspondingly, OS_{ij}^k and OF_{ij}^k are the actual output start and finish time of P_{ij}^k, which are the time that the first bit and the last bit of P_{ij}^k leave B_{ij} in FLAPS-FLAPS, respectively. It is obvious that

$$OF_{ij}^k = OS_{ij}^k + \frac{L_{ij}^k}{R} \tag{8}$$

4 Performance Analysis

In this section, we theoretically analyze the performance of FLAPS. We show that FLAPS has a bounded crosspoint buffer size, achieves strong stability, and provides tight delay guarantees.

4.1 Crosspoint Buffer Size Bound

To avoid overflow at crosspoint buffers, we would like to find the maximum number of bits buffered at any crosspoint.

Based on the description of the FLAPS algorithm, we have the following properties.

Property 1. *For any packet, its actual input start time in FLAPS-* is larger than or equal to its virtual input start time in GPS-*, i.e.,*

$$IS_{ij}^{k} \geq \widehat{IS}_{ij}^{k} \tag{9}$$

Property 2. *For any packet, its actual output start time in FLAPS-FLAPS is larger than or equal to its virtual output start time in FLAPS-GPS, i.e.,*

$$OS_{ij}^{k} \geq \widehat{OS}_{ij}^{k} \tag{10}$$

First, we define some notations for input scheduling. We say that Q_{ij} is backlogged at time t, if there exists k such that $\widehat{IS}_{ij}^{k} \leq t \leq \widehat{IF}_{ij}^{k}$. Intuitively, Q_{ij} is backlogged at t if Q_{ij} has buffered bits at t in GPS-*. Define $\hat{q}_{ij}(t)$ to represent the backlog status of Q_{ij} at t. $\hat{q}_{ij}(t) = 1$ or 0 means that Q_{ij} is backlogged or empty at t.

Use $toB_{ij}(t_1, t_2)$ and $\widehat{toB}_{ij}(t_1, t_2)$ to represent the numbers of bits transmitted by F_{ij} from In_i to B_{ij} during interval $[t_1, t_2]$ in FLAPS-* and GPS-*, respectively. Based on the definition of GPS, $\widehat{toB}_{ij}(t_1, t_2)$ can be calculated as

$$\widehat{toB}_{ij}(t_1, t_2) = \int_{t_1}^{t_2} r_{ij}(x)\hat{q}_{ij}(x)dx \tag{11}$$

Next, we define some corresponding notations for output scheduling. We say that B_{ij} is backlogged at time t, if there exists k such that $\widehat{OS}_{ij}^{k} \leq t \leq \widehat{OF}_{ij}^{k}$. Define $\hat{b}_{ij}(t)$ to represent the backlog status of B_{ij} at t. $\hat{b}_{ij}(t) = 1$ or 0 means that B_{ij} is backlogged or empty at t.

Use $toO_{ij}(t_1, t_2)$ and $\widehat{toO}_{ij}(t_1, t_2)$ to represent the numbers of bits transmitted by F_{ij} from B_{ij} to Out_j during interval $[t_1, t_2]$ in FLAPS-FLAPS and FLAPS-GPS, respectively. $\widehat{toO}_{ij}(t_1, t_2)$ can be calculated as

$$\widehat{toO}_{ij}(t_1, t_2) = \int_{t_1}^{t_2} r_{ij}(x)\hat{b}_{ij}(x)dx \tag{12}$$

The following lemma gives the relationship between the service time of a packet in FLAPS-* and GPS-*.

Lemma 1. *For any packet, its actual input start time in FLAPS-* is less than or equal to its virtual input finish time in GPS-*, i.e.,*

$$IS_{ij}^{k} \leq \widehat{IF}_{ij}^{k} \tag{13}$$

The proofs of Lemmas 1 to 4 are similar to the performance analysis of WF^{2}Q in [25]. They are omitted in this paper due to space limitations.

Correspondingly, there is a lemma for output scheduling.

Lemma 2. *For any packet, its actual output start time in FLAPS-FLAPS is less than or equal to its virtual output finish time in FLAPS-GPS, i.e.,*

$$OS_{ij}^k \leq \widehat{OF}_{ij}^k \tag{14}$$

The next lemma compares $toB_{ij}(t_1, t_2)$ and $\widehat{toB}_{ij}(t_1, t_2)$.

Lemma 3. *During interval* $[0, t]$*, the difference between the number of bits sent from input port* In_i *to crosspoint buffer* B_{ij} *in FLAPS-* and GPS-* is greater than or equal to* $-L$ *and less than or equal to* L*, i.e.,*

$$-L \leq toB_{ij}(0, t) - \widehat{toB}_{ij}(0, t) \leq L \tag{15}$$

For output scheduling, there is a similar lemma as follows.

Lemma 4. *During interval* $[0, t]$*, the difference between the number of bits sent from crosspoint buffer* B_{ij} *to output port* Out_j *in FLAPS-FLAPS and FLAPS-GPS is greater than or equal to* $-L$ *and less than or equal to* L*, i.e.,*

$$-L \leq toO_{ij}(0, t) - \widehat{toO}_{ij}(0, t) \leq L \tag{16}$$

The next lemma compares the number of bits transmitted by the same flow in the input scheduling of GPS-* and the output scheduling of FLAPS-GPS.

Lemma 5. *During interval* $[0, t]$*, the number of bits transmitted by flow* F_{ij} *from input port* In_i *to crosspoint buffer* B_{ij} *in GPS-* is less than or equal to that from crosspoint buffer* B_{ij} *to output port* Out_j *in FLAPS-GPS plus* $2L$*, i.e.,*

$$\widehat{toB}_{ij}(0, t) \leq \widehat{toO}_{ij}(0, t) + 2L \tag{17}$$

Proof. Assume that B_{ij} in FLAPS-GPS is empty immediately before time s and is continuously backlogged during $[s, t]$. If B_{ij} is not backlogged at t, then $s = t$.

By Lemma 3, we have $toB_{ij}(0, s) \geq \widehat{toB}_{ij}(0, s) - L$. Because B_{ij} is empty before s and backlogged after s in FLAPS-GPS, all packets arriving at B_{ij} before s have been transmitted to Out_j, and a new packet arrives at B_{ij} at s. Thus

$$\begin{aligned} \widehat{toO}_{ij}(0, s) \geq\ & toB_{ij}(0, s) - L \\ \geq\ & \widehat{toB}_{ij}(0, s) - 2L \end{aligned} \tag{18}$$

On the other hand, because B_{ij} is continuously backlogged during $[s, t]$, $\hat{b}_{ij}(t)$ is equal to 1 in the interval. Therefore

$$\begin{aligned}
\widehat{toO}_{ij}(s,t) &= \int_s^t r_{ij}(x)\hat{b}_{ij}(x)dx \\
&= \int_s^t r_{ij}(x)dx \\
&\geq \int_s^t r_{ij}(x)\hat{q}_{ij}(x)dx \\
&= \widehat{toB}_{ij}(s,t) \qquad (19)
\end{aligned}$$

Adding (18) and (19), we obtain

$$\widehat{toO}_{ij}(0,t) \geq \widehat{toB}_{ij}(0,t) - 2L \qquad (20)$$

The following theorem gives the bound for the crosspoint buffer size.

Theorem 1. *In FLAPS-FLAPS, the maximum number of bits buffered at a crosspoint buffer is upper bounded by* $4L$*, i.e.,*

$$toB_{ij}(0,t) - toO_{ij}(0,t) \leq 4L \qquad (21)$$

Proof. By Lemma 4,

$$toO_{ij}(0,t) + L \geq \widehat{toO}_{ij}(0,t) \qquad (22)$$

By Lemma 5,

$$\widehat{toO}_{ij}(0,t) + 2L \geq \widehat{toB}_{ij}(0,t) \qquad (23)$$

By Lemma 3,

$$\widehat{toB}_{ij}(0,t) + L \geq toB_{ij}(0,t) \qquad (24)$$

Summing the above equations, we have proved the theorem.

4.2 Switch Stability

We have shown in the above that FLAPS has a bounded crosspoint buffer size. In this subsection, we show that the lengths of input virtual queues are finite, and thus FLAPS achieves strong stability.

As discussed in Section 3, because each flow is allocated a specific amount of bandwidth, it is necessary to have admission control for the flow to avoid over-subscription. The leaky bucket scheme [1] is a widely used traffic shaping scheme, and we will use it for admission control. In the classical definition of a leaky bucket, the flow rate is a constant, which we extend in this paper to be a variable. The reason is that the allocated bandwidth of a flow may change after bandwidth allocation calculations. Use $toI_{ij}(t_1, t_2)$ to denote the number of incoming bits of F_{ij} during interval $[t_1, t_2]$. If F_{ij} is leaky bucket $(r_{ij}(t), \sigma_{ij})$ complaint, then during any interval $[t_1, t_2]$

$$toI_{ij}(t_1, t_2) \leq \int_{t_1}^{t_2} r_{ij}(x)dx + \sigma_{ij} \tag{25}$$

where σ_{ij} can be an arbitrary positive constant and is called the burst size of F_{ij}. Intuitively, during any time interval, F_{ij} can have σ_{ij} more incoming traffic than what it can transmit.

Use $Q_{ij}(t)$ to represent the queue occupancy of Q_{ij} at t, i.e. the number of bits buffered in Q_{ij} at t. Define $X(t) = (Q_{11}(t), ..., Q_{ij}(t), ..., Q_{NN}(t))$, and use $||X||$ to represent the Euclidean norm of vector $X = (x_1, x_2, ..., x_n)$, i.e.

$$||X|| = \sqrt{\sum_{i=1}^{n} x_i^2} \tag{26}$$

Following the definition in [26], we say that a system of queues is strongly stable if

$$\lim_{n\to\infty} \sup E[||X(t)||] < \infty \tag{27}$$

Note that strong stability implies 100% throughput [26].

Theorem 2. *When flows are leaky bucket complaint, FLAPS is strongly stable.*

Proof. Assume that flow F_{ij} is leaky bucket $(r_{ij}(t), \sigma_{ij})$ compliant. Also assume that Q_{ij} is empty immediately before s and continuously backlogged during $[s, t]$. This indicates that all packets of F_{ij} arriving at Q_{ij} before s have finished transmission by s in GPS-*, and the next packet has not arrived. Therefore

$$toI_{ij}(0, s) \leq \widehat{toB}_{ij}(0, s) + L \tag{28}$$

During $[s, t]$, Q_{ij} is continuously backlogged, and thus

$$\widehat{toB}_{ij}(s, t) = \int_{s}^{t} r_{ij}(x)\hat{q}_{ij}(x)dx = \int_{s}^{t} r_{ij}(x)dx \tag{29}$$

Because the arrival traffic is leaky bucket $(r_{ij}(t), \sigma_{ij})$ compliant, we have

$$toI_{ij}(s, t) \leq \int_{s}^{t} r_{ij}(x)dx + \sigma_{ij} \tag{30}$$

By (28), (29), and (30)

$$toI_{ij}(0, t) \leq \widehat{toB}_{ij}(0, t) + L + \sigma_{ij} \tag{31}$$

We know from Lemma 3 that $\widehat{toB}_{ij}(0, t) \leq toB_{ij}(0, t) + L$. Thus

$$\begin{aligned} toI_{ij}(0, t) &\leq \widehat{toB}_{ij}(0, t) + L + \sigma_{ij} \\ &\leq toB_{ij}(0, t) + 2L + \sigma_{ij} \end{aligned} \tag{32}$$

By the definition of $Q_{ij}(t)$, we can obtain

$$Q_{ij}(t) = toI_{ij}(0,t) - toB_{ij}(0,t) \leq 2L + \sigma_{ij} \tag{33}$$

Since both L and σ_{ij} are finite, we have

$$||X(t)|| = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} Q_{ij}(t)^2} < \infty \tag{34}$$

4.3 Delay Guarantees

In this subsection, we show that FLAPS can provide bounded delay guarantees. For easy analysis, we assume that the allocated bandwidth $r_{ij}(t)$ of F_{ij} is a constant r_{ij} during interval
$\left[\min\left(IS_{ij}^k, \widehat{IS}_{ij}^k\right), \max\left(OF_{ij}^k, \widehat{OF}_{ij}^k\right)\right]$.

Use $\widetilde{OF}_{ij}^k$ to denote the departure time of P_{ij}^k in GPS-GPS. By neglecting the propagation delay, we have $\widetilde{OF}_{ij}^k = \widehat{IF}_{ij}^k$. Similarly, OF_{ij}^k is the departure time of packet P_{ij}^k in FLAPS-FLAPS, if the propagation delay is neglected.

Theorem 3. *The difference between the departure time of packet P_{ij}^k in FLAPS-FLAPS and GPS-GPS is greater than or equal to $-L_{ij}^k\left(\frac{1}{r_{ij}} - \frac{2}{R}\right)$ and less than or equal to $L\left(\frac{3}{r_{ij}} + \frac{2}{R}\right)$, i.e.*

$$-L_{ij}^k\left(\frac{1}{r_{ij}} - \frac{2}{R}\right) \leq OF_{ij}^k - \widetilde{OF}_{ij}^k \leq L\left(\frac{3}{r_{ij}} + \frac{2}{R}\right) \tag{35}$$

Proof. First, we prove $OF_{ij}^k - \widetilde{OF}_{ij}^k \geq -L_{ij}^k\left(\frac{1}{r_{ij}} - \frac{2}{R}\right)$. It is obvious that

$$OF_{ij}^k \geq OA_{ij}^k + \frac{L_{ij}^k}{R} = IF_{ij}^k + \frac{L_{ij}^k}{R} \tag{36}$$

Based on Property 1, we know $\widehat{IS}_{ij}^k \leq IS_{ij}^k$ or in other words $\widehat{IF}_{ij}^k - \frac{L_{ij}^k}{r_{ij}} \leq IF_{ij}^k - \frac{L_{ij}^k}{R}$, and thus we obtain

$$\begin{aligned}\widetilde{OF}_{ij}^k &= \widehat{IF}_{ij}^k \\ &\leq IF_{ij}^k + L_{ij}^k\left(\frac{1}{r_{ij}} - \frac{1}{R}\right) \\ &\leq OF_{ij}^k + L_{ij}^k\left(\frac{1}{r_{ij}} - \frac{2}{R}\right)\end{aligned} \tag{37}$$

Next, we prove $OF_{ij}^k - \widetilde{OF}_{ij}^k \leq L\left(\frac{3}{r_{ij}} + \frac{2}{R}\right)$. Based on Lemma 2, we know $OS_{ij}^k \leq \widehat{OF}_{ij}^k$ and thus $OF_{ij}^k \leq \widehat{OF}_{ij}^k + \frac{L_{ij}^k}{R}$. By Lemma 5, we have $\widehat{toB}_{ij}(0,t) - \widehat{toO}_{ij}(0,t) \leq$

$2L$, and by Lemma 3, we have $toB_{ij}(0,t) - \widehat{toB}_{ij}(0,t) \leq L$. Combining them, we obtain $toB_{ij}(0,t) - \widehat{toO}_{ij}(0,t) \leq 3L$. This indicates that, after P_{ij}^k arrives at B_{ij}, the maximum queue length at B_{ij} in FLAPS-GPS is $3L$. Because B_{ij} is served by GPS output scheduling with fixed allocated bandwidth r_{ij} in FLAPS-GPS, we have

$$\widehat{OF}_{ij}^k \leq OA_{ij}^k + \frac{3L}{r_{ij}} \leq IF_{ij}^k + \frac{3L}{r_{ij}} \tag{38}$$

By Lemma 1, $IS_{ij}^k \leq \widehat{IF}_{ij}^k$ and thus $IF_{ij}^k \leq \widehat{IF}_{ij}^k + \frac{L_{ij}^k}{R}$. Combing the above equations, we obtain

$$\begin{aligned} OF_{ij}^k \leq & \ \widehat{OF}_{ij}^k + \frac{L_{ij}^k}{R} \\ \leq & \ IF_{ij}^k + \frac{3L}{r_{ij}} + \frac{L_{ij}^k}{R} \\ \leq & \ \widehat{IF}_{ij}^k + \frac{3L}{r_{ij}} + \frac{2L_{ij}^k}{R} \\ \leq & \ \widetilde{OF}_{ij}^k + L\left(\frac{3}{r_{ij}} + \frac{2}{R}\right) \end{aligned} \tag{39}$$

5 Simulation Results

We have conducted simulations to verify the analytical results obtained in Section 4 and evaluate the performance of FLAPS.

In the simulations, we consider a 16×16 buffered crossbar switch without speedup. Each input port and output port has bandwidth of 1G bps. Since FLAPS can directly handle variable length packets, we set packet length to be uniformly distributed between 40 and 1500 bytes [27]. For bandwidth allocation, we use the same model as that in [15] and [17]. The allocated bandwidth $r_{ij}(t)$ of flow F_{ij} at time t is defined by an unbalanced probability w as follows

$$r_{ij}(t) = \begin{cases} R\left(w + \frac{1-w}{N}\right), & \text{if } i = j \\ R\frac{1-w}{N}, & \text{if } i \neq j \end{cases} \tag{40}$$

When $w = 0$, In_i has the same amount of allocated bandwidth at each output port. Otherwise, In_i has more allocated bandwidth at Out_i, which is called the hotspot destination. Arrival of a flow F_{ij} is constrained by a leaky bucket $(l * r_{ij}(t), \sigma_{ij})$, where l is the effective load. We set the burst size σ_{ij} of every flow to a fixed value of 10,000 bytes, and the burst may arrive at any time during a simulation run. We use two traffic patterns in the simulations. For traffic pattern one, each flow has fixed allocated bandwidth during a single simulation run. l is fixed to 1 and w is one of the 11 possible values from 0 to 1 with a step of 0.1. For traffic pattern two, a flow has variable allocated bandwidth. l is one of the 10 possible values from 0.1 to 1 with a step of 0.1, and for a specific l value, a random permutation of the 11 different w values is used. Each simulation run lasts for 10 seconds.

5.1 Crosspoint Buffer Size

Theorem 1 in Section 4.1 gives the bound of the crosspoint buffer size as $4L$. In this subsection, we look at the maximum and average crosspoint buffer occupancies in the simulations.

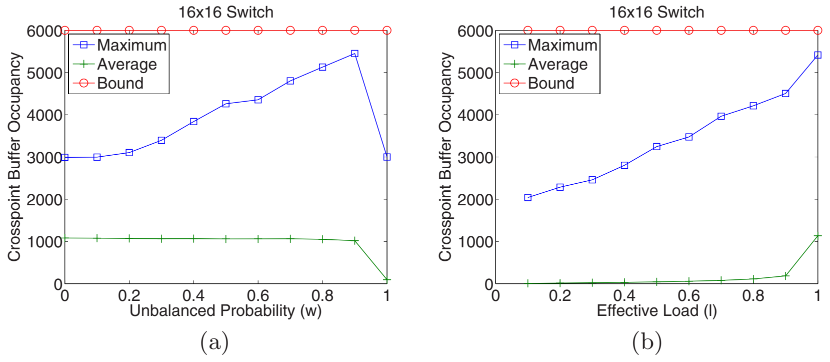

Fig. 3. Crosspoint buffer occupancy of FLAPS. (a) With different unbalanced probabilities. (b) With different loads.

Figure 3(a) shows the maximum and average crosspoint buffer occupancies under traffic pattern one. As can be see, the maximum occupancy is always smaller than the theoretical bound. It grows as the unbalanced probability increases, but suddenly drops when the unbalanced probability becomes 1. This is because when the unbalanced probability is 1, all packets of In_i go to Out_i. Thus, there is no switching necessary, and the crosspoint buffer occupancy becomes smaller. For the average occupancy, it does not change significantly with different unbalanced probabilities, and drops when the unbalanced probability becomes 1 for the same reason. We can find that the average occupancy is more affected by the load than the unbalanced probability. Figure 3(b) shows the maximum and average crosspoint buffer occupancies under traffic pattern two. We can see that the maximum occupancy increases as the load increases, but does not exceed the theoretical bound. On the other hand, the average occupancy does not change much and is smaller than 200 bytes before the load increase to 1. This also confirms the previous observation that the average occupancy is determined by the load.

5.2 Throughput

Theorem 2 in Section 4.2 shows that FLAPS achieves strong stability, which implies 100% throughput. Next, we present the simulation data on throughput of FLAPS.

Figure 4(a) shows the throughput under traffic pattern one. We can see that the throughput for all unbalanced probabilities is greater than 99.99%, which

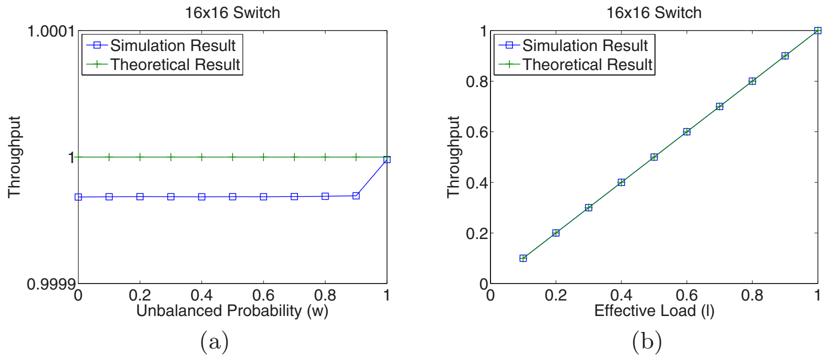

Fig. 4. Throughput of FLAPS. (a) With different unbalanced probabilities. (b) With different loads.

demonstrates that FLAPS practically achieves 100% throughput. Figure 4(b) shows the throughput under traffic pattern two. As can be seen, the throughput grows consistently with the effective load, and finally reaches 1.

5.3 Jitter

In this subsection, we present the simulation data on jitter, which is the difference between the packet departure time in FLAPS and GPS. Theorem 3 in Section 4.3 gives the lower bound and upper bound for the jitter of packet P_{ij}^k. Because Theorem 3 assumes fixed allocated bandwidth r_{ij}, we use only traffic pattern one for this part of simulations. Note that the lower bound value depends on the packet length L_{ij}^k. For easy plotting of the figure, we calculate the jitter lower bound for all packets of flow F_{ij} as follows

$$-L_{ij}^k\left(\frac{1}{r_{ij}}-\frac{2}{R}\right) \geq \begin{cases} -L\left(\frac{1}{r_{ij}}-\frac{2}{R}\right), & \text{if } r_{ij} \leq \frac{R}{2} \\ 0, & \text{if } r_{ij} > \frac{R}{2} \end{cases} \tag{41}$$

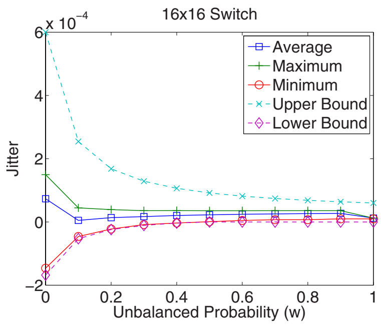

Fig. 5. Jitter of FLAPS with different unbalanced probabilities

Figure 5 shows the minimum, maximum, and average jitters of a representative flow F_{11} under traffic pattern one. We can see that the minimum jitter is almost coincident with but always greater than the lower bound, and the maximum jitter is always less than the upper bound. As the unbalanced probability increases, the minimum jitter increases and the maximum jitter decreases. For most of the time, the average jitter is very close to zero, indicating that FLAPS and GPS have similar average packet delay.

6 Conclusions

Buffered crossbar switches are special crossbar switches with crosspoint buffers. The introduction of crosspoint buffers greatly simplifies the scheduling process. In this paper, we have proposed the Fair and Localized Asynchronous Packet Scheduling (FLAPS) algorithm, which does not require speedup for the crossbar and can directly handle variable length packets without segmentation and reassembly (SAR). FLAPS uses a time stamp based approach for both input scheduling and output scheduling. We theoretically analyze the performance of FLAPS, and show that it has a crosspoint buffer size bound of $4L$, independent of the switch size. We also prove that it achieves strong stability, and provides bounded delay guarantees. Finally, we present simulation data and show that they are consistent with the analytical results.

References

1. Kurose, J., Ross, K.: Computer networking: a top-down approach, 4th edn. Addison Wesley, Reading (2007)
2. McKeown, N.: A fast switched backplane for a gigabit switched router. Business Communications Review 27(12) (1997)
3. Katevenis, M., Passas, G.: Variable-size multipacket segments in buffered crossbar (CICQ) architectures. In: IEEE ICC 2005, Seoul, Korea (May 2005)
4. Kornaros, G.: BCB: a buffered crossBar switch fabric utilizing shared memory. In: 9th EUROMICRO Conference on Digital System Design, Croatia, August 2006, pp. 180–188 (2006)
5. Mhamdi, L., Kachris, C., Vassiliadis, S.: A reconfigurable hardware based embedded scheduler for buffered crossbar switches. In: 14th ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, February 2006, pp. 143–149 (2006)
6. Papaefstathiou, I., Kornaros, G., Chrysos, N.: Using Buffered crossbars for chip interconnection. In: 17th Great Lakes Symposium on VLSI, Stresa-Lago Maggiore, Italy, March 2007, pp. 90–95 (2007)
7. Yoshigoe, K., Christensen, K., Jacob, A.: The RR/RR CICQ switch: hardware design for 10-Gbps link speed. In: 22nd IEEE International Performance, Computing, and Communications Conference, Phoenix, AZ, April 2003, pp. 481–485 (2003)
8. He, S., et al.: On Guaranteed Smooth Switching for Buffered Crossbar Switches. IEEE/ACM Transactions on Networking 16(3), 718–731 (2008)
9. Magill, B., Rohrs, C., Stevenson, R.: Output-queued switch emulation by fabrics with limited memory. IEEE Journal on Selected Areas in Communications 21(4), 606–615 (2003)

10. Mhamdi, L., Hamdi, M.: Output queued switch emulation by a one-cell-internally buffered crossbar switch. In: IEEE GLOBECOM 2003, San Francisco, CA (December 2003)
11. Stephens, D., Zhang, H.: Implementing distributed packet fair queueing in a scalable switch architecture. In: IEEE INFOCOM 1998, San Francisco, CA (March 1998)
12. Chuang, S., Iyer, S., McKeown, N.: Practical algorithms for performance guarantees in buffered crossbars. In: IEEE INFOCOM 2005, Miami, FL (March 2005)
13. Pan, D., Yang, Y.: Providing flow based performance guarantees for buffered crossbar switches. In: IEEE IPDPS 2008, Miami, FL (April 2008)
14. Turner, J.: Strong performance guarantees for asynchronous crossbar schedulers. IEEE/ACM Transactions on Networking (to appear, 2009)
15. Rojas-Cessa, R., Oki, E., Jing, Z., Chao, H.: CIXB-1: Combined input-once-cell-crosspoint buffered switch. In: IEEE HPSR 2001, Dallas, TX (July 2001)
16. Rojas-Cessa, R., Oki, E., Chao, H.: CIXOB-k: Combined input-crosspoint-output buffered packet switch. In: IEEE Globecom 2001, San Antonio, TX (November 2001)
17. Mhamdi, L., Hamdi, M.: MCBF: a high-performance scheduling algorithm for buffered crossbar switches. IEEE Communications Letters 7(9), 451–453 (2003)
18. Zhang, X., Bhuyan, L.: An efficient scheduling algorithm for combined-input-crosspoint-queued (CICQ) switches. In: IEEE Globecom 2004, Dallas, TX (November 2004)
19. Katevenis, M., Passas, G., Simos, D., Papaefstathiou, I., Chrysos, N.: Variable packet size buffered crossbar (CICQ) switches. In: Proc. IEEE ICC 2004, Paris, France (June 2004)
20. Pan, D., Yang, Y.: Localized independent packet scheduling for buffered crossbar switches. IEEE Transactions on Computers 58(2), 260–274 (2009)
21. Parekh, A., Gallager, R.: A generalized processor sharing approach to flow control in integrated services networks: the single node case. IEEE/ACM Trans. Networking 1(3), 344–357 (1993)
22. Pan, D., Yang, Y.: Max-min fair bandwidth allocation algorithms for packet switches. In: IEEE IPDPS 2007, Long Beach, CA (March 2007)
23. Hosaagrahara, M., Sethu, H.: Max-min fairness in input-queued switches. IEEE Transactions on Parallel and Distributed Systems 19(4), 462–475 (2008)
24. McKeown, N., Mekkittikul, A., Anantharam, V., Walrand, J.: Achieving 100% throughput in an input queued switch. IEEE Trans. Commun. 47(8), 1260–1267 (1999)
25. Bennett, J., Zhang, H.: WF2Q: worst-case fair weighted fair queueing. In: IEEE INFOCOM 1996, San Francisco, CA (March 1996)
26. Leonardi, E., Mellia, M., Neri, F., Marsan, M.: On the stability of input-queued switches with speed-up. IEEE/ACM Trans. Networking 9(1), 104–118 (2001)
27. Farleigh, C., et al.: Packet-level traffic measurements from the Sprint IP backbone. IEEE Network 17(6), 6–16 (2003)