Fair Queueing based Packet Scheduling for Buffered Crossbar Switches

Deng Pan, Kia Makki, and Niki Pissinou
Florida International University, Miami, FL

***Abstract*—Recent development in VLSI technology makes it feasible to integrate on-chip memories to crossbar switching fabrics. Switches using such crossbars are called buffered crossbar switches, in which each crosspoint has a small exclusive buffer. The crosspoint buffers decouple input ports and output ports, and reduce the switch scheduling problem to the fair queueing problem. In this paper, we present the fair queueing based packet scheduling scheme for buffered crossbar switches, which requires no speedup and directly handles variable length packets without segmentation and reassembly (SAR). The presented scheme makes scheduling decisions in a distributed manner, and provides performance guarantees. Given the properties of the actual fair queueing algorithm used in the scheduling scheme, we calculate the crosspoint buffer size bound to avoid overflow, and analyze the fairness and delay guarantees provided by the scheduling scheme. In addition, we use WF^{2}Q, the fair queueing algorithm with the tightest performance guarantees, as a case study, and present simulation data to verify the analytical results.**

I. Introduction

Recent development in VLSI technology has made on-chip memories feasible for crossbar switching fabrics [1] [2]. Each crosspoint of the crossbar can thus have a small exclusive buffer. Crossbar switches with such crosspoint buffers are called buffered crossbar switches [3] [4]. They demonstrate significant advantages over traditional unbuffered crossbar switches [5] [6].

Unbuffered crossbar switches have no buffers on the crossbars, and packets have to be directly transmitted from input ports to output ports. They usually work with fixed length cells in a synchronous time slot mode [7]. To maximize throughput and accelerate scheduling, all the scheduling and transmission units must have the same length. Variable length packets are segmented into fixed length cells at input ports, and those cells will be reassembled into original packets at output ports. This process is called segmentation and reassembly (SAR) [8].

For buffered crossbar switches, crosspoint buffers decouple input ports and output ports, and simplify the scheduling process. They can directly handle variable length packets without SAR and work in an asynchronous mode. To be specific, input ports periodically send packets of arbitrary length to the corresponding 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 [5] [10], such as high throughput, low latency, and reduced hardware cost.

Existing scheduling schemes for buffered crossbar switches are designed with two alternative objectives: to provide performance guarantees, such as PLF [5] and sBUX [6], or to achieve high throughput, such as LIPS [10] and CIXB-1 [11]. The former is a stronger requirement than the latter. A scheme with tight performance guarantees usually delivers 100% throughput, but the reverse is not always true. For the existing schemes providing performance guarantees, they either consider cell scheduling or require speedup of two or more, i.e. the crossbar running faster than input ports and output ports. In contrast, the scheduling scheme that will be presented in this paper requires no speedup. Therefore, output ports need no buffers, and a packet arriving at the output port is directly sent to the output line.

Sending or retrieving packets to or from crosspoint buffers is similar to the fair queueing problem [12], in which a gateway transmits packets of different flows to ensure fairness. The fair queueing problem is a well studied area, and many algorithms have been proposed. They can be broadly classified into two categories: time stamp based algorithms, such as WFQ [12] and WF^{2}Q [13] [14], and round robin based algorithms, such as DRR [15] and FRR [16].

In this paper, we present the fair queueing based packet scheduling scheme for buffered crossbar switches to provide performance guarantees. We show that the $N \times N$ scheduling problem of buffered crossbar switches can be reduced to the $N \times 1$ scheduling problem, to which fair queueing algorithms apply. With the presented scheduling scheme, different input ports and output ports can make independent scheduling decisions based on only local information. Given the properties of the fair queueing algorithm used in the scheduling scheme, we calculate the crosspoint buffer size bound to avoid overflow, and analyze the fairness and delay guarantees provided by the scheduling scheme. Finally, we use WF^{2}Q, the fair queueing algorithm with the tightest performance guarantees, as a case study, and present simulation data to verify the analytical results.

The rest of the paper is organized as follows. In Section II, we present the fair queueing based packet scheduling scheme for buffered crossbar switches. In Section III, we theoretically analyze the performance of the presented scheme. In Section IV, we use WF^{2}Q as a case study and present simulation data. In Section V, we conclude the paper.

II. Fair Queueing Based Packet Scheduling for Buffered Crossbar Switches

In this section, we present the fair queueing based packet scheduling scheme for buffered crossbar switches, and discuss the fairness model that will be used in the rest of the paper.

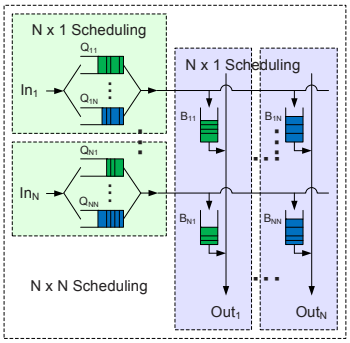

Fig. 1. Structure of buffered crossbar switches.

A. Scheduling Scheme Description

The considered switch structure 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) [18]. In other words, there are N virtual queues at an input port, each storing the packets destined to a different output port. 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. Define the traffic from In_i to Out_j to be a flow F_{ij}, and denote the k^{th} arriving packet of F_{ij} as P_{ij}^k. After P_{ij}^k arrives at the switch, it is first stored in Q_{ij}, and waits to be sent to B_{ij}. It will then be sent from B_{ij} to Out_j and immediately delivered to the output line. We say that a packet arrives at or departs from a buffer when its last bit arrives at or departs from the buffer.

Due to crosspoint buffers, the $N \times N$ switch scheduling problem can be reduced to two types of $N \times 1$ scheduling problems as illustrated in Figure 1, which we call input scheduling and output scheduling. In input scheduling, an input port selects a packet from one of its N input queues, and sends it to the corresponding crosspoint buffer. In output scheduling, an output port selects a packet from one of its N crosspoint buffers, and sends it to the output line. As can be seen, input scheduling and output scheduling rely on only local information, and can be conducted in a distributed manner.

We assume that each flow is allocated a specific amount of bandwidth, and the objective of the scheduling scheme is to ensure the allocated bandwidth for each flow and provide performance guarantees. It can be noted that input scheduling and output scheduling are similar to the fair queueing problem [19], in which a gateway is shared by N flows, and it schedules packets to ensure fairness among the flows. There are many fair queueing algorithms available in the literature, and they provide different performance guarantees with different computational complexity [17]. Depending on the applications that the buffered crossbar switch is used for, a particular fair queueing algorithm can be chosen for input scheduling and output scheduling.

B. Ideal Fairness Model

To effectively evaluate the fairness performance of a scheduling scheme, it is necessary to have an ideal fairness model as the comparison reference. Generalized Processor Sharing (GPS) [19] is a widely used fairness model for packet scheduling. When GPS applies to fair queueing, it calculates allocated bandwidth for flows in a proportional manner. It divides the gateway bandwidth into multiple logical channels. Each flow has its own channel, and the channel bandwidth is proportional to its requested bandwidth. 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.

However, simple proportional bandwidth allocation of GPS is not proper for switch scheduling [20] [21]. The reason is that, while flows of a gateway are constrained by only the gateway 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. Calculating bandwidth allocation for switches is an interesting problem, and some algorithms [20] [21] have been proposed to solve it. In this paper, we assume that bandwidth allocation has been calculated using such algorithms, and the scheduling scheme just needs to schedule packets to ensure the allocated bandwidth of each flow.

Given the calculated bandwidth allocation, GPS can divide the bandwidth of a switch into logical channels. Each flow has an independent channel, and the channel bandwidth is equal to the allocated bandwidth of the flow. When we use GPS as the ideal fairness model to evaluate a switch scheduling scheme, we compare the amount of service that a flow receives in the scheme and in GPS, under the same allocated bandwidth.

III. Performance Analysis

In this section, we analyze the performance of the fair queueing based packet scheduling scheme.

As discussed in Section II, we can adopt a fair queueing algorithm (FQA) for input scheduling and output scheduling to provide performance guarantees. To evaluate the performance of the scheduling scheme, we need to compare FQA with GPS, and we use the notation "A-B" to differentiate the algorithms used for input scheduling and output scheduling. A and B are the algorithms for input scheduling and output scheduling, respectively, and each can be either FQA or GPS. If we do not care the algorithm for output scheduling, we use a * mark for B. In the case of GPS-GPS, each flow has an independent channel, and its packets can stream like a fluid from the input port to the output port without buffering in the middle. Otherwise, if FQA is used for input scheduling or output scheduling, because of the store-and-forward nature [22] of practical networks, we assume that a packet has to be fully buffered at the crosspoint buffer before it is sent to the output port.

We make some general assumptions about the properties of FQA, which will be used to analyze the performance of the switch scheduling scheme. Use $A_{flow}(0,t)$ and $\widehat{A}_{flow}(0,t)$ to represent the amount of service that a flow receives during the

interval $[0,t]$ in FQA and GPS, respectively. We assume that FQA has bounded fairness guarantees.

Property 1: FQA has a fairness guarantee lower bound of A_{LB} and upper bound of A_{UB}, i.e.

$$A_{LB} \leq A_{flow}(0,t) - \widehat{A}_{flow}(0,t) \leq A_{UB} \tag{1}$$

Use F^k_{flow} and $\widehat{F}^k_{flow}$ to represent the service finish time of the k^{th} packet of a particular flow in FQA and GPS, respectively. We assume that FQA has bounded delay guarantees.

Property 2: FQA has a delay guarantee lower bound of A_{LB} and upper bound of A_{UB}, i.e.

$$F_{LB} \leq F^k_{flow} - \widehat{F}^k_{flow} \leq F_{UB} \tag{2}$$

The bandwidth allocation algorithms mentioned in Section II calculate for each flow F_{ij} its allocated bandwidth $r_{ij}(t)$, which is a function of time t with discrete values in practice. The calculated bandwidth allocation should be feasible, i.e. no over-subscription at any input port or output port

$$\forall i, \sum_j r_{ij}(t) \leq R, \text{ and } \forall j, \sum_i r_{ij}(t) \leq R \tag{3}$$

We first define some notations for input scheduling. Define the virtual input start time and finish time of P^k_{ij}, denoted as $\widehat{IS}^k_{ij}$ and $\widehat{IF}^k_{ij}$, to be the service start time and finish time of P^k_{ij} at In_i in GPS-*, respectively. In other words, if GPS is the input scheduler, $\widehat{IS}^k_{ij}$ and $\widehat{IF}^k_{ij}$ are the time that the first bit and last bit of P^k_{ij} leave Q_{ij}, respectively. According to the definition of GPS, $\widehat{IS}^k_{ij}$ can be calculated as follows

$$\widehat{IS}^k_{ij} = \max\left(IA^k_{ij}, \widehat{IF}^{k-1}_{ij}\right) \tag{4}$$

where IA^k_{ij} is the arrival time of P^k_{ij} at the input port. $\widehat{IF}^k_{ij}$ satisfies the following relationship

$$\int_{\widehat{IS}^k_{ij}}^{\widehat{IF}^k_{ij}} r_{ij}(x)dx = L^k_{ij} \tag{5}$$

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

$$\widehat{IF}^k_{ij} = \widehat{IS}^k_{ij} + \frac{L^k_{ij}}{r_{ij}} \tag{6}$$

We say that Q_{ij} is backlogged at time t, if there exists k such that $\widehat{IS}^k_{ij} \leq t \leq \widehat{IF}^k_{ij}$. 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.

Correspondingly, define the actual input start time and finish time of P^k_{ij}, denoted as IS^k_{ij} and IF^k_{ij}, to be the time that the first bit and last bit of P^k_{ij} leave Q_{ij} in FQA-*. Apparently

$$IF^k_{ij} = IS^k_{ij} + \frac{L^k_{ij}}{R} \tag{7}$$

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 FQA-* and GPS-*, respectively. Based on the definition of GPS, $\widehat{toB}_{ij}(t_1,t_2)$ can be calculated as

$$\widehat{toB}_{ij}(t_1,t_2) = \int_{t_1}^{t_2} r_{ij}(x)\hat{q}_{ij}(x)dx \tag{8}$$

Next, we define the notations for output scheduling. Define the virtual output start time and finish time of P^k_{ij}, denoted as $\widehat{OS}^k_{ij}$ and $\widehat{OF}^k_{ij}$, to be the time that the first bit and last bit of P^k_{ij} leave B_{ij} in FQA-GPS, respectively. In other words, after P^k_{ij} is delivered to B_{ij} in FQA, if GPS is the output scheduler, P^k_{ij} will start transmission at $\widehat{OS}^k_{ij}$ and finish at $\widehat{OF}^k_{ij}$. $\widehat{OS}^k_{ij}$ is calculated as

$$\widehat{OS}^k_{ij} = \max\left(OA^k_{ij}, \widehat{OF}^{k-1}_{ij}\right) \tag{9}$$

where OA^k_{ij} is the arrival time of P^k_{ij} at B_{ij} in FQA-*, and is equal to IF^k_{ij} by neglecting propagation delay. $\widehat{OF}^k_{ij}$ satisfies the following relationship

$$\int_{\widehat{OS}^k_{ij}}^{\widehat{OF}^k_{ij}} r_{ij}(x)dx = L^k_{ij} \tag{10}$$

We say that B_{ij} is backlogged at time t, if there exists k such that $\widehat{OS}^k_{ij} \leq t \leq \widehat{OF}^k_{ij}$. 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.

Define the actual output start time and finish time of P^k_{ij}, denoted as OS^k_{ij} and OF^k_{ij}, to be the time that the first bit and the last bit of P^k_{ij} leave B_{ij} in FQA-FQA, respectively. It is obvious that

$$OF^k_{ij} = OS^k_{ij} + \frac{L^k_{ij}}{R} \tag{11}$$

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 FQA-FQA and FQA-GPS, respectively. $\widehat{toO}_{ij}(t_1,t_2)$ is calculated as

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

A. Crosspoint Buffer Size Bound

To avoid buffer overflow at crosspoints, we would like to find the maximum number of bits that may be buffered at any crosspoint.

The next two lemmas compare the number of bits transmitted by the same flow in the input scheduling of GPS-* and the output scheduling of FQA-GPS.

Lemma 1: During interval $[0,t]$, the number of bits transmitted by flow F_{ij} from crosspoint buffer B_{ij} to output port Out_j in FQA-GPS is less than or equal to that from input port In_i to B_{ij} in GPS-* plus A_{UB}, i.e.

$$\widehat{toO}_{ij}(0,t) \leq \widehat{toB}_{ij}(0,t) + A_{UB} \tag{13}$$

Proof: By Property 1, we have $toB_{ij}(0,t) \leq \widehat{toB}_{ij}(0,t) + A_{UB}$. In addition, because every packet sent from B_{ij} to Out_j in the output scheduling of FQA-GPS has to be first sent from In_i to B_{ij} in the input scheduling, we can obtain

$$\widehat{toO}_{ij}(0,t) \le toB_{ij}(0,t) \le \widehat{toB}_{ij}(0,t) + A_{UB} \quad (14)$$

■

Lemma 2: During interval $[0,t]$, the number of bits transmitted by flow F_{ij} from crosspoint buffer B_{ij} to output port Out_j in FQA-GPS is greater than or equal to that from input port In_i to B_{ij} in GPS-* plus $A_{LB} - L$, where L is the maximum packet length, i.e.

$$\widehat{toO}_{ij}(0,t) \ge \widehat{toB}_{ij}(0,t) + A_{LB} - L \quad (15)$$

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

By Property 1, we have $toB_{ij}(0,s) \ge \widehat{toB}_{ij}(0,s) + A_{LB}$. Because B_{ij} is empty before s and backlogged after s in FQA-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

$$\begin{aligned} \widehat{toO}_{ij}(0,s) &\ge toB_{ij}(0,s) - L \\ &\ge \widehat{toB}_{ij}(0,s) + A_{LB} - L \end{aligned} \quad (16)$$

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

$$\begin{aligned} \widehat{toO}_{ij}(s,t) &= \int_s^t r_{ij}(x)\hat{b}_{ij}(x)dx \\ &= \int_s^t r_{ij}(x)dx \\ &\ge \int_s^t r_{ij}(x)\hat{q}_{ij}(x)dx \\ &= \widehat{toB}_{ij}(s,t) \end{aligned} \quad (17)$$

Adding (16) and (17), we have proved the lemma. ■

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

Theorem 1: In FQA-FQA, the maximum number of bits buffered at a crosspoint buffer is bounded by $L+A_{UB}-2A_{LB}$, i.e.

$$toB_{ij}(0,t) - toO_{ij}(0,t) \le L + A_{UB} - 2A_{LB} \quad (18)$$

Proof: We can obtain by Property 1

$$toB_{ij}(0,t) \le \widehat{toB}_{ij}(0,t) + A_{UB} \quad (19)$$

$$\widehat{toO}_{ij}(0,t) \le toO_{ij}(0,t) - A_{LB} \quad (20)$$

and by Lemma 2

$$\widehat{toB}_{ij}(0,t) \le \widehat{toO}_{ij}(0,t) + L - A_{LB} \quad (21)$$

Summing (19), (20), and (21), we have proved the theorem. ■

B. Performance Guarantees

In this subsection, we show that FQA-FQA provides bounded fairness and delay guarantees.

Use $\widetilde{toO}_{ij}(t_1,t_2)$ to denote the number of bits transmitted by flow F_{ij} to the output port during interval $[t_1,t_2]$ in GPS-GPS. By neglecting propagation delay, we can obtain $\widetilde{toO}_{ij}(0,t) = \widehat{toB}_{ij}(0,t)$. The next theorem gives the fairness guarantees of FQA-FQA.

Theorem 2: At any time, the difference between the numbers of bits transmitted by a flow to the output port in FQA-FQA and GPS-GPS is greater than or equal to $2A_{LB} - L$ and less than or equal to $2A_{UB}$, i.e.

$$2A_{LB} - L \le toO_{ij}(0,t) - \widetilde{toO}_{ij}(0,t) \le 2A_{UB} \quad (22)$$

Proof: First, we prove $toO_{ij}(0,t) - \widetilde{toO}_{ij}(0,t) \ge 2A_{LB} - L$. By Lemma 2, we have $\widehat{toO}_{ij}(0,t) \ge \widehat{toB}_{ij}(0,t) + A_{LB} - L$. By Property 1, we can obtain

$$\begin{aligned} toO_{ij}(0,t) &\ge \widehat{toO}_{ij}(0,t) + A_{LB} \\ &\ge \widehat{toB}_{ij}(0,t) + 2A_{LB} - L \\ &= \widetilde{toO}_{ij}(0,t) + 2A_{LB} - L \end{aligned} \quad (23)$$

Next, we prove $toO_{ij}(0,t) - \widetilde{toO}_{ij}(0,t) \le 2A_{UB}$. According to Lemma 1, $\widehat{toO}_{ij}(0,t) \le \widehat{toB}_{ij}(0,t) + A_{UB}$. By Property 1, we have

$$\begin{aligned} toO_{ij}(0,t) &\le \widehat{toO}_{ij}(0,t) + A_{UB} \\ &\le \widehat{toB}_{ij}(0,t) + 2A_{UB} \\ &= \widetilde{toO}_{ij}(0,t) + 2A_{UB} \end{aligned} \quad (24)$$

■

Now we look at the delay guarantees of FQA-FQA. Because packet transmission time depends on the allocated bandwidth of the flow, 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 represent the departure time of P_{ij}^k leaving Out_j 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 FQA-FQA, if the propagation delay is neglected. The next theorem gives the delay guarantees of FQA-FQA.

Theorem 3: The difference between the departure time of packet P_{ij}^k in FQA-FQA and GPS-GPS is greater than or equal to $F_{LB} + \frac{L_{ij}^k}{R}$ and less than or equal to $2F_{UB} + \frac{L+A_{UB}-A_{LB}}{r_{ij}}$, i.e.

$$F_{LB} \le OF_{ij}^k - \widetilde{OF}_{ij}^k \le 2F_{UB} + \frac{L + A_{UB} - A_{LB}}{r_{ij}} \quad (25)$$

Proof: First, we prove $OF_{ij}^k - \widetilde{OF}_{ij}^k \ge F_{LB}$. It is obvious that $OF_{ij}^k \ge OA_{ij}^k + \frac{L_{ij}^k}{R}$. To obtain a universal delay guarantee lower bound, we remove $\frac{L_{ij}^k}{R}$ from the right side, because it depends on the length of a particular packet. By Property 2, we know $IF_{ij}^k \ge \widehat{IF}_{ij}^k + F_{LB}$, and thus

$$OF_{ij}^k \ge OA_{ij}^k = IF_{ij}^k \ge \widehat{IF}_{ij}^k + F_{LB} = \widetilde{OF}_{ij}^k + F_{LB} \quad (26)$$

Next, we prove $OF_{ij}^k - \widetilde{OF}_{ij}^k \le 2F_{UB} + \frac{L+A_{UB}-A_{LB}}{r_{ij}}$. By Lemma 2, we have $\widehat{toB}_{ij}(0,t) - \widehat{toO}_{ij}(0,t) \le L - A_{LB}$, and by Property 1, we have $toB_{ij}(0,t) - \widehat{toB}_{ij}(0,t) \le A_{UB}$. Combining them, we obtain $toB_{ij}(0,t) - \widehat{toO}_{ij}(0,t) \le L + A_{UB} - A_{LB}$. This indicates that, after P_{ij}^k arrives at B_{ij} in FQA-GPS, the maximum queue length at B_{ij} is $L + A_{UB} - A_{LB}$. Because B_{ij} is served with fixed allocated bandwidth r_{ij} in FQA-GPS, we have

$$\widehat{OF}_{ij}^{k} \leq OA_{ij}^{k} + \frac{L + A_{UB} - A_{LB}}{r_{ij}}$$
$$= IF_{ij}^{k} + \frac{L + A_{UB} - A_{LB}}{r_{ij}} \quad (27)$$

By Property 2, we know

$$OF_{ij}^{k} \leq \widehat{OF}_{ij}^{k} + F_{UB} \quad (28)$$
$$IF_{ij}^{k} \leq \widehat{IF}_{ij}^{k} + F_{UB} \quad (29)$$

By (27), (28), and (29), we obtain

$$OF_{ij}^{k} \leq \widehat{OF}_{ij}^{k} + F_{UB}$$
$$\leq IF_{ij}^{k} + F_{UB} + \frac{L + A_{UB} - A_{LB}}{r_{ij}}$$
$$\leq \widehat{IF}_{ij}^{k} + 2F_{UB} + \frac{L + A_{UB} - A_{LB}}{r_{ij}}$$
$$= \widehat{OF}_{ij}^{k} + 2F_{UB} + \frac{L + A_{UB} - A_{LB}}{r_{ij}} \quad (30)$$

■

IV. Case Study: WF^{2}Q

In this section, we use WF^{2}Q as a case study for fair queueing based packet scheduling, and present simulation data to verify the analytical results obtained in Section III.

WF^{2}Q is a time stamp based fair queueing algorithm, and provides the tightest performance guarantees among the existing fair queueing algorithms [14]. It has a fairness guarantee lower bound $A_{LB} = -L$ and upper bound $A_{UB} = L$ [13]. It has a delay guarantee upper bound of $F_{UB} = \frac{L}{R}$ [13]. By the analytical results in Section III, we can easily know the performance of WF^{2}Q-WF^{2}Q.

Next, we verify the performance of WF^{2}Q-WF^{2}Q by simulation. In the simulations, we consider a 16×16 buffered crossbar switch without speedup. Each input port and output port has bandwidth of 1G bps. Packet length is uniformly distributed between 40 and 1500 bytes [9]. For bandwidth allocation, we use a similar model as that in [4] and [11]. The allocated bandwidth $r_{ij}(t)$ of flow F_{ij} at time t is defined by an unbalanced probability w as follows

$$r_{ij}(t) = \begin{cases} R\left(w + \frac{1-w}{N}\right), & \text{if } i = j \\ R\frac{1-w}{N}, & \text{if } i \neq j \end{cases} \quad (31)$$

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.

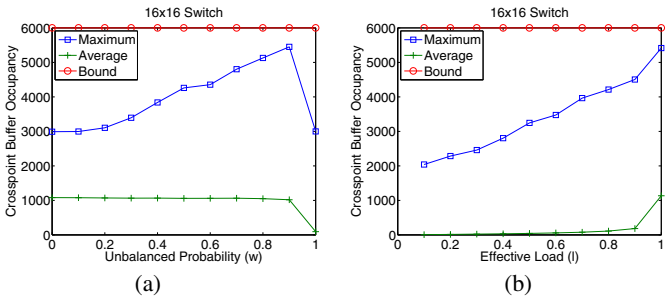

Fig. 2. Crosspoint buffer occupancy of WF^{2}Q-WF^{2}Q. (a) With different unbalanced probabilities. (b) With different loads.

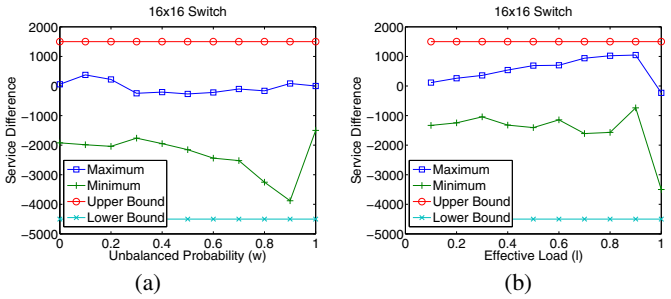

Fig. 3. Fairness guarantees of WF^{2}Q-WF^{2}Q. (a) With different unbalanced probabilities. (b) With different loads.

A. Crosspoint Buffer Size

By Theorem 1, we know that WF^{2}Q-WF^{2}Q has a crosspoint buffer size bound of $4L$. In this subsection, we look at the maximum and average crosspoint buffer occupancies of WF^{2}Q-WF^{2}Q in the simulations.

Figure 2(a) shows the maximum and average crosspoint buffer occupancies under traffic pattern one. As can be seen, 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 one. The reason is that when the unbalanced probability is one, all packets of In_i go to Out_i, and thus no switching is necessary. For the average occupancy, it does not change significantly with different unbalanced probabilities, and drops when the unbalanced probability becomes one for the same reason. Figure 2(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 increases to one.

B. Fairness Guarantees

By Theorem 2, we know that WF^{2}Q-WF^{2}Q has a fairness guarantee lower bound of $-3L$ and upper bound of $2L$. Next, we look at the simulation data on fairness guarantees.

Define the service difference to be the difference between the numbers of bits transmitted by a flow in WF^{2}Q-WF^{2}Q and GPS-GPS. Figure 3(a) shows the maximum and minimum service differences among all the flows under traffic pattern one. We can see that the maximum service difference is always smaller than the upper bound and close to zero. Furthermore, it

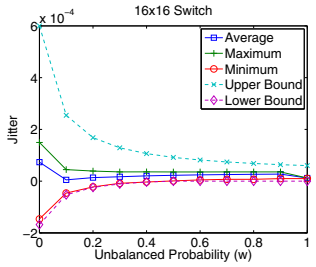

Fig. 4. Jitter of WF^{2}Q-WF^{2}Q with different unbalanced probabilities.

is sometimes negative. This indicates that every flow transmits a smaller number of bits in WF^{2}Q-WF^{2}Q than in GPS-GPS, which is reasonable. On the other hand, the minimum service difference is always greater than the lower bound. It drops when the unbalanced probability increases but suddenly jumps when the unbalanced probability becomes one. Figure 3(b) shows the maximum and minimum service differences under traffic pattern two. Similarly, the maximum service difference is smaller than the upper bound and close to zero. The minimum service difference is greater than the lower bound, and keeps relatively constant but drops when the load becomes one.

C. Delay Guarantees

By Theorem 3, we know that WF^{2}Q-WF^{2}Q has a delay guarantee lower bound of $-\frac{L}{r_{ij}}$ and upper bound of $\frac{2L}{R}+\frac{3L}{r_{ij}}$. Because Theorem 3 assumes fixed allocated bandwidth r_{ij}, we use only traffic pattern one for this part of simulations.

Define the jitter to be the difference between the packet departure time in WF^{2}Q-WF^{2}Q and GPS-GPS. Figure 4 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, which means that WF^{2}Q-WF^{2}Q and GPS-GPS have similar average packet delay.

V. Conclusions

The introduction of crosspoint buffers enables buffered crossbar switches to simplify the scheduling process. In this paper, we have presented the fair queueing based packet scheduling scheme for buffered crossbar switches. The main features of the presented scheme can be summarized as follows. First, unlike specific scheduling algorithms, the presented scheme is a general framework that can work with different fair queueing algorithms to provide different performance guarantees. Second, the presented scheme schedules variable length packets without segmentation-and-reassembly (SAR), avoiding padding bits and improving the switch throughput. Third, while most existing scheduling algorithms for buffered crossbar switches require speedup of two or more, the presented scheme needs no speedup, and thus reduces the hardware cost. Fourth, the presented scheme enables distributed scheduling, because each input port and output port needs only local information to make scheduling decisions.

By theoretically analysis, we show that the presented scheme has a bounded crosspoint buffer size, independent of the switch size. In addition, we prove that it provides bounded performance guarantees. Finally, we use WF^{2}Q as a case study, and present simulation data to verify the analytical results.

References

[1] L. Mhamdi, C. Kachris, and S. Vassiliadis, "A reconfigurable hardware based embedded scheduler for buffered crossbar switches," *14th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 143-149, Monterey, CA, Feb. 2006.

[2] 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.

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

[4] 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, Sep. 2003

[5] J. Turner, "Strong performance guarantees for asynchronous crossbar schedulers," *IEEE/ACM Transactions on Networking*, to appear, 2009.

[6] S. He et al., "On Guaranteed Smooth Switching for Buffered Crossbar Switches," *IEEE/ACM Transactions on Networking* vol. 16, no. 3, pp. 718-731, Jun. 2008.

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

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

[9] C. Farleigh et al., "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, no. 6, pp. 6C16, Nov. 2003.

[10] D. Pan and Y. Yang, "Localized independent packet scheduling for buffered crossbar switches," *IEEE Transactions on Computers*, vol. 58, no. 2, pp. 260-274, Feb. 2009.

[11] R. Rojas-Cessa, E. Oki, Z. Jing, and H. Chao, "CIXB-1: Combined input-once-cell-crosspoint buffered switch," *IEEE HPSR 2001*, Dallas, TX, Jul. 2001.

[12] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM 1989*, Austin, TX, Sep. 1989.

[13] J. Bennett and H. Zhang, "WF2Q: worst-case fair weighted fair queueing," *IEEE INFOCOM 1996*, San Francisco, CA, Mar. 1996.

[14] P. Valente, "Exact GPS simulation with logarithmic complexity, and its application to an optimally fair scheduler," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1454-1466, Dec. 2007.

[15] 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.

[16] X. Yuan and Z. Duan, "Fair round robin: A low complexity packet scheduler with proportional and worst-case fairness," *IEEE Transactions on Computers*, vol. 58, no. 3, pp. 365-379, Mar. 2009.

[17] 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.

[18] 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, Aug. 1999.

[19] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 344-357, Jun. 1993.

[20] D. Pan and Y. Yang, "Max-min fair bandwidth allocation algorithms for packet switches," *IEEE IPDPS 2007*, Long Beach, CA, Mar. 2007.

[21] M. Hosaagrahara and H. Sethu, "Max-min fairness in input-queued switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 4, pp. 462-475, Apr. 2008.

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