Achieve Constant Performance Guarantees using Asynchronous Crossbar Scheduling without Speedup

Deng Pan
Florida International University
Miami, FL

Kia Makki
TUM
Miami, FL

Niki Pissinou
Florida International University
Miami, FL

***Abstract*—Buffered crossbar switches are special crossbar switches with a small exclusive buffer at each crosspoint of the crossbar. They demonstrate unique advantages, such as variable length packet handling and distributed scheduling, over traditional unbuffered crossbar switches. The current main approach for buffered crossbar switches to provide performance guarantees is to emulate push-in-first-out output queued switches. However, such an approach has several drawbacks, and in particular it has difficulty in providing tight constant performance guarantees. To address the issue, we propose in this paper the guaranteed-performance asynchronous packet scheduling (GAPS) algorithm for buffered crossbar switches. GAPS intends to provide tight performance guarantees, and requires no speedup. It directly handles variable length packets without segmentation and reassembly, and makes scheduling decisions in a distributed manner. We show by theoretical analysis that GAPS achieves constant performance guarantees. We also prove that GAPS has a bounded crosspoint buffer size of $3L$, where L is the maximum packet length. Finally, we present simulation data to verify the analytical results and show the effectiveness of GAPS.**

***Keywords*-buffered crossbar switches; performance guarantees; speedup;**

I. Introduction

Buffered crossbar switches have recently attracted considerable attentions [1] - [16] as promising high speed interconnects. They are special crossbar switches with a small exclusive buffer at each crosspoint of the crossbar, as shown in Figure 1. Such a switch architecture was once regarded as not scalable [17]. Fortunately, recent development in VLSI technology has made it feasible to integrate on-chip memories to crossbar switching fabrics, and thus build moderate-size buffered crossbar switches [1] - [4]. Buffered crossbar switches demonstrate unique advantages over traditional unbuffered crossbar switches [5] - [7] .

Unbuffered crossbar switches have no buffers on the crossbar, 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 [18]. To maximize throughput and accelerate scheduling, all the scheduling and transmission units must have the same length. In each time slot, all input-output pairs transmit cells at the same time. When variable length packets arrive, they will be segmented into fixed length cells at input ports. The cells are then 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) [14].

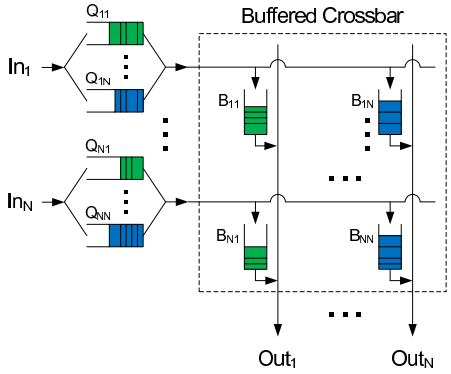


Figure 1. Structure of buffered crossbar switches.

For buffered crossbar switches, crosspoint buffers decouple input ports and output ports, and simplify the scheduling process [8]. They can directly handle variable length packets without SAR and work in an asynchronous mode [7]. 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. Note that packets in most practical networks are of variable length [19]. Compared with fixed length cell scheduling of unbuffered crossbar switches, variable length packet scheduling of buffered crossbar switches has some unique advantages [7] [8], such as high throughput, low latency, and reduced hardware cost.

In this paper, we study fair scheduling algorithms for buffered crossbar switches to provide performance guarantees. The considered problem is that each flow of the switch is allocated a specific amount of bandwidth, and the fair scheduling algorithm arranges packet transmission to ensure that the flow receives its allocated bandwidth, and thus provides guaranteed delay and jitter. There exist a number of solutions [5] - [12] in the literature, and the main approach is to emulate push-in-first-out (PIFO)

output queued (OQ) switches [5], i.e. duplicating the packet departure time in PIFO OQ switches. As indicated by the name, OQ switches have buffer space only at output ports. Because input ports have no buffers, all arriving packets have to be immediately transferred to the output buffers by the crossbar. Thus, the crossbar of an $N \times N$ OQ switch needs to run N times faster than the input port and output port, or in other words has speedup of N [20]. The speedup requirement makes OQ switches difficult to scale. On the other hand, because all packets are already stored at the output ports, it is easy for OQ switches to run various fair queueing algorithms, such as Deficit Round Robin (DRR) [21], Weighted Fair Queueing (WFQ) [22], and Worst-case Fair Weighted Fair Queueing (WF^{2}Q) [23], to provide performance guarantees. The objective is for each output port to emulate the packet departure sequence in the ideal Generalized Processor Sharing (GPS) [24] fairness model. Specifically, a PIFO OQ switch is an OQ switch with a push-in-first-out queueing policy. For such a switch, an arriving packet can be put anywhere in the output queue and a departing packet can only be removed from the head of the queue [5].

However, the above emulation approach has several drawbacks. In particular, it has difficulty in providing constant performance guarantees. Constant performance guarantees mean that for any flow, the difference between its received bandwidth in a specific algorithm and in the ideal GPS model is bounded by constants, i.e. the equations in Theorem 1 of [23]. They are the key properties to assure worst-case fairness [23]. The reason is that WF^{2}Q (including its variants) [23] is the only known fair queueing algorithm to achieve constant performance guarantees. Unfortunately, WF^{2}Q does not use a PIFO queueing policy [25], because a packet at the head of a queue may not be eligible for departure because it has not started transmission in GPS [23].

In this paper, we propose the Guaranteed-performance Asynchronous Packet Scheduling (GAPS) algorithm for buffered crossbar switches to provide constant performance guarantees. The considered buffered crossbar switches do not need speedup. Because the crossbar runs at the same speed as the output ports, no buffer space is necessary at the output ports. When a packet is transmitted to its output port, it will be immediately sent to the output line. GAPS uses time stamps of packets in GPS as the scheduling criteria, and perfectly emulates the ideal GPS model. It directly handles variable length packets without SAR, and allows input ports and output ports to make independent scheduling decisions based on only local information without data exchange. Specifically, an input port needs only the statuses of its input queues, and an output port needs only the statuses of its crosspoint buffers. We show by theoretical analysis that GAPS provides constant performance guarantees. Furthermore, we prove that GAPS has a crosspoint buffer size bound of $3L$, where L is the maximum packet length. Finally, we conduct simulations to verify the analytical results and evaluate the effectiveness of GAPS.

The rest of the paper is organized as follows. In Section II, we introduce some preliminaries for the paper. In Section III, we propose the GAPS algorithm. In Section IV, we theoretically analyze the performance of GAPS. In Section V, we present simulation data. In Section VI, we conclude the paper.

II. Preliminaries

In this section, we first provide an overview of the approach to provide performance guarantees by emulating PIFO OQ switches. We then analyze in detail the drawbacks of the emulation approach, and describe the ideal fairness model used in this paper.

A. Emulating PIFO OQ Switches

The emulation of PIFO OQ switches is the current main approach in the literature for crossbar switches to provide performance guarantees. It was proved in [10] that a buffered crossbar switch with speedup of two satisfying non-negative slackness insertion and lowest time to live (LTTL) blocking, and LTTL fabric scheduling can exactly emulate a PIFO OQ switch. In [11], the MCAF-LTF cell scheduling scheme for one-cell buffered crossbar switches was proposed. MCAF-LTF does not require costly time stamping mechanism, and is able to emulate an PIFO OQ switch with speedup of two. [5] studied practical scheduling algorithms for buffered crossbar switches. It showed that with speedup of two, a buffered crossbar switch can mimic a restricted PIFO OQ switch (a PIFO-OQ switch with the restriction that the cells of an input-output pair depart the switch in the same order as they arrive), and that with speedup of three, a buffered crossbar switch can mimic an arbitrary PIFO OQ switch and hence provide delay guarantees. [12] presented a cell scheduling algorithm for buffered crossbar switches with speedup of two to emulate an arbitrary PIFO OQ switch and achieve flow based performance guarantees. The performance guarantees of packet scheduling for asynchronous buffer crossbar switches were discussed in [7]. The Packet GVOQ and Packet LOOFA scheduling algorithms were designed based on existing cell scheduling algorithms. They require $2L$ or more buffer space at each crosspoint. Besides buffered crossbar switches, Combined-Input-Output-Queued switches are also proved to be able to emulate PIFO OQ switches with speedup of two [26] [27].

The above algorithms were designed to make exact emulation of PIFO OQ switches. There are also some other schemes that intend to emulate OQ switches but cannot duplicate the same packet departure sequence. [28] proposed the Distributed Packet Fair Queueing architecture for physically dispersed line cards to emulate an OQ switch with fair queueing, and used simulation results to demonstrate its

effectiveness with modest speedup. iFS was proposed in [29] for virtual output queued (VOQ) switches to emulate WFQ [22] at each output port. iFS uses a grant-accept two stage iterative matching method, and uses the virtual time as the grant criterion. Similarly, iDRR in [30] emulates DRR [21] at each output port of VOQ switches. iDRR uses the round robin principle in its iterative matching steps, and thus is able to make fast arbitration.

B. Drawbacks of Emulation Approach

There are two main drawbacks with the above approach to provide performance guarantees by emulating PIFO OQ switches. First, as discussed in Section I, it has difficulty in providing tight performance guarantees. Second, the proportional bandwidth allocation policy of PIFO OQ switches is not practical, because it does not consider the bandwidth constraints at the input ports, while flows may oversubscribe input ports [16].

The objective of the emulation approach is to emulate a fair queueing algorithm at each output port. Fair queueing algorithms schedule packets from multiple flows of a shared output link to ensure fair bandwidth allocation, and they allocate bandwidth to the flows proportional to their requested bandwidth [24]. Numerically, assume that the available bandwidth of the shared output link is R, and ϕ_i and R_i are the requested bandwidth and allocated bandwidth of the i^{th} flow, respectively. With proportional bandwidth allocation, we have $\forall i, \forall j, \frac{R_i}{\phi_i} = \frac{R_j}{\phi_j}$ and $\sum_i R_i \leq R$. However, simple proportional bandwidth allocation is not suitable for switches [31] [32]. 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.

In the following, we use an example to illustrate the issue. Consider a 2×2 switch. For easy representation, denote the i^{th} input port as In_i and the j^{th} output port as Out_j. Assume that each input port or output port has available bandwidth of one unit. Use ϕ_{ij} and R_{ij} to represent the requested bandwidth and allocated bandwidth of In_i at Out_j, respectively. Assume that each output port uses the proportional bandwidth allocation policy, i.e. the policy used by fair queueing algorithms for shared output links. First we look at only Out_1. Because $\phi_{11} = 0.9$ and $\phi_{21} = 0.6$, by the proportional policy we have $R_{11} = 0.6$ and $R_{21} = 0.4$. The same applies to Out_2. The allocated bandwidth R_{ij} is thus shown in (1). However, this allocation is not feasible, because the total bandwidth allocated at In_1 is $R_{11} + R_{12} = 0.6 + 0.6 = 1.2$, exceeding the available bandwidth of 1. For the same reason, if bandwidth allocation is conducted independently by each input port using the proportional policy, the allocation will not be feasible either.

$$\phi = \begin{bmatrix} 0.9 & 0.75 \\ 0.6 & 0.5 \end{bmatrix} \Rightarrow R = \begin{bmatrix} \mathbf{0.6} & \mathbf{0.6} \\ 0.4 & 0.4 \end{bmatrix} \quad (1)$$

In addition, to improve utilization, fair queueing algorithms will reallocate the leftover bandwidth of empty flows using the proportional policy. In other words, when a flow temporarily becomes empty, the fair queueing algorithm will reallocate its bandwidth to the remaining backlogged flows in proportion to their requested bandwidth. However, this strategy does not apply to switches either, and we use an additional example to explain. Consider the same 2×2 switch, and assume that initially $\forall i \forall j, R_{ij} = 0.5$, as shown in (2). Now that In_1 temporarily has no traffic to Out_1, i.e. $R_{11} = 0.5$ changing to $R'_{11} = 0$. The fair bandwidth allocation policy would allocate the leftover bandwidth of R_{11} to R_{21}, because now only In_2 has traffic to Out_1. However, it is not possible here, because it will oversubscribe In_2 by 0.5. As a matter of fact, the leftover bandwidth of R_{11} cannot be reallocated at all in this case.

$$R = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \Rightarrow R' = \begin{bmatrix} 0 & 0.5 \\ \mathbf{1} & \mathbf{0.5} \end{bmatrix} \quad (2)$$

C. Our Fairness Model

To effectively evaluate the performance guarantees achieved by 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 allocated bandwidth for flows based on their requested bandwidth. The second role is to schedule packets of different flows to ensure that the actual received bandwidth of each flow is equal to its allocated bandwidth.

As we have seen in Section II-B, the simple proportional bandwidth allocation policy does not apply to switches. Fortunately, there have been some solutions in the literature [31] [32] to fairly allocate bandwidth for flows in a switch based on their requested bandwidth. In this paper, we focus on addressing the first drawback of the emulation approach. In other words, we assume that bandwidth allocation has been calculated by such algorithms, and the scheduling algorithms should provide tight performance guarantees 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.

We use GPS as the ideal model for packet scheduling. Specifically, given the allocated bandwidth, we compare the received service of a flow in our algorithm and in GPS. GPS views flows as fluids of continuous bits, and creates an independent logical channel for each flow based on its

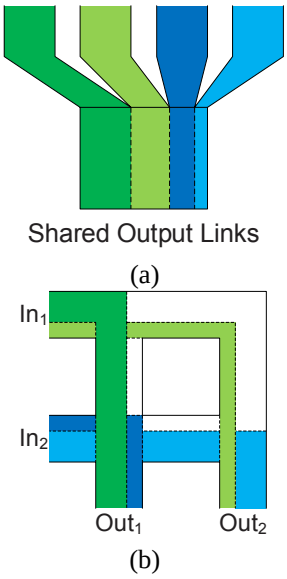


Figure 2. GPS as the ideal fairness model. (a) For shared output links. (b) For switches.

allocated bandwidth. Since the channel bandwidth of a flow is equal to its allocated bandwidth, GPS achieves perfect fairness. Fair queueing algorithms for shared output links also use GPS as the ideal model for packet scheduling, as shown in Figure 2(a). Similarly, GPS can apply to switch packet scheduling by creating logical channels for different flows based on their allocated bandwidth, as shown in Figure 2(b). Note that because GPS is a fluid based system, traffic of a flow can smoothly stream from the input port to the output port without buffering in the middle. We thus assume that packets in GPS do not need to be buffered at the crosspoint buffers of the switch.

III. Guaranteed-performance Asynchronous Packet Scheduling

In this section, we describe the considered switch structure, formulate the problem, and present the guaranteed-performance asynchronous packet scheduling (GAPS) algorithm.

A. Switch Structure

The switch structure that we consider is shown in Figure 1. N input ports and N output ports are connected by a buffered crossbar without 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) [33]. 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.

B. Problem Formulation

As explained in Section II-C, specific bandwidth allocation algorithms will calculate explicit allocated bandwidth for each flow, and the objective of GAPS is to provide service guarantees for each flow.

Use $R_{ij}(t)$ to represent the allocated bandwidth of F_{ij}, 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

$$\forall i, \sum_{x=1}^{N} R_{ix}(t) \leq R, \text{ and } \forall j, \sum_{x=1}^{N} R_{xj}(t) \leq R \qquad (3)$$

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, temporary overload is allowed for an input port or output port.

Use $toO_{ij}(0,t)$ and $\widehat{toO}_{ij}(0,t)$ to represent the numbers of bits transmitted by F_{ij} to Out_j during interval $[0,t]$ in GAPS and in GPS, respectively. The objective of GAPS is to ensure that $toO_{ij}(0,t) - \widehat{toO}_{ij}(0,t)$ is bounded by constants.

C. Algorithm Description

GAPS uses time stamps as scheduling criteria. There are two types of time stamps. The first time stamp for P_{ij}^k is called virtual start time, denoted as $\widehat{VS}_{ij}^k$, which is the service start time of P_{ij}^k in GPS. The second time stamp is virtual finish time, denoted as $\widehat{VF}_{ij}^k$, which is the service finish time of P_{ij}^k in GPS. In other words, if the switch uses GPS to schedule packets, $\widehat{VS}_{ij}^k$ and $\widehat{VF}_{ij}^k$ are the time that the first bit and last bit of P_{ij}^k depart from the switch, respectively. $\widehat{VS}_{ij}^k$ can be calculated as follows

$$\widehat{VS}_{ij}^k = \max\left(IA_{ij}^k, \widehat{VF}_{ij}^{k-1}\right) \qquad (4)$$

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

$$\int_{\widehat{VS}_{ij}^k}^{\widehat{VF}_{ij}^k} R_{ij}(x)dx = L_{ij}^k \qquad (5)$$

where L_{ij}^k is the length of P_{ij}^k. Because $R_{ij}(x)$ has only discrete values in practice, $\widehat{VF}_{ij}^k$ can be easily calculated. For

Table 1
PSEUDO CODE DESCRIPTION OF GAPS

Input Scheduling:
for In_i do {
  while true do {
    if there are packets in input queues with virtual start time less than or equal to current system time {
      select among such packets the one with the smallest virtual 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 start time;
    }
  }
}

Output Scheduling:
for Out_j do {
  while true do {
    if there are packets in crosspoint buffers with virtual start time less than or equal to current system time minus $\frac{L}{R}$ {
      select among such packets the one with the smallest virtual 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 start time plus $\frac{L}{R}$;
    }
  }
}

example, if $R_{ij}(t)$ is a constant R_{ij} during $\left[\widehat{VS}_{ij}^k, \widehat{VF}_{ij}^k\right]$, $\widehat{VF}_{ij}^k$ can be calculated as

$$\widehat{VF}_{ij}^k = \widehat{VS}_{ij}^k + \frac{L_{ij}^k}{R_{ij}} \tag{6}$$

There are two types of scheduling in GAPS, 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.

Input scheduling of each input port In_i is independent, and In_i only needs to checks its input queues. First, In_i identifies eligible packets. A packet P_{ij}^k is eligible for input scheduling if its virtual start time $\widehat{VS}_{ij}^k$ is less than or equal to the current system time t. In other words, a packet that has started transmission in GPS is eligible for input scheduling. Next, In_i selects an eligible packet. If there exist multiple eligible packets in the input buffer, In_i will select among such packets the one P_{ij}^k with the smallest virtual finish time $\widehat{VF}_{ij}^k$, and send it to B_{ij}. If there are no eligible packets, In_i will wait until the next earliest virtual start time of a packet. Note that when In_i is waiting for an eligible packet, if an empty input queue has a new arriving packet, whose virtual start time is equal to its arrival time, In_i should immediately start transmitting this new packet.

Output scheduling of GAPS is similar to input scheduling with different eligibility criteria. In the first step, Out_j identifies eligible packets. A packet P_{ij}^k in a crosspoint buffer with virtual start time $\widehat{VS}_{ij}^k$ less than or equal to $t - L/R$ is eligible, where t is the current system time. If there are multiple eligible packets, Out_j selects the one P_{ij}^k with the smallest virtual finish time $\widehat{VS}_{ij}^k$, retrieves it from B_{ij}, and sends it to the output line. If there are no eligible packets, Out_j waits for one. Note that the virtual start time and virtual finish time of a packet are carried with the packet when it is sent to the crosspoint buffer, and will be removed before it is sent to the output line. For easy understanding, the pseudo code description of GAPS is given in Table 1.

As can be seen, the input scheduling and output scheduling of GAPS are similar to WF^{2}Q. However, GAPS is different in that the leftover bandwidth of empty flows is not reallocated by the scheduling algorithm but by the bandwidth allocation algorithm.

We define the actual input start time and finish time of P_{ij}^k, denoted as IS_{ij}^k and IF_{ij}^k, to be the time that the first bit and last bit of P_{ij}^k leave Q_{ij} in GAPS, respectively. Apparently

$$IF_{ij}^k = IS_{ij}^k + \frac{L_{ij}^k}{R} \tag{7}$$

Define the actual output start time and finish time of P_{ij}^k, denoted as OS_{ij}^k and OF_{ij}^k, to be the time that the first bit and the last bit of P_{ij}^k leave B_{ij} in GAPS, respectively. It is obvious that

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

IV. PERFORMANCE ANALYSIS

In this section, we theoretically analyze the performance of GAPS. We will show that GAPS provides constant performance guarantees and has a bounded crosspoint buffer size.

A. Performance Guarantees

In this subsection, we show that GAPS achieves constant performance guarantees. According to the description of the GAPS algorithm, we have the following property.

Property 1: For any packet, its actual input start time is larger than or equal to its virtual start time, and its actual output start time is larger than or equal to its virtual start time plus $\frac{L}{R}$, i.e.

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

$$OS_{ij}^k \geq \widehat{VS}_{ij}^k + \frac{L}{R} \tag{10}$$

First we define some notations for input scheduling. We say that Q_{ij} is backlogged at time t, if there exists k such that $\widehat{VS}_{ij}^k \leq t \leq \widehat{VF}_{ij}^k$. 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 GAPS and GPS, respectively. Based on the definition of GPS, $\widehat{toB}_{ij}(t_1, t_2)$ can be calculated as

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

Use $toB_{i*}(t_1, t_2)$ and $\widehat{toB}_{i*}(t_1, t_2)$ to represent the total numbers of bits sent from In_i to all its crosspoint buffers during interval $[t_1, t_2]$ in GAPS and GPS, respectively, i.e.

$$toB_{i*}(t_1, t_2) = \sum_{x=1}^{N} toB_{ix}(t_1, t_2) \tag{12}$$

$$\widehat{toB}_{i*}(t_1, t_2) = \sum_{x=1}^{N} \widehat{toB}_{ix}(t_1, t_2) \tag{13}$$

The following lemma gives the relationship between $toB_{i*}(0, t)$ and $\widehat{toB}_{i*}(0, t)$.

Lemma 1: At any time, the number of bits sent from a specific input port in GAPS is larger than or equal to that in GPS, i.e.

$$toB_{i*}(0, t) \geq \widehat{toB}_{i*}(0, t) \tag{14}$$

Proof: We say that In_i is busy at time t if there exists k such that $IS_{ij}^k \leq t \leq IF_{ij}^k$, i.e. In_i sending bits at t in GAPS. Otherwise, In_i is idle.

Assume that $[t', t]$ is the last continuous busy period for In_i before t in GAPS. In other words, In_i is idle immediately before t', and is busy during $[t', t]$. Assume P_{ij}^k to be a packet that finished transmission in GPS before t', and thus $\widehat{VS}_{ij}^k < t'$, which means that P_{ij}^k is eligible for input scheduling in GAPS before t'. Since In_i was idle immediately before t', it indicates that P_{ij}^k finished transmission in input scheduling of GAPS before t'. The analysis applies to any packet transmitted in GPS before t', which means that all packets transmitted in GPS before t' have finished transmission in input scheduling of GAPS by t'. In other words,

$$toB_{i*}(0, t') \geq \widehat{toB}_{i*}(0, t') \tag{15}$$

On the other hand, because In_i is busy during $[t', t]$ in GAPS, we know that

$$\begin{aligned} toB_{i*}(t', t) &= R(t - t') \\ &\geq \int_{t'}^{t} \sum_{j=1}^{N} R_{ij}(x)dx \\ &\geq \sum_{j=1}^{N} \int_{t'}^{t} R_{ij}(x)\hat{q}_{ij}(x)dx \\ &= \sum_{j=1}^{N} \widehat{toB}_{ij}(t', t) \\ &= \widehat{toB}_{i*}(t', t) \end{aligned} \tag{16}$$

Adding (15) and (16), we have $toB_{i*}(0, t) \geq \widehat{toB}_{i*}(0, t)$. ■

The following lemma compares the service time of a packet in GAPS and in GPS.

Lemma 2: For any packet, its actual input start time in GAPS is less than or equal to its virtual finish time in GPS, i.e.

$$IS_{ij}^k \leq \widehat{VF}_{ij}^k \tag{17}$$

Proof: By contradiction, we assume that $IS_{ij}^k > \widehat{VF}_{ij}^k$. We look at the numbers of bits transmitted by time IS_{ij}^k in GAPS and GPS, respectively.

In GAPS, P_{ij}^k has not started transmission in input scheduling by IS_{ij}^k. Because input scheduling of GAPS schedules eligible packets based on their virtual finish time, all packets with virtual finish time greater than $\widehat{VF}_{ij}^k$ have not started transmission either. As a result,

$$toB_{i*}(0, IS_{ij}^k) \leq \sum_{j'=1}^{N} \sum_{k' \text{ s.t. } \widehat{VF}_{ij'}^{k'} \leq \widehat{VF}_{ij}^k} L_{ij'}^{k'} - L_{ij}^k \tag{18}$$

In GPS, since $IS_{ij}^k > \widehat{VF}_{ij}^k$ by the assumption, P_{ij}^k and all other packets with virtual finish time less than or equal to $\widehat{VF}_{ij}^k$ have finished transmission by IS_{ij}^k. Therefore

$$\begin{aligned} \widehat{toB}_{i*}(0, IS_{ij}^k) &\geq \widehat{toB}_{i*}(0, \widehat{VF}_{ij}^k) \\ &\geq \sum_{j'=1}^{N} \sum_{k' \text{ s.t. } \widehat{VF}_{ij'}^{k'} \leq \widehat{VF}_{ij}^k} L_{ij'}^{k'} \end{aligned} \tag{19}$$

Combining (18) and (19), we have

$$toB_{i*}(0, IS_{ij}^k) < \widehat{toB}_{i*}(0, IS_{ij}^k) \tag{20}$$

which is a contradiction to Lemma 1. ■

The next lemma compares $toB_{ij}(0, t)$ and $\widehat{toB}_{ij}(0, t)$.

Lemma 3: At any time, the difference between the numbers of bits sent from input port In_i to crosspoint buffer B_{ij}

in GAPS and GPS is greater than or equal to $-L$ and less than or equal to L, i.e.

$$-L \le toB_{ij}(0,t) - \widehat{toB}_{ij}(0,t) \le L \tag{21}$$

Proof: Without loss of generality, assume $t \in \left[\widehat{VF}_{ij}^{k-1}, \widehat{VF}_{ij}^{k}\right)$.

First, we prove $\widehat{toB}_{ij}(0,t) - toB_{ij}(0,t) \le L$. Based on Lemma 2, we know that $IS_{ij}^{k-1} \le \widehat{VF}_{ij}^{k-1} \le t$, and thus P_{ij}^{k-1} has started transmission by time t in input scheduling of GAPS. As a result

$$\begin{aligned} & toB_{ij}(0,t) \\ \ge\ & toB_{ij}\left(0, IS_{ij}^{k-1}\right) + \min\left(R\left(t - IS_{ij}^{k-1}\right), L_{ij}^{k-1}\right) \\ =\ & \sum_{x=1}^{k-2} L_{ij}^{x} + \min\left(R\left(t - IS_{ij}^{k-1}\right), L_{ij}^{k-1}\right) \end{aligned} \tag{22}$$

On the other hand, since $t < \widehat{VF}_{ij}^{k}$, P_{ij}^{k} has not finished transmission by t in GPS. Therefore

$$\begin{aligned} \widehat{toB}_{ij}(0,t) =\ & \int_{0}^{\widehat{VF}_{ij}^{k-1}} R_{ij}(x)\hat{q}_{ij}dx + \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{q}_{ij}(x)dx \\ =\ & \sum_{x=1}^{k-1} L_{ij}^{x} + \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{q}_{ij}(x)dx \end{aligned} \tag{23}$$

By (22) and (23), we have

$$\begin{aligned} & \widehat{toB}_{ij}(0,t) - toB_{ij}(0,t) \\ \le\ & L_{ij}^{k-1} + \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{q}_{ij}(x)dx \\ & - \min\left(R\left(t - IS_{ij}^{k-1}\right), L_{ij}^{k-1}\right) \\ =\ & \max\left(L_{ij}^{k-1} + \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{q}_{ij}(x)dx - R\left(t - IS_{ij}^{k-1}\right),\right. \\ & \left.\int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{q}_{ij}(x)dx\right) \end{aligned} \tag{24}$$

Because $IS_{ij}^{k-1} \le \widehat{VF}_{ij}^{k-1}$, we know

$$\begin{aligned} \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{q}_{ij}(x)dx \le\ & \int_{IS_{ij}^{k-1}}^{t} R_{ij}(x)dx \\ \le\ & \int_{IS_{ij}^{k-1}}^{t} R dx \\ =\ & R\left(t - IS_{ij}^{k-1}\right) \end{aligned} \tag{25}$$

Since $t < \widehat{VF}_{ij}^{k}$, we can obtain

$$\begin{aligned} \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{q}_{ij}(x)dx \le\ & \int_{\widehat{VF}_{ij}^{k-1}}^{\widehat{VF}_{ij}^{k}} R_{ij}(x)\hat{q}_{ij}(x)dx \\ =\ & L_{ij}^{k} \end{aligned} \tag{26}$$

Combining (24), (25), and (26)

$$\widehat{toB}_{ij}(0,t) - toB_{ij}(0,t) \le \max\left(L_{ij}^{k-1}, L_{ij}^{k}\right) \le L \tag{27}$$

Next, we prove $toB_{ij}(0,t) - \widehat{toB}_{ij}(0,t) \le L$. Because $t < \widehat{VF}_{ij}^{k} \le \widehat{VS}_{ij}^{k+1} \le IS_{ij}^{k+1}$, P_{ij}^{k+1} has not started transmission by t in input scheduling of GAPS. Thus, $toB_{ij}(0,t) \le \sum_{x=1}^{k} L_{ij}^{x}$. On the other hand, since $t \ge \widehat{VF}_{ij}^{k-1}$, P_{ij}^{k-1} has finished transmission by t in GPS. Therefore, $\widehat{toB}_{ij}(0,t) \ge \sum_{x=1}^{k-1} L_{ij}^{x}$. Combining the above two equations, we obtain

$$toB_{ij}(0,t) - \widehat{toB}_{ij}(0,t) \le L_{ij}^{k} \le L \tag{28}$$

■

Correspondingly, we define some notations for output scheduling. We say that B_{ij} is backlogged at time t, if there exists k such that $\widehat{VS}_{ij}^{k} \le t \le \widehat{VF}_{ij}^{k}$. B_{ij} is backlogged at t if B_{ij} has buffered bits at t in GPS. 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. Note that $\hat{b}_{ij}(t) = \hat{q}_{ij}(t)$ by neglecting the propagation delay, because GPS is a fluid based system.

As defined earlier, 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 GAPS and GPS, respectively. $\widehat{toO}_{ij}(t_1,t_2)$ can be calculated as

$$\widehat{toO}_{ij}(t_1,t_2) = \int_{t_1}^{t_2} R_{ij}(x)\hat{b}_{ij}(x)dx \tag{29}$$

Use $toO_{*j}(t_1,t_2)$ and $\widehat{toO}_{*j}(t_1,t_2)$ to represent the total numbers of bits sent from all the crosspoint buffers to Out_j during interval $[t_1,t_2]$ in GAPS and GPS, respectively, i.e.

$$toO_{*j}(t_1,t_2) = \sum_{x=1}^{N} toO_{xj}(t_1,t_2) \tag{30}$$

$$\widehat{toO}_{*j}(t_1,t_2) = \sum_{x=1}^{N} \widehat{toO}_{xj}(t_1,t_2) \tag{31}$$

We have a corresponding version of Lemma 1 for output scheduling as follows.

Lemma 4: The number of bits received by a specific output port in GAPS by time t is larger than or equal to that in GPS by time $t - \frac{L}{R}$, i.e.

$$toO_{*j}(0,t) \ge \widehat{toO}_{*j}(0, t - \frac{L}{R}) \tag{32}$$

Proof: We say that Out_j is busy at time t if there exists k such that $OS_{ij}^{k} \le t \le OF_{ij}^{k}$, i.e. Out_j receiving bits at t in GAPS. Otherwise, Out_j is idle.

Assume that $[t',t]$ is the last continuous busy period for Out_j before t in GAPS. In other words, Out_j is idle immediately before t', and is busy during $[t',t]$. Assume P_{ij}^{k} to be a packet that finished transmission in GPS before

$t' - \frac{L}{R}$, and thus $\widehat{VF}_{ij}^{k} < t' - \frac{L}{R}$. By Lemma 2, we have $IS_{ij}^{k} \leq \widehat{VF}_{ij}^{k}$, or

$$IF_{ij}^{k} \leq \widehat{VF}_{ij}^{k} + \frac{L}{R} < t' \tag{33}$$

This indicates that P_{ij}^{k} arrived at B_{ij} before t' in GAPS, and it is eligible for output scheduling before t'. Since Out_j was idle immediately before t', it means that P_{ij}^{k} finished transmission in output scheduling of GAPS before t'. The analysis applies to any packet transmitted in GPS before $t' - \frac{L}{R}$, which means that all packets transmitted in GPS before $t' - \frac{L}{R}$ have finished transmission in output scheduling of GAPS by t'. In other words,

$$toO_{*j}(0, t') \geq \widehat{toO}_{*j}(0, t' - \frac{L}{R}) \tag{34}$$

On the other hand, because Out_j is busy during $[t', t]$ in GAPS, we know that

$$\begin{aligned} toO_{*j}(t', t) =& \ R(t - t') \\ =& \ R\left(\left(t - \frac{L}{R}\right) - \left(t' - \frac{L}{R}\right)\right) \\ \geq& \ \int_{t' - \frac{L}{R}}^{t - \frac{L}{R}} \sum_{i=1}^{N} R_{ij}(x)dx \\ \geq& \ \sum_{i=1}^{N} \int_{t' - \frac{L}{R}}^{t - \frac{L}{R}} R_{ij}(x)\hat{b}_{ij}(x)dx \\ =& \ \widehat{toO}_{*j}(t' - \frac{L}{R}, t - \frac{L}{R}) \end{aligned} \tag{35}$$

Adding (34) and (35), we have $toO_{*j}(0, t) \geq \widehat{toO}_{*j}(0, t - \frac{L}{R})$. ■

Similarly, we have the corresponding version of Lemma 2 for output scheduling of GAPS.

Lemma 5: For any packet, its actual output start time in GAPS is less than or equal to its virtual finish time in GPS plus $\frac{L}{R}$, i.e.

$$OS_{ij}^{k} \leq \widehat{VF}_{ij}^{k} + \frac{L}{R} \tag{36}$$

The proof is similar to that of Lemma 2 but based on Lemma 4, and is omitted.

The follow theorem shows that GAPS achieves constant performance guarantees.

Theorem 1: At any time, the difference between the numbers of bits transmitted by a flow to the output port in GAPS and GPS is greater than or equal to $2L$ and less than or equal to L, i.e.

$$-2L \leq toO_{ij}(0, t) - \widehat{toO}_{ij}(0, t) \leq L \tag{37}$$

Proof: Without loss of generality, assume that $t \in \left[\widehat{VF}_{ij}^{k-1} + \frac{L}{R}, \widehat{VF}_{ij}^{k} + \frac{L}{R}\right)$.

First, we prove $\widehat{toO}_{ij}(0, t) - toO_{ij}(0, t) \leq 2L$. Based on Lemma 5, we know that $OS_{ij}^{k-1} \leq \widehat{VF}_{ij}^{k-1} + \frac{L}{R} \leq t$, and thus P_{ij}^{k-1} has started transmission by time t in output scheduling of GAPS. As a result

$$\begin{aligned} & toO_{ij}(0, t) \\ \geq& \ toO_{ij}\left(0, OS_{ij}^{k-1}\right) + \min\left(R\left(t - OS_{ij}^{k-1}\right), L_{ij}^{k-1}\right) \\ =& \ \sum_{x=1}^{k-2} L_{ij}^{x} + \min\left(R\left(t - OS_{ij}^{k-1}\right), L_{ij}^{k-1}\right) \end{aligned} \tag{38}$$

On the other hand, in GPS we have

$$\begin{aligned} & \widehat{toO}_{ij}(0, t) \\ =& \int_{0}^{\widehat{VF}_{ij}^{k-1}} R_{ij}(x)\hat{b}_{ij}(x)dx + \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{b}_{ij}(x)dx \\ \leq& \sum_{x=1}^{k-1} L_{ij}^{x} + \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{b}_{ij}(x)dx \end{aligned} \tag{39}$$

By (38) and (39), we have

$$\begin{aligned} & \widehat{toO}_{ij}(0, t) - toO_{ij}(0, t) \\ \leq& \ L_{ij}^{k-1} + \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{b}_{ij}(x)dx \\ & - \min\left(R\left(t - OS_{ij}^{k-1}\right), L_{ij}^{k-1}\right) \\ =& \ \max\left(L_{ij}^{k-1} + \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{b}_{ij}(x)dx - R\left(t - OS_{ij}^{k-1}\right), \right. \\ & \left. \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{b}_{ij}(x)dx\right) \end{aligned} \tag{40}$$

Because $OS_{ij}^{k-1} - \frac{L}{R} \leq \widehat{VF}_{ij}^{k-1}$ by Lemma 5, we know

$$\begin{aligned} \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{b}_{ij}(x)dx \leq& \int_{OS_{ij}^{k-1} - \frac{L}{R}}^{t} R_{ij}(x)dx \\ \leq& \int_{OS_{ij}^{k-1} - \frac{L}{R}}^{t} Rdx \\ =& \ R\left(t - OS_{ij}^{k-1} + \frac{L}{R}\right) \\ =& \ R\left(t - OS_{ij}^{k-1}\right) + L \end{aligned} \tag{41}$$

Since $t < \widehat{VF}_{ij}^{k} + \frac{L}{R}$, we can obtain

$$\begin{aligned} & \int_{\widehat{VF}_{ij}^{k-1}}^{t} R_{ij}(x)\hat{b}_{ij}(x)dx \\ \leq& \int_{\widehat{VF}_{ij}^{k-1}}^{\widehat{VF}_{ij}^{k} + \frac{L}{R}} R_{ij}(x)\hat{b}_{ij}(x)dx \\ \leq& \int_{\widehat{VF}_{ij}^{k-1}}^{\widehat{VF}_{ij}^{k}} R_{ij}(x)\hat{b}_{ij}(x)dx + \int_{\widehat{VF}_{ij}^{k}}^{\widehat{VF}_{ij}^{k} + \frac{L}{R}} R_{ij}(x)dx \\ \leq& \ L_{ij}^{k} + \int_{\widehat{VF}_{ij}^{k}}^{\widehat{VF}_{ij}^{k} + \frac{L}{R}} Rdx \\ \leq& \ L_{ij}^{k} + L \end{aligned} \tag{42}$$

Combing (40), (41), and (42)

$$\begin{aligned}\widehat{toO}_{ij}(0,t) - toO_{ij}(0,t) \leq & \max\left(L_{ij}^{k-1} + L, L_{ij}^{k} + L\right) \\ \leq & \ 2L \end{aligned} \tag{43}$$

Next, we prove $toO_{ij}(0,t) - \widehat{toO}_{ij}(0,t) \leq L$. Because a packet P_{ij} has to be transmitted to B_{ij} before sent to Out_j, it is obvious that $toO_{ij}(0,t) \leq toB_{ij}(0,t)$. In addition, by neglecting propagation delay, we have $\widehat{toO}_{ij}(0,t) = \widehat{toB}_{ij}(0,t)$. Thus

$$\begin{aligned} toO_{ij}(0,t) - \widehat{toO}_{ij}(0,t) \leq & \ toB_{ij}(0,t) - \widehat{toB}_{ij}(0,t) \\ \leq & \ L \end{aligned} \tag{44}$$

■

The next theorem gives the delay bounds. For easy analysis of the delay difference lower bound, we assume that the allocated bandwidth $R_{ij}(t)$ of F_{ij} is a constant R_{ij} during interval $\left[\min\left(IS_{ij}^k, \widehat{VS}_{ij}^k\right), \max\left(OF_{ij}^k, \widehat{VF}_{ij}^k\right)\right]$.

Theorem 2: For any packet P_{ij}^k, the difference between its departure time in GAPS and 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 $\frac{2L}{R}$, i.e.

$$-L_{ij}^k\left(\frac{1}{R_{ij}} - \frac{2}{R}\right) \leq OF_{ij}^k - \widehat{VF}_{ij}^k \leq \frac{2L}{R} \tag{45}$$

Proof: First, we prove $OF_{ij}^k - \widehat{VF}_{ij}^k \geq -L_{ij}^k\left(\frac{1}{R_{ij}} - \frac{2}{R}\right)$. Because a packet P_{ij}^k has to be buffered at B_{ij} before sent to Out_j, it is obvious that

$$OF_{ij}^k \geq IF_{ij}^k + \frac{L_{ij}^k}{R} \tag{46}$$

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

$$\begin{aligned} \widehat{VF}_{ij}^k \leq & \ IF_{ij}^k + L_{ij}^k\left(\frac{1}{R_{ij}} - \frac{1}{R}\right) \\ \leq & \ OF_{ij}^k + L_{ij}^k\left(\frac{1}{R_{ij}} - \frac{2}{R}\right) \end{aligned} \tag{47}$$

Next, we prove $OF_{ij}^k - \widehat{VF}_{ij}^k \leq \frac{2L}{R}$. By Lemma 5, we know $OS_{ij}^k \leq \widehat{VF}_{ij}^k + \frac{L}{R}$. Because $OF_{ij}^k = OS_{ij}^k + \frac{L}{R}$, we obtain $OF_{ij}^k - \widehat{VF}_{ij}^k \leq \frac{2L}{R}$. ■

B. Crosspoint Buffer Size Bound

Crosspoint buffers are expensive on-chip memories, and it is desired that each crosspoint has only a limited size buffer. To avoid overflow at crosspoint buffers, we would like to find the maximum number of bits buffered at any crosspoint.

Theorem 3: In GAPS, the maximum number of bits buffered at any crosspoint buffer is upper bounded by $3L$, i.e.

$$toB_{ij}(0,t) - toO_{ij}(0,t) \leq 3L \tag{48}$$

Proof: By Lemma 3,

$$toB_{ij}(0,t) - \widehat{toB}_{ij}(0,t) \leq L \tag{49}$$

By Theorem 1,

$$\widehat{toO}_{ij}(0,t) - toO_{ij}(0,t) \leq 2L \tag{50}$$

Because GPS is a fluid based system, we have $\widehat{toB}_{ij}(0,t) = \widehat{toO}_{ij}(0,t)$ by neglecting the propagation delay. Summing the above equations, we have proved the theorem. ■

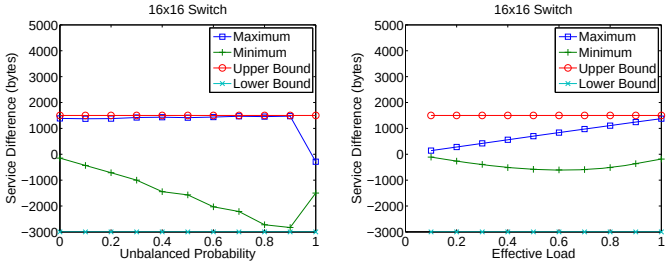
V. Simulation Results

We have conducted simulations to verify the analytical results obtained in Section IV and evaluate the effectiveness of GAPS.

In the simulations, we consider a 16×16 buffered crossbar switch without speedup. Each input port and output port have bandwidth of 1G bps. Since GAPS can directly handle variable length packets, we set packet length to be uniformly distributed between 40 and 1500 bytes [19]. For bandwidth allocation, we use the same model as that in [9] and [15]. The allocated bandwidth $R_{ij}(t)$ of flow F_{ij} at time t is defined by an unbalanced probability w as follows

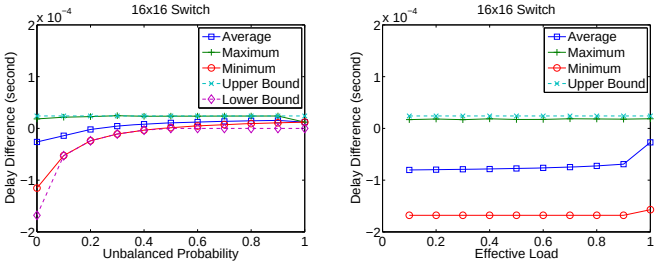
$$R_{ij}(t) = \begin{cases} R\left(w + \frac{1-w}{N}\right), & \text{if } i = j \\ R\frac{1-w}{N}, & \text{if } i \neq j \end{cases} \tag{51}$$

When $w = 0$, an input port In_i has the same amount of allocated bandwidth $\frac{R}{N}$ at each output port. Otherwise, In_i has more allocated bandwidth at Out_i, which is called the hotspot destination. Because each flow is allocated a specific amount of bandwidth, it is necessary to have admission control flow to avoid over-subscription. Arrival of a flow F_{ij} is constrained by a leaky bucket $(l \times R_{ij}(t), \sigma_{ij})$, 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.



(a) (b)

Figure 3. Service difference of GAPS. (a) With different unbalanced probabilities. (b) With different loads.



(a) (b)

Figure 4. Delay difference of GAPS. (a) With different unbalanced probabilities. (b) With different loads.

A. Service Guarantees

By Theorem 1, we know that the service difference of a flow in GAPS and GPS at any time has a lower bound of $-2L$ and upper bound of L. We look at the simulation data on service guarantees.

Figure 3(a) shows the minimum and maximum service differences among all the flows during the entire simulation run under traffic pattern one. As can be seen, the minimum service difference is always greater than the lower bound. It drops gradually when the unbalanced probability increases. This indicates that when the traffic distribution is more unbalanced, flows tend to transmit less traffic in GAPS than in GPS. Note that when the unbalanced probability becomes one, the minimum service difference jumps suddenly to -1500 bytes. The reason is that when the unbalanced probability is one, all packets of In_i go to Out_i, and there is no switching necessary. The only difference between GAPS and GPS is that a packet needs to be buffered at the crosspoint buffer in GAPS but not in GPS. Thus, the service difference in the worst case is equal to the maximum packet length. On the other hand, the maximum service difference is always less than but very close to the upper bound. However, when the unbalanced probability becomes one, the maximum service difference drops to a negative value. As analyzed in the above, when the unbalanced probability is one, the only difference between GAPS and GPS is the extra buffering at the crosspoint buffer. As a result, a flow always transmits less traffic in GAPS than in GPS, and the actual maximum service difference depends on the length of the first packet, which is a random number between 40 and 1500.

Figure 3(b) shows the minimum and maximum service differences under traffic pattern two. The minimum service difference is always greater than the lower bound, and keeps relatively constant. This indicates that the minimum service difference is not sensitive to the change of effective load. The maximum service difference increases steadily with the effective load, but is always less than the upper bound.

B. Delay Guarantees

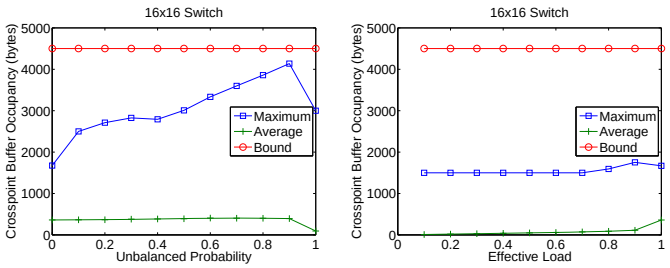
Theorem 2 gives the lower bound and upper bound for the delay difference of a packet in GAPS and GPS. In this subsection, we present the simulation data on delay guarantees.

Figure 4(a) shows the minimum, maximum, and average delay differences of a representative flow F_{11} under traffic pattern one. Note that the delay difference lower bound in Theorem 2 assumes fixed allocated bandwidth R_{ij} and depends on the packet length L_{ij}^k. For easy plotting of the figure, we calculate the delay difference lower bound for all packets of flow F_{ij} as follows

$$-L_{ij}^k\left(\frac{1}{R_{ij}}-\frac{2}{R}\right) \geq \begin{cases} -L\left(\frac{1}{R_{ij}}-\frac{2}{R}\right), & \text{if } R_{ij} \leq \frac{R}{2} \\ 0, & \text{if } R_{ij} > \frac{R}{2} \end{cases} \quad (52)$$

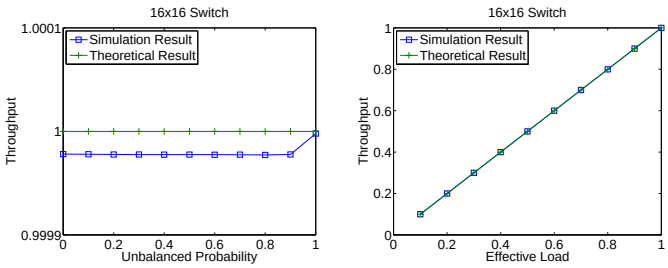
As can be seen, the minimum delay difference is almost coincident with the theoretical lower bound, and the maximum delay difference is almost identical with the upper bound. This shows that the theoretical bounds are tight. While the minimum delay difference increase as the unbalanced probability increases, the maximum delay difference is not sensitive to the change of the unbalanced probability. The average delay difference is initially negative. This is reasonable because when the traffic is uniformly distributed, most packets leave earlier than their departure time in GPS. When the unbalanced probability increases, the average delay difference also increases. Note that when the unbalanced probability becomes one, the minimum, maximum, and average delay differences all become 1.2×10^{-5} second. The explanation is the same as above, and a packet P_{ij}^k needs to wait until $\widehat{VS}_{ij}^k+\frac{L}{R}=\widehat{VS}_{ij}^k+1.2\times10^{-5}$ second to start transmission from B_{ij} in GAPS.

Figure 4(b) shows the minimum, maximum, and average delay differences of flow F_{11} under traffic pattern two. Because the delay difference lower bound in Theorem 2 assumes fixed allocated bandwidth, the lower bound curve cannot be plotted. We can still see that the maximum delay difference is always less than and very close to the upper bound. Both the minimum and average delay differences



(a) (b)

Figure 5. Crosspoint buffer occupancy of GAPS. (a) With different unbalanced probabilities. (b) With different loads.



(a) (b)

Figure 6. Throughput of GAPS. (a) With different unbalanced probabilities. (b) With different loads.

are relatively constant, and the average delay difference is always negative, which means that most packets depart earlier in GAPS than in GPS.

C. Crosspoint Buffer Size

Theorem 3 gives the bound of the crosspoint buffer size as $3L$. In this subsection, we look at the maximum and average crosspoint buffer occupancies in the simulations.

Figure 5(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 to about 3000 bytes when the unbalanced probability becomes one. The reason is that now there will be at most $2L$ bits buffered in crosspoint buffers B_{ii}. For the average occupancy, it does not change significantly with different unbalanced probabilities. It drops to about 100 bytes, when the unbalanced probability becomes one. This is because only crosspoint buffers B_{ii} are now used, and the remaining crosspoint buffers are empty. We can find that the average occupancy is more affected by the load than the unbalanced probability.

Figure 5(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 150 bytes before the load increases to one. This also confirms the previous observation that the average occupancy is determined by the effective load.

D. Throughput

Next, we present the simulation data on throughput. Figure 6(a) shows the throughput under traffic pattern one. We can see that the throughput for all unbalanced probabilities is greater than 99.99%, which demonstrates that GAPS practically achieves 100% throughput. Figure 6(b) shows the throughput under traffic pattern two. As can be seen, the throughput grows consistently with the effective load, and finally reaches one.

VI. Conclusions

Recent development in VLSI technology has made buffered crossbar switches to be feasible, and they demonstrate unique advantages over traditional unbuffered crossbar switches. The current emulation approach for buffered crossbar switches to provide performance guarantees has difficulty in providing tight constant performance guarantees, because of its inability to emulate WF^{2}Q. To address the issue, we have presented in this paper the guaranteed-performance asynchronous packet scheduling (GAPS) algorithm for buffered crossbar switches. GAPS requires no speedup, and directly handles variable length packets without segmentation and reassembly (SAR). Different input ports and output ports conduct scheduling independently without any data exchange. We show by theoretical analysis that GAPS achieves constant performance guarantees. In addition, we prove that GAPS has a crosspoint buffer size bound of $3L$. Finally, we present simulation data to verify the analytical results and evaluate the effectiveness of GAPS.

References

[1] G. Kornaros, "BCB: a buffered crossBar switch fabric utilizing shared memory," *EUROMICRO 2006*, Aug. 2006.

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

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

[4] K. Yoshigoe, K. Christensen, and A. Jacob, "The RR/RR CICQ switch: hardware design for 10-Gbps link speed," *22nd IEEE International Performance, Computing, and Communications Conference*, pp. 481-485, Phoenix, AZ, Apr. 2003.

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

[6] S. He et al., "On Guaranteed Smooth Switching for Buffered Crossbar Switches," *IEEE/ACM Transactions on Networking*, Jun. 2008.

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

[8] 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.

[9] 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

[10] B. Magill, C. Rohrs and R. Stevenson, "Output-queued switch emulation by fabrics with limited memory," *IEEE Journal on Selected Areas in Communications*, vol 21, no. 4, pp. 606-615, May 2003.

[11] L. Mhamdi and M. Hamdi, "Output queued switch emulation by a one-cell-internally buffered crossbar switch," *IEEE GLOBECOM 2003*, San Francisco, CA, Dec. 2003.

[12] D. Pan and Y. Yang, "Providing flow based performance guarantees for buffered crossbar switches," *IEEE IPDPS 2008*, Miami, FL, Apr. 2008.

[13] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, and N. Chrysos, "Variable packet size buffered crossbar (CICQ) switches," *Proc. IEEE ICC 2004*, Paris, France, June 2004.

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

[15] R. Rojas-Cessa et al., "CIXB-1: Combined input-once-cell-crosspoint buffered switch," *IEEE HPSR 2001*, Jul. 2001.

[16] X. Zhang, S. Mohanty, and L. Bhuyan, "Adaptive max-min fair scheduling in buffered crossbar switches without speedup," *IEEE INFOCOM 2007*, Anchorage, AK, May 2007.

[17] H. Ahmadi and W. Denzel, "A survey of modern high-performance switching techniques," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 1091-1103, Sep. 1989.

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

[19] C. Farleigh et al., "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, no. 6, pp. 6-16, Nov. 2003.

[20] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," *IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000.

[21] 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.

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

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

[24] 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.

[25] S. Iyer and N. McKeown, "Analysis of the parallel packet switch architecture," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 2, pp. 314-324, Apr. 2003.

[26] S. Chuang, A. Goel, N. McKeown and B. Prabhkar, "Matching output queueing with a combined input output queued switch," *IEEE INFOCOM'99*, pp. 1169-1178, New York, 1999.

[27] I. Stoica and H. Zhang, "Exact emulation of an output queueing switch by a combined input output queueing Switch," *6th IEEE/IFIP IWQoS '98*, Napa, CA, May 1998.

[28] D. Stephens and H. Zhang, "Implementing distributed packet fair queueing in a scalable switch architecture," *IEEE INFOCOM 1998*, San Francisco, CA, March 1998.

[29] N. Ni and L. Bhuyan, "Fair scheduling for input buffered switches," *Cluster Computing*, Apr. 2003.

[30] X. Zhang and L. Bhuyan, "Deficit round-robin scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, no. 4, pp. 584-594, May 2003.

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

[32] 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.

[33] 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.

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

[35] E. Leonardi, M. Mellia, F Neri, and M. Marsan, "On the stability of input-queued switches with speed-up," *IEEE/ACM Trans. Networking*, vol. 9, no. 1, pp. 104-118, Feb. 2001.