Strong Performance Guarantees for Asynchronous Buffered Crossbar Schedulers

Jonathan S. Turner, *Fellow, IEEE*

***Abstract*—Crossbar-based switches are commonly used to implement routers with throughputs up to about 1 Tb/s. The advent of crossbar scheduling algorithms that provide strong performance guarantees now makes it possible to engineer systems that perform well, even under extreme traffic conditions. Until recently, such performance guarantees have only been developed for crossbars that switch cells rather than variable length packets. Cell-based crossbars incur a worst-case bandwidth penalty of up to a factor of two, since they must fragment variable length packets into fixed length cells. In addition, schedulers for cell-based crossbars may fail to deliver the expected performance guarantees when used in routers that forward packets. We show how to obtain performance guarantees for asynchronous crossbars that are directly comparable to those previously developed for synchronous, cell-based crossbars. In particular we define derivatives of the Group by Virtual Output Queue (GVOQ) scheduler of Chuang *et al.* and the Least Occupied Output First Scheduler of Krishna *et al.* and show that both can provide strong performance guarantees in systems with speedup 2. Specifically, we show that these schedulers are work-conserving and that they can emulate an output-queued switch using any queueing discipline in the class of restricted Push-In, First-Out queueing disciplines. We also show that there are schedulers for segment-based crossbars, (introduced recently by Katevenis and Passas) that can deliver strong performance guarantees with small buffer requirements and no bandwidth fragmentation.**

***Index Terms*—Asynchronous crossbars, crossbar schedulers, performance guarantees, routers, switches.**

I. Introduction

CROSSBAR switches have long been a popular choice for transferring data from inputs to outputs in mid-range performance switches and routers [1]. Unlike bus-based switches, crossbars can provide throughputs approaching 1 Tb/s, while allowing individual line cards to operate at speeds comparable to the external links.

However the control of high performance crossbars is challenging, requiring crossbar schedulers that match inputs to outputs in the time it takes for a minimum length packet to be forwarded. The matching selected by the scheduler has a major influence on system performance, placing a premium on algorithms that can produce high quality matchings in a very short period of time.

Traditionally, crossbars schedulers have been evaluated largely on the basis of how they perform on random traffic arrival patterns that do not cause long term overloads at inputs or outputs. Most often, such evaluations have been carried out using simulation [14]. Recently, there has been a growing body of work providing rigorous performance guarantees for such systems [11], [15] in the context of well-behaved, random traffic. A separate thread of research concentrates on schedulers that can provide strong performance guarantees that apply to arbitrary traffic patterns [3], [8], [18], including adversarial traffic that may overload some outputs for extended periods of time. The work reported here belongs to this second category. Since the internet lacks comprehensive mechanisms to manage traffic, extreme traffic conditions can occur in the internet due to link failures, route changes or simply unusual traffic conditions. For these reasons, we argue that it is important to understand how systems perform when they are subjected to such extreme conditions. Moreover, we argue that strong performance guarantees are desirable in backbone routers, if they can be obtained at an acceptable cost.

There are two fundamental properties that are commonly used to evaluate crossbar schedulers in this worst-case sense. A scheduler is said to be *work-conserving* if an output link is kept busy so long as there are packets addressed to the output, anywhere in the system. A scheduler is said to be *order-preserving* if it is work-conserving and it always forwards packets in the order in which they arrived. A crossbar with an order-preserving scheduler faithfully emulates an ideal nonblocking switch with FIFO output queues. In their seminal paper, Chuang, *et al.* provided the first example of an order-preserving scheduler [3] for a crossbar with small speedup, where the speedup of a crossbar switch is the ratio of its ideal throughput to the total capacity of its external links. So a crossbar with a speedup of S has the potential to forward data S times faster than the input links can supply it. In fact, Chuang, *et al.* showed a stronger property; that certain schedulers can be specialized to emulate an output queued switch that implements any one of a large class of scheduling algorithms at the outputs.

Until recently, strong performance guarantees have been available only for crossbars that forward fixed length cells. There is a sound practical justification for concentrating on such systems, since routers commonly use cell-based crossbars. Variable length packets are received at input line cards, segmented into fixed length cells for transmission through the crossbar and reassembled at the output line cards. This simplifies the implementation of the crossbar and allows for synchronous operation, which allows the scheduler to make

Manuscript received November 25, 2007; revised May 22, 2008; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C.-S. Chang. First published November 25, 2008; current version published August 19, 2009. This work was supported by the National Science Foundation (Award #0325291). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

The author is with the Department of Computer Science and Engineering, Washington University, St. Louis, MO 63130-4899 USA (e-mail: jon.turner@wustl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2008.2006221

better decisions than would be possible with asynchronous operation. Unfortunately, cell-based crossbar schedulers that deliver strong performance guarantees when viewed from the edge of the crossbar, can fail to deliver those guarantees for the router as a whole. For example, a system using a work-conserving cell-based scheduler can fail to keep an outgoing link busy, even when there are complete packets for that output present in the system.

We show that strong performance guarantees can be provided for packets, using asynchronous crossbars that directly handle packets, rather than cells, if the crossbars are equipped with a moderate amount of internal buffer space. Specifically, we define packet-oriented derivatives of the Group by Virtual Output Queue algorithm (GVOQ) of [3] and the Least Occupied Output First Algorithm (LOOFA) of [8], [18] and show that they can deliver strong performance guarantees for systems with a speedup of 2. Because our crossbar schedulers operate asynchronously, we have had to develop new methods for analyzing their performance. These methods now make it possible to evaluate asynchronous crossbars in a way that is directly comparable to synchronous crossbars.

The use of buffered crossbars is not new. An early ATM switch from Fujitsu used buffered crossbars, for example [17]. However, most systems use unbuffered crossbars, because the addition of buffers to each of the n^2 crosspoints in an $n \times n$ crossbar has been viewed as prohibitively expensive. There has recently been renewed interest in buffered crossbars [4], [6], [9], [10], [12], [16], [19], [20]. A recent paper by Chuang *et al.* [4] advocates the use of buffers in cell-based crossbars in order to reduce the complexity of the scheduling algorithms. The authors argue that ongoing improvements in electronics now make it feasible to add buffering to a crossbar, without requiring an increase in the number of integrated circuit components. Hence, the cost impact of adding buffering is no longer a serious obstacle. Our results add further weight to the case for buffered crossbars, as the use of buffering allows inputs and outputs to operate independently and asynchronously, allowing variable length packets to be handled directly. Katevenis *et al.* [9], [10] have also advocated the use of buffered crossbars for variable length packets and have demonstrated their feasibility by implementing a 32 port buffered crossbar with 2 KB buffers at each crosspoint.

Section II discusses the differences between switching cells and switching packets, and explains how buffered crossbars are particularly advantageous for systems that directly switch packets. Section III defines the terminology and notation used in the analysis to follow. Section IV collects several key lemmas that are used repeatedly in the analysis. Section V presents strong performance guarantees for a packet variant of the Group by Virtual Output Queue crossbar scheduler. Section VI presents a similar set of guarantees for a packet variant of the Least Occupied Output First scheduler. Section VII explains how our asynchronous crossbar scheduling algorithms can be used in systems that switch variable length segments rather than cells, reducing the amount of memory required by crossbar buffers by more than order of magnitude. Finally, Section VIII provides some closing remarks, including a discussion of several ways this work can be extended.

II. Switching Packets vs. Cells

As noted in the introduction, most crossbar-based routers, segment packets into cells at input line cards, before forwarding them through the crossbar to output line cards, where they are reassembled into packets. This enables synchronous operation, allowing the crossbar scheduler to make decisions involving all inputs and outputs at one time.

Unfortunately, cell-based crossbars have some drawbacks. One is simply the added complication of segmentation and reassembly. More seriously, the segmentation of packets into cells can lead to degraded performance if the incoming packets cannot be efficiently packed into fixed length cells. In the worst-case, arriving packets may be slightly too large to fit into a single cell, forcing the input line cards to forward them in two cells. This effectively doubles the bandwidth that the crossbar requires in order to handle worst-case traffic. While one can reduce the impact of this problem by allowing parts of more than one packet to occupy the same cell, this adds complexity and does nothing to improve performance in the worst-case.

In addition, crossbar schedulers that operate on cells, without regard to packet boundaries, can fail to deliver the expected guarantees from the perspective of the system as a whole. In a system that uses a cell-based crossbar scheduler, an output line card can typically begin transmission of a packet on its outgoing link only after all cells of the packet have been received. Consider a scenario in which n input line cards receive packets of length L at time t, all addressed to the same output. If the length of the cell used by the crossbar is C, each packet must be segmented into $\lceil L/C \rceil$ cells for transmission through the fabric. A crossbar scheduler that operates on cells has no reason to prefer one input over another. Assuming that it forwards cells from each input in a fair fashion, at least $n(\lceil L/C \rceil - 1)$ cells must pass through the crossbar before the output line card has a complete packet that it can forward on the output link. While some delay between the arrival of a packet and its transmission on the output link is unavoidable, delays that are substantially longer than the time it takes to receive a packet on the link are clearly undesirable. In this situation, the delay is about n times larger than the time taken for the packet to be received. Interestingly, one can obtain strong performance guarantees for packets using cell-based schedulers that are *packet-aware*. We discuss this in Section VII.

There are a few previous studies of the performance of bufferless crossbars that switch packets, rather than cells. References [5], [13] focus on performance for well-behaved random traffic, so are not directly comparable to the results presented here. On the other hand, [2] studies packet-mode emulation of unbuffered crossbars and shows that strong performance guarantees can be obtained for such systems. However, the frame-based scheduling methods used in [2] impose a delay that can be several orders of magnitude larger than the very modest delays imposed by the schedulers studied here.

Asynchronous crossbars offer an alternative to cell-based crossbars. They eliminate the need for segmentation and reassembly and are not subject to bandwidth fragmentation, allowing one to halve the worst-case bandwidth required by the crossbar. Unfortunately, there is no obvious way to obtain

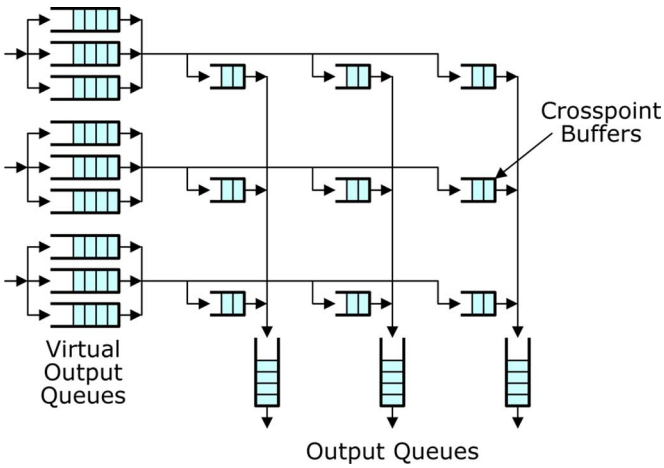

Fig. 1. Buffered crossbar.

strong performance guarantees for unbuffered asynchronous crossbars, since the ability of the scheduler to coordinate the movement of traffic through the system, seems to depend on its ability to make decisions involving all inputs and outputs at one time. A scheduler that operates on packets must deal with the asynchronous nature of packet arrivals, and must schedule packets as they arrive and as the inputs and outputs of the crossbar become available. In particular, if a given input line card finishes sending a packet to the crossbar at time t, it must then select a new packet to send to the crossbar. It may have packets that it can send to several different outputs, but its choice of output is necessarily limited to those outputs that are not currently receiving packets from other inputs. This can prevent it from choosing the output that it would prefer, were its choices not so constrained. One can conceivably ameliorate this situation by allowing an input to select an output that will become available in the near future, but this adds complication and sacrifices some of the crossbar bandwidth. Moreover, it is not clear that such a strategy can lead to a scheduling algorithm with good worst-case performance and small speedup.

The use of buffered crossbars offers a way out of this dilemma. The addition of buffers to each crosspoint of an $n \times n$ crossbar effectively decouples inputs from outputs, enabling the asynchronous operation that variable length packets seem to require. A diagram of a system using a buffered crossbar is shown in Fig. 1. In addition to the now conventional Virtual Output Queues (VOQ) at each input, a buffered crossbar has a small buffer at each of its crosspoints. As pointed out in [4], the buffers allow inputs and outputs to operate independently, enabling the use of simpler crossbar scheduling mechanisms, but the buffers have an even greater import for asynchronous crossbars. With buffers, whenever an input finishes sending a packet to the crossbar, it can select a packet from one of its VOQs, so long as the corresponding crosspoint buffer has room for the packet. We show that crosspoint buffers of modest size are sufficient to allow strong performance guarantees with the same speedup required by cell-based schedulers.

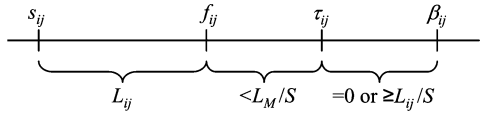

Fig. 2. Basic definitions for active periods.

III. Preliminaries

To start, we introduce common notations that will be used in the analysis to follow. We say a packet x is an ij-packet if it arrived at input i and is to be forwarded on output j. We let $s(x)$ denote the time at which the first bit of x is received on an input link and we let $f(x)$ be the time at which the last bit is received. We let $L(x)$ denote the number of bits in x and L_M denote the maximum packet length (in bits). The time unit is the time it takes for a single bit to be transferred on an external link, so $f(x) - s(x) = L(x)$. The time at which a new packet is selected by an input and sent to the crossbar is referred to as an *input scheduling event*. We also define to the time at which an active period ends to be an input event. The time at which an output selects a packet from one of its crosspoint buffers is referred to as an *output scheduling event*. We use *event* to refer to either type, when the type is clear from the context.

We let V_{ij} denote the VOQ at input i that contains packets for output j and we let $V_{ij}(t)$ denote the number of bits in V_{ij} at time t. Similarly, we let B_{ij} denote the crosspoint buffer for packets from input i to output $j, B_{ij}(t)$ denote the number of bits in B_{ij} at time t, and B denote the capacity of the crosspoint buffers. For all quantities that include a time parameter, we sometimes omit the time parameter.

We focus on schedulers for systems in which packets are fully buffered at the input line cards where they arrive before they are sent to the crossbar. A packet is deemed to have arrived only when the last bit has arrived. Consequently, an ij-packet that is in the process of arriving at time t is not included in $V_{ij}(t)$. We say that a VOQ is *active*, whenever the last bit of its first packet has been received. For an active VOQ V_{ij}, we refer to the time period since it last became active as the *current active period*. For a particular active period of V_{ij}, we define notations for several quantities. In particular, if x was the first packet to arrive in the active period, we let $s_{ij} = s(x), f_{ij} = f(x)$. and $L_{ij} = L(x)$. The time of the first input event in the active period is denoted by τ_{ij}. We say an input event is a *backlog event* for V_{ij} if when the event occurs, B_{ij} is too full to accept the first packet in V_{ij}, and we let β_{ij} denote the time of the first backlog event of an active period. We say that V_{ij} is backlogged if it is active, and its most recent input event was a backlog event. These definitions are illustrated in Fig. 2. Note that $\tau_{ij} < f_{ij} + L_M/S$ and that if $\beta_{ij} \neq \tau_{ij}$, then $\beta_{ij} \geq \tau_{ij} + L_{ij}/S$.

While we require that packets be fully buffered at inputs, we assume that packets can be streamed directly though crossbar buffers, and through output buffers to outgoing links. The first assumption is the natural design choice. The second was made to simplify the analysis slightly, but is not essential. Extending our analyses to the case where outputs fully buffer packets is straightforward.

To define a specific crossbar scheduler, we must specify an *input scheduling policy* and an *output scheduling policy*. The input scheduling policy selects an active VOQ from which to transfer a packet to the crossbar. We assume that the input scheduler is defined by an ordering of the active VOQs. At each input scheduling event, the scheduler selects the first active VOQ in this ordering that is not backlogged, and transfers the first packet in this VOQ to the crossbar. We also assume that the output scheduling policy is defined by an ordering imposed on the packets to be forwarded from each output. At each output scheduling event, the scheduler selects the crosspoint buffer whose first packet comes first in this packet ordering.

Given a VOQ ordering for an input, we say that one VOQ *precedes* another if it comes before the other in this VOQ ordering. We extend the precedes relation to the packets in the VOQs and the bits in those packets by ordering the packets (bits) in different VOQs according to the VOQ ordering, and packets (bits) in the same VOQ according to their position in the VOQ. To simplify the language used in the analysis to follow, we include the bits in V_{ij} in the set of bits that are said to precede V_{ij}. For packets (bits) at different inputs going to the same output, we say that one precedes the other, if it comes first in the ordering that defines the output scheduling policy.

For an active VOQ V_{ij}, we let $p_{ij}(t)$ equal the number of bits in VOQs at input i that precede V_{ij} at time t (note, this includes the bits in V_{ij}), plus the number of bits in the current incoming packet that have been received so far (if there is such a packet). We define $q_j(t)$ to be the number of bits at output j at time t and $q_{ij}(t)$ to be the number of bits at output j that precede the last bit in V_{ij}.

With these preliminaries, we can now define two key quantities, *slack* and *margin*. Specifically, we define $slack_{ij}(t) = q_j(t) - p_{ij}(t)$ and $margin_{ij}(t) = q_{ij}(t) - p_{ij}(t)$. In the analysis to follow, we will show that shortly after the start of an active period for V_{ij}, $slack_{ij}$ becomes non-negative and stays non-negative. This is useful, because when an output j becomes idle, q_j is necessarily zero. If $slack_{ij}$ is not negative, then p_{ij} must be zero also. Since this implies that V_{ij} is empty, there can be no packet at input i that should be going out on output j. Consequently, we can show that a scheduler is work-conserving by showing that the *slack* is non-negative. We can use *margin* in a very similar way when showing that a crossbar-based system emulates an output-queued switch with a specific scheduling policy.

Our worst-case performance guarantees are defined relative to a reference system consisting of an ideal output-queued switch followed by a fixed delay of length T. An output-queued switch is one in which packets are transferred directly to output-side queues as soon as they have been completely received. An output-queued switch is fully specified by the queueing discipline used at the outputs.

In [3], the class of *Push in, First Out* (PIFO) queueing disciplines is defined to include all queueing disciplines that can be implemented by inserting arriving packets into a list, and selecting packets for transmission from the front of the list. That is, a PIFO discipline is one in which the relative transmission order of two packets is fixed when the later arriving packet arrives. Most queueing disciplines of practical interest belong to this class. In [4], the *restricted PIFO* queueing disciplines are defined as those PIFO disciplines in which any two ij-packets are transmitted in the same order they were received. Note that this does not restrict the relative transmission order of packets received at different inputs. Our emulation results for buffered crossbars apply to restricted PIFO queueing disciplines.

		S	B/L_M	T/L_M
Packet Group-by-VOQ	T-work conservation	2	2	$2/(S-1)$ (**2** for S=2)
	T-emulation restricted PIFO	2	$3+2/(S-1)$ (**5** for S=2)	$2/(S-1)+1/S$ (**2.5** for S=2)
Packet Least-Occupied Output First	T-work conservation	2	$2S/(S-1)$ (**4** for S=2)	$2/(S-1)$ (**2** for S=2)
	T-emulation restricted PIFO	2	$3S/(S-1)$ (**6** for S=2)	$2/(S-1)+1/S$ (**2.5** for S=2)

Fig. 3. Quantitative results.

We say that a crossbar T-*emulates* an output-queued switch using a specific queueing discipline if, when presented with an input packet sequence, it forwards each packet in the sequence, at the same time that it would be forwarded by the reference system, with an output delay of T. We say that a switch is work-conserving, if whenever there is a packet in the system for output j, output j is sending data. A crossbar-based system is T-*work-conserving* if it T-emulates some work-conserving output-queued switch. Alternatively, we can say that a system is T-work-conserving if output j is busy whenever there is a packet in the system for output j that arrived at least T time units before the current time.

A crossbar that T-emulates an output-queued switch is defined by a specific crossbar scheduling algorithm and by the output queueing discipline of the emulated switch. To achieve the emulation property, the output line cards of the crossbar must hold each packet until T time units have passed since its arrival. While it is being held, other packets that reach the output after it, may be inserted in front of it in the PIFO list. Whenever the output becomes idle, the linecard selects for transmission the first packet in the list which arrived at least T time units in the past. This may not be the first packet in the list, since the PIFO ordering need not be consistent with the arrival order. In the next few sections, we will prove work-conservation and emulation results for two crossbar scheduling algorithms. These results all require that the speedup S, crossbar buffer size B, and time delay T be at least as large as some minimum threshold. Fig. 3 summarizes these thresholds. Note that the values for B and T are stated relative to the maximum packet length L_M.

IV. Common Properties

In this section, we prove a number of properties that apply to certain large classes of crossbar schedulers. Readers may want to skip this section on first reading, referring back to the individual lemmas as they are used in later sections.

A. Prompt Schedulers

All the schedulers we consider have the property that they keep the inputs and outputs busy whenever possible. In particular, if an input line card has any packet x at the head of one of

its VOQs and the VOQ is not backlogged, then the input must be transferring bits to some crosspoint buffer at rate S. Similarly, if any crosspoint buffer for output j is not empty, then output j must be transferring bits from some crosspoint buffer at rate S. A scheduler that satisfies these properties is called a *prompt scheduler*.

The first two lemmas provide lower bounds on q_j that apply to all prompt schedulers. These are useful when attempting to establish lower bounds on $slack_{ij}$.

Lemma 1: For a buffered crossbar using any prompt scheduler, $q_j(t) \geq (1 - 1/S)B_{ij}(t)$ for all i.

Proof: If $B_{ij}(t) > 0$, then B_{ij} became non-empty at some time no later that $t - B_{ij}(t)/S$, since B_{ij} can grow at a rate no faster than S. That is, $B_{ij} > 0$ throughout the interval $[t - B_{ij}(t)/S, t]$. For any prompt scheduler, whenever a crosspoint buffer for a given output is non-empty, the crossbar transfers bits to the output at rate S. Since an output sends bits from the output queue to the link at rate 1, an output queue grows at rate $S - 1$ during any period during which one or more of its crosspoint buffers is non-empty. It follows that $q_j(t) \geq (1-1/S)B_{ij}(t)$. ■

Lemma 2: Consider an active period for V_{ij}. For any prompt scheduler

$$q_j(\tau_{ij}) \geq (1 - 1/S)B_{ij}(\tau_{ij}) + (S - 1)(\tau_{ij} - f_{ij})$$

if $B_{ij}(\tau_{ij}) > 0$.

Proof: Note that since V_{ij} is inactive just before f_{ij}, B_{ij} cannot grow between f_{ij} and τ_{ij}, hence $B_{ij}(f_{ij}) \geq B_{ij}(\tau_{ij}) > 0$. Consequently, q_j must increase at rate $S - 1$ throughout the interval $[f_{ij}, \tau_{ij}]$, so

$$q_j(\tau_{ij}) \geq q_j(f_{ij}) + (S - 1)(\tau_{ij} - f_{ij})$$

By Lemma 1,

$$q_j(f_{ij}) \geq (1 - 1/S)B_{ij}(f_{ij}) \geq (1 - 1/S)B_{ij}(\tau_{ij})$$

Combining the two inequalities yields the desired result. ■

B. Invariant Schedulers

We say that a scheduling algorithm is *invariant* if it does not change the relative order of any two VOQs during a period when they are both continuously active. This property is shared by a number of different crossbar schedulers, including one we consider in detail in the next section.

The next lemma can be used to show that for prompt and invariant schedulers, *slack* does not decrease following the first scheduling event of an active period, and it applies to any prompt and invariant scheduler.

Lemma 3: Let t_1 be the time of an input scheduling event in an active period of V_{ij} and let $t > t_1$ be no later than the next event at input i. For any prompt and invariant scheduler,

$$slack_{ij}(t) \geq slack_{ij}(t_1) + (S - 2)(t - t_1)$$

if $B \geq 2L_M$.

Proof: If V_{ij} is backlogged at time t_1, then $B_{ij}(t_1) > L_M$ which implies that B_{ij} remains non-empty until at least $t_1 + L_M/S \geq t$. Consequently, $q_j(t) \geq q_j(t_1) + (S-1)(t-t_1)$. Since the VOQ ordering is invariant in the interval $[t_1, t]$, any increase in p_{ij} during this interval can only result from the arrival of bits on the input link. Consequently, $p_{ij}(t) \leq p_{ij}(t_1) + (t - t_1)$ and $slack_{ij}(t) \geq slack_{ij}(t_1) + (S - 2)(t - t_1)$.

If V_{ij} is not backlogged at t_1, then either V_{ij} or another VOQ that precedes V_{ij} must be selected at t_1. In either case, $p_{ij}(t) \leq p_{ij}(t_1) - (S - 1)(t - t_1)$. Since $q_j(t) \geq q_j(t_1) - (t - t_1)$, it follows that $slack_{ij}(t) \geq slack_{ij}(t_1) + (S - 2)(t - t_1)$. ■

We can prove a stronger version of Lemma 3 that can be used to obtain more precise results.

Lemma 4: Let t_1 be the time of an input scheduling event in an active period of V_{ij} and let $t > t_1$ be no later than the next event at input i. For any prompt and invariant scheduler with $S \geq 2$ and $B \geq 2L_M$, if $slack_{ij}(t_1) \geq -V_{ij}(t_1)$ then $slack_{ij}(t) \geq -V_{ij}(t)$.

Proof: If $V_{ij}(t) \geq V_{ij}(t_1)$ then the result follows from Lemma 3. Assume then that $V_{ij}(t) < V_{ij}(t_1)$. This implies that V_{ij} was selected at t_1. Consequently, q_j increases at rate $S - 1$ during the interval $[t_1, t]$ (since the scheduling algorithm is prompt), while p_{ij} decreases at rate $\geq S - 1$ (since the scheduling algorithm is invariant). Thus, $slack_{ij}(t) \geq slack_{ij}(t_1) + 2(S - 1)(t - t_1)$. Since V_{ij} can decrease at a rate no faster than $S, V_{ij}(t) \geq V_{ij}(t_1) - S(t - t_1)$. Consequently,

$$\begin{aligned} slack_{ij}(t) &\geq slack_{ij}(t_1) + 2(S - 1)(t - t_1) \\ &\geq -V_{ij}(t_1) + 2(S - 1)(t - t_1) \\ &\geq -(V_{ij}(t) + S(t - t_1)) + 2(S - 1)(t - t_1) \\ &= -V_{ij}(t) + (S - 2)(t - t_1) \\ &\geq -V_{ij}(t) \end{aligned}$$

since $S \geq 2$. ■

C. ij-FIFO Schedulers

We say that a system is ij-*FIFO* if for all inputs i and outputs j, all ij-packets are forwarded in the same order they were received. Note that systems that implement restricted PIFO queueing disciplines are ij-FIFO.

In this subsection, we prove several lemmas that are useful in proving emulation results. The first two lemmas provide lower bounds on q_{ij} for prompt and ij-FIFO schedulers. These are useful for proving lower bounds on $margin_{ij}$.

Lemma 5: For any prompt and ij-FIFO scheduler, $q_{ij}(t) \geq (1 - 1/S)(B_{ij}(t) - L_M)$.

Proof: The statement is trivially true if $B_{ij}(t) \leq L_M$. So assume, $B_{ij}(t) > L_M$, and note that this implies that B_{ij} became non-empty at some time no later that $t - B_{ij}(t)/S$, since B_{ij} can grow at a rate no faster than S. Consequently, there must be a scheduling event at output j in the interval $[t - B_{ij}(t)/S, (t - B_{ij}(t)/S) + L_M/S]$ and from the time of that event until t, output j must be receiving bits that precede V_{ij}, since the scheduler is ij-FIFO. Consequently, q_{ij} increases at rate $S-1$ throughout the interval $[(t-B_{ij}(t)/S)+L_M/S, t]$ and so $q_{ij}(t) \geq (1 - 1/S)(B_{ij}(t) - L_M)$. ■

Lemma 6: Consider an active period for V_{ij}. For any prompt and ij-FIFO scheduler,

$$q_{ij}(\tau_{ij}) \geq (1 - 1/S)(B_{ij}(\tau_{ij}) - L_M) + (S - 1)(\tau_{ij} - f_{ij})$$

if $B_{ij}(\tau_{ij}) \geq L_M$.

Proof: Since B_{ij} cannot grow between f_{ij} and $\tau_{ij}, B_{ij}(f_{ij}) \geq B_{ij}(\tau_{ij}) \geq L_M$. Consequently, B_{ij} became non-empty no later than $f_{ij} - L_M/S$, which implies that q_{ij} increases at rate $S-1$ throughout the interval $[f_{ij}, \tau_{ij}]$. Hence,

$$\begin{aligned} q_{ij}(\tau_{ij}) &\geq q_{ij}(f_{ij}) + (S-1)(\tau_{ij} - f_{ij}) \\ &\geq (1-1/S)(B_{ij}(f_{ij}) - L_M) + (S-1)(\tau_{ij} - f_{ij}) \\ &\geq (1-1/S)(B_{ij}(\tau_{ij}) - L_M) + (S-1)(\tau_{ij} - f_{ij}) \end{aligned}$$

■

The next lemma can be used to show that $margin$ does not decrease after the first event of an active period. It applies to any scheduler that is prompt, invariant and ij-FIFO.

Lemma 7: Let t_1 be the time of an input scheduling event in an active period of V_{ij} and let $t > t_1$ be no later than the next event at input i. For any prompt, invariant and ij-FIFO scheduler with speedup S and $B \geq 2L_M, margin_{ij}(t) \geq margin_{ij}(t_1) + (S-2)(t-t_1)$.

Proof: If V_{ij} is backlogged at t_1, then $B_{ij}(t_1) > L_M$ and B_{ij} became non-empty before $t_1 - L_M/S$ and will remain non-empty until at least $t_1 + L_M/S$. This implies that q_{ij} increases at rate $S-1$ throughout the interval $[t_1, t]$ (since the scheduler is ij-FIFO). Since p_{ij} can increase at a rate no faster than 1 during this period, $margin_{ij}(t) \geq margin_{ij}(t_1) + (S-2)(t-t_1)$. If V_{ij} is not backlogged at t_1, p_{ij} decreases at rate $\geq S-1$ in the interval $[t_1, t]$ (since the scheduler is invariant) and since q_{ij} can decrease at a rate no faster than 1, $margin_{ij}(t) \geq margin_{ij}(t_1) + (S-2)(t-t_1)$. ■

V. Packet Group by VOQ

Group by Virtual Output Queue (GVOQ) is a cell switch scheduling algorithm first described in [3] and extended to buffered crossbars in [4]. We define the Packet GVOQ (PGV) scheduler by defining an ordering that it imposes on the VOQs. In this ordering, the relative order of two VOQs does not change so long as they both remain active. Hence, PGV is invariant. When an inactive VOQ becomes active, it is placed first in the VOQ ordering. When a VOQ becomes inactive, it is removed from the VOQ ordering. Different variants of PGV can be defined by specifiying different output scheduling strategies.

A. T-Work-Conservation

In this section, we show that regardless of the specific output scheduling policy used, PGV is T-work-conserving. We prove two versions of the work-conservation result. The first is a bit weaker than the second, but is included because the analysis is more straightforward and hence it provides a useful stepping stone to the more difficult results to follow.

Theorem 1: Any PGV scheduler is T-work-conserving if $S \geq 2, B \geq (2 + 1/(S-1))L_M$ and $T \geq 2L_M/(S-1)$.

The proof of this theorem involves four steps. The first step is to show that $slack$ does not decrease after the first scheduling event of an active period. This was shown in Lemma 3 in the previous section. The second step, is to show that a backlog event must occur near the start of an active period, and the third step is to show that when the first backlog event occurs, $slack$ is non-negative. These two steps are shown in the proofs of the next two lemmas. The final step, which appears as the proof of the theorem, is to show that when an output is idle, no input can have a packet that has been present for more than time T.

Lemma 8: Consider an active period for V_{ij} in a crossbar using a PGV scheduler with speedup S. If the duration of the active period is at least $2L_M/(S-1)$, then it includes at least one backlog event for V_{ij} and $\beta_{ij} \leq f_{ij} + 2L_M/(S-1)$.

Proof: Suppose there is no backlog event in the interval $[\tau_{ij}, t]$ for $t = f_{ij} + 2L_M/(S-1)$. Then, at each event in this interval, the input scheduler selects either V_{ij} or some other VOQ that precedes V_{ij}. Since the scheduling algorithm is invariant, any contribution to increasing p_{ij} during this interval can only result from the arrival of new bits from the input link. Consequently, p_{ij} decreases at a rate $\geq (S-1)$ throughout this period. Since $p_{ij}(\tau_{ij}) \leq L_{ij} + (\tau_{ij} - f_{ij})$,

$$\begin{aligned} p_{ij}(t) &\leq L_{ij} + (\tau_{ij} - f_{ij}) - (S-1)(t - \tau_{ij}) \\ &= L_{ij} + (\tau_{ij} - f_{ij}) \\ &\quad - (S-1)((f_{ij} + 2L_M/(S-1)) - \tau_{ij}) \\ &= L_{ij} + S(\tau_{ij} - f_{ij}) - 2L_M \\ &< L_M + S(L_M/S) - 2L_M = 0. \end{aligned}$$

The first line in the above inequality follows from the fact that $p_{ij}(f_{ij}) = L_{ij}$ and that p_{ij} can increase at rate at most 1 during the interval $[s_{ij}, \tau_{ij}]$ and must decrease at rate $S-1$ after τ_{ij}. The second and third lines follow directly from the definitions, and the last line from the fact that $\tau_{ij} - f_{ij} < L_M/S$. The above result contradicts the premise that the duration of the active period is at least $2L_M/(S-1)$. ■

Our next lemma shows that within a short time following the start of an active period, $slack_{ij} \geq 0$.

Lemma 9: Consider some active period for V_{ij} that includes the time $t \geq f_{ij} + 2L_M/(S-1)$. For any PGV scheduler, $slack_{ij}(t) > 0$ if $S \geq 2$ and $B \geq (2 + 1/(S-1))L_M$.

Proof: We show that $slack_{ij}(\beta_{ij}) > 0$. The result then follows from Lemmas 3 and 8.

If $\beta_{ij} = \tau_{ij}$, then by Lemma 2,

$$q_j(\beta_{ij}) > (1-1/S)(B - L_M) + (S-1)(\tau_{ij} - f_{ij}).$$

For any PGV scheduler,

$$p_{ij}(\beta_{ij}) = p_{ij}(\tau_{ij}) \leq L_M + (\tau_{ij} - f_{ij}).$$

Combining the inequalities for p_{ij} and q_j, we obtain

$$slack_{ij}(\beta_{ij}) > (1-1/S)(B-L_M) + (S-2)(\tau_{ij} - f_{ij}) - L_M \geq 0$$

since $S \geq 2$ and $B \geq (2 + 1/(S-1))L_M$.

Now, suppose $\beta_{ij} > \tau_{ij}$. Since at least one packet must be sent from V_{ij} during the active period, in order for it to become backlogged, $\beta_{ij} \geq \tau_{ij} + L_{ij}/S$. During the interval $[\tau_{ij}, \beta_{ij}], p_{ij}$ decreases at rate $\geq S-1$. Consequently,

$$\begin{aligned} p_{ij}(\beta_{ij}) &\leq L_{ij} + (\tau_{ij} - f_{ij}) - (S-1)(\beta_{ij} - \tau_{ij}) \\ &< L_{ij}/S + L_M/S \leq 2L_M/S. \end{aligned}$$

By Lemma 1, $q_j(\beta_{ij}) > (1-1/S)(B-L_M) \geq 2(1-1/S)L_M$, so $slack_{ij}(\beta_{ij}) > 2(1-1/S)L_M - 2L_M/S \geq 0$. ■

We can now proceed to the proof of the theorem.

Proof of Theorem 1: Suppose some output j is idle at time t and no input is currently sending it a packet, but some input i has a packet x for output j with $f(x) + 2L_M/(S-1) < t$. By Lemma 9, $slack_{ij}(t) > 0$. Since, $q_j(t) = 0$, this implies that $p_{ij}(t) < 0$, which contradicts the fact that V_{ij} is active at t. ∎

Using a more precise analysis, we can reduce the required crossbar buffer size to $2L_M/(S-1)$.

Theorem 2: Any PGV scheduler with $S \geq 2$ and $B \geq 2L_M$ is T-work-conserving for $T \geq 2L_M/(S-1)$.

To prove this, we must first show that $slack$ is bounded from below, shortly after the start of an active period.

Lemma 10: Consider an active period for V_{ij} that includes the time $t \geq f_{ij} + 2L_M/(S-1)$. For any PGV scheduler with speedup $S \geq 2$ and $B \geq 2L_M$, $slack_{ij}(t) > -V_{ij}(t)$.

Proof: If $B_{ij}(\tau_{ij}) > 0$ then by Lemma 2, $q_j(\tau_{ij}) > \tau_{ij} - f_{ij}$. Since

$$p_{ij}(\tau_{ij}) \leq L_{ij} + (\tau_{ij} - f_{ij}) \leq V_{ij}(\tau_{ij}) + (\tau_{ij} - f_{ij})$$

it follows that $slack_{ij}(\tau_{ij}) > -V_{ij}(\tau_{ij})$. By Lemma 4, $slack_{ij}(t) > -V_{ij}(t)$.

Now, suppose that $B_{ij}(\tau_{ij}) = 0$. By Lemma 8, $\beta_{ij} \leq f_{ij} + 2L_M/(S-1) \leq t$. If x is the first packet in V_{ij} at β_{ij}, then $\beta_{ij} > \tau_{ij} + (B - L(x))/S$. During the interval $[\tau_{ij}, \beta_{ij}]$, p_{ij} decreases at rate $\geq S-1$. Consequently,

$$\begin{aligned} p_{ij}(\beta_{ij}) &< L_{ij} + (\tau_{ij} - f_{ij}) - (S-1)(\beta_{ij} - \tau_{ij}) \\ &\leq (1 + 1/S)L_M - (1 - 1/S)(B - L(x)). \end{aligned}$$

By Lemma 1, $q_j(\beta_{ij}) > (1 - 1/S)(B - L(x))$ so,

$$\begin{aligned} slack_{ij}(\beta_{ij}) &> 2(1 - 1/S)(B - L(x)) - (1 + 1/S)L_M \\ &\geq 4(1 - 1/S)L_M - (1 + 1/S)L_M \\ &\quad - 2(1 - 1/S)L(x) \\ &= (3 - 5/S)L_M - (1 - 2/S)L(x) - L(x) \\ &> -L(x) \geq -V_{ij}(\beta_{ij}). \end{aligned}$$

By Lemma 4, $slack_{ij}(t) > -V_{ij}(t)$. ∎

We can now proceed to prove the theorem.

Proof of Theorem 2: Suppose some output j is idle at time t and no input is currently sending it a packet, but some input i has a packet x for output j with $f(x) + 2L_M/(S-1) < t$. By Lemma 10, $slack_{ij}(t) > -V_{ij}(t)$. Since, $q_j(t) = 0$, this implies that $p_{ij}(t) < V_{ij}(t)$, which contradicts the definition of p_{ij}. ∎

B. T-Emulation Results for PGV

We refer to a PGV scheduler defined by a restricted PIFO queueing discipline as a PGV-RP sechduler. We show that for any restricted PIFO queueing discipline, the corresponding PGV-RP scheduler T-emulates an ideal output-queued switch using the same discipline. Our result for PGV generalizes the corresponding result for cell-based crossbars given in [3].

Theorem 3: Let X be an output-queued switch using a restricted PIFO scheduler. A crossbar using the corresponding PGV-RP scheduler T-emulates X if $S \geq 2, B \geq (3 + 2/(S-1))L_M$, and $T \geq (2/(S-1) + 1/S)L_M$.

The analysis leading to this result is similar to the analysis used to establish work-conservation. The first step is to show that $margin$ does not decrease following the first input event of an active period (Lemma 7). The second step is to establish a lower bound on $margin$ at the time of the first backlog event. We then use this lower bound to prove the emulation result.

Lemma 11: Consider an active period for V_{ij} that includes $t \geq f_{ij} + 2L_M/(S-1)$. For any PGV-RP scheduler, $margin_{ij}(t) > L_M/S$ if $S \geq 2$ and $B \geq (3 + 2/(S-1))L_M$.

Proof: We show that $margin_{ij}$ satisfies the bound at the time of the first backlog event. If V_{ij} is backlogged at τ_{ij}, then $B_{ij}(\tau_{ij}) > B - L_M$ and by Lemma 6,

$$q_{ij}(\tau_{ij}) > (1 - 1/S)(B - 2L_M) + (S-1)(\tau_{ij} - f_{ij})$$

Since $p_{ij}(\tau_{ij}) \leq L_M + (\tau_{ij} - f_{ij})$,

$$\begin{aligned} margin_{ij}(\tau_{ij}) &> (1 - 1/S)(B - 2L_M) \\ &\quad + (S-1)(\tau_{ij} - f_{ij}) \\ &\quad - (L_M + (\tau_{ij} - f_{ij})) \\ &\geq (1 - 1/S)B - (3 - 2/S)L_M. \end{aligned}$$

This is $\geq L_M/S$ so long as $B \geq (3 + 2/(S-1))L_M$. By Lemma 7, $margin_{ij}(t) > L_M/S$.

Now suppose V_{ij} is not backlogged at τ_{ij}. By Lemma 8, $\beta_{ij} \leq t$ and since V_{ij} is backlogged at β_{ij}, $B_{ij}(\beta_{ij}) > B - L_M$, so by Lemma 5, $q_{ij}(\beta_{ij}) > (1 - 1/S)(B - 2L_M)$. Since, $\beta_{ij} \geq \tau_{ij} + L_{ij}/S$, it follows that

$$p_{ij}(\beta_{ij}) \leq L_{ij} + L_M/S - (1 - 1/S)L_{ij} \leq 2L_M/S$$

and

$$\begin{aligned} margin_{ij}(\beta_{ij}) &> (1 - 1/S)(B - 2L_M) - 2L_M/S \\ &= (1 - 1/S)B - 2L_M. \end{aligned}$$

This is $\geq L_M/S$ so long as $B \geq ((2 + 3/(S-1))L_M$ which is implied by the condition on B in the statement of the lemma. By Lemma 7 $margin_{ij}(t) > L_M/S$. ∎

We can now proceed to the proof of the theorem.

Proof of Theorem 3: Suppose that up until time t, the PGV-RP crossbar faithfully emulates the output-queued switch, but that at time t, the output-queued switch begins to forward an ij-packet x, while the crossbar does not.

Now suppose that in the crossbar, one or more bits of x have reached B_{ij} by time $t - L_M/S$. Note that the interval $[t - L_M/S, t)$ must contain at least one scheduling event at output j and all such events must select packets that precede x. However, this implies that during some non-zero time interval $[t_1, t]$, output j is continuously receiving bits that precede x at a faster rate than it can forward them to the output. This contradicts that fact that by time t, the crossbar has forwarded all the bits that precede x (since it faithfully emulates the output-queued switch up until time t).

Assume then that at time $t - L_M/S$, no bits of x have reached B_{ij}. Since the output queued switch has an output delay of T, $f(x) \leq t - T$, so $t - L_M/S \geq f(x) + 2L_M/(S-1)$. Since the crossbar has sent everything sent by the output-queued

switch up until t, it follows that $q_{ij}(t - L_M/S) \leq L_M/S$. By Lemma 11, $margin_{ij}(t - L_M/S) > L_M/S$ and hence $p_{ij}(t - L_M/S) < 0$, which is not possible. ■

The analysis of Lemma 11 requires a crossbar buffer of size at least $5L_M$ when $S = 2$. We conjecture that this can be reduced using a more sophisticated analysis.

VI. Packet LOOFA

The Least Occupied Output First Algorithm (LOOFA) is a cell scheduling algorithm described in [8]. We define an asynchronous crossbar scheduling algorithm based on LOOFA, called Packet LOOFA (PLF). Like PGV, PLF is defined by the ordering it imposes on the VOQs at each input. The ordering of the VOQs is determined by the number of bits in the output queues. In particular, when a VOQ V_{ij} becomes active, it is inserted immediately after the last VOQ V_{ih}, for which $q_h \leq q_j$. If there is no such VOQ, it is placed first in the ordering. At any time, active VOQs may be re-ordered, based on the output occupancy. We allow one VOQ to move ahead of another during this re-ordering, only if its output has strictly fewer bits. The work-conservation result for PLF is comparable to that for PGV, but the required analysis is technically more difficult because in PLF, the relative orders of VOQs can change.

Because the order of VOQs can change, PLF is also more responsive to changing traffic conditions than PGV. While this has no effect on work-conservation when $S \geq 2$, it does provide better fairness when used with smaller speedups. As one example of this, consider the following traffic pattern. From time 0 to time T, a switch with speedup of $S < 4/3$, receives packets on inputs A and B for output X at the link rate of 1. After time T, input A receives packets for output Y (at rate 1) while input B receives packets for output Z. Due to the symmetry of the traffic pattern, a scheduler has no reason to favor one input over the other, so we assume that the inputs are treated fairly by the output scheduling policy. Up until time T, the two inputs each send packets to X at rate $S/2$ and X forwards packets at rate 1, while building a backlog. If a PGV scheduler is used, then after time T, input A gives preference to output Y, while input B gives preference to output Z. Consequently, output X receives packets only at rate $2(S-1)$. As a result, the output side backlog at X is fully consumed by time $((2 - S)/(3 - 2S))T$, after which X starts forwarding packets at rate $2(S - 1)$, while both outputs Y and Z continue to forward packets at rate 1. So for $S = 1.2$, X is limited to an output rate of 0.4. On the other hand, a PLF scheduler attempts to keep the output queue lengths equal, so after time $((2-S)/(3-2S))T$, outputs X, Y and Z will all receive packets at rate $2S/3$. So for $S = 1.2$, all three outputs will forward packets at rate 0.8. This doubles the rate at which X is able to send, dramatically improving the fairness with respect to the other outputs.

A. More Definitions

To facilitate the analysis of PLF, it's helpful to separate the analysis of "old bits" from "new bits". When considering an active period for V_{ij}, the *old bits* at input i are those bits that arrived before s_{ij}. All other bits at input i are considered *new*. Also, we say that a VOQ V is *older* than a VOQ W at time t if both are active, and V last became active before W did. We say that a VOQ V *passes* a VOQ W during a given time interval, if W precedes V at the start of the interval and V precedes W at the end of the interval.

For an active VOQ, V_{ij}, we let $new_{ij}(t)$ be the number of bits present at input i at time t that arrived in the interval $[s_{ij}, t]$. We let $\bar{p}_{ij}(t)$ equal $new_{ij}(t)$ plus the number of bits that precede V_{ij} at time t that arrived before s_{ij}. Note that $p_{ij}(t) \leq \bar{p}_{ij}(t)$ and consequently,

$$slack_{ij}(t) \geq \underline{slack}_{ij}(t) = q_j(t) - \bar{p}_{ij}(t)$$

and

$$margin_{ij}(t) \geq \underline{margin}_{ij}(t) = q_{ij}(t) - \bar{p}_{ij}(t).$$

B. Additional General Lemmas

Here we give several more lemmas that apply to a broad class of scheduling algorithms and are useful for establishing both T-work-conservation and T-emulation results for PLF. The reader may want to skip this section on first reading, and refer back to the lemmas presented here, as they are used. The proofs of the first two lemmas are omitted, as they are very similar to proofs of earlier lemmas.

Lemma 12: Let t_1 be the time of an input scheduling event in an active period of V_{ij} and let $t > t_1$ be no later than the next event at input i. For any prompt scheduler,

$$\underline{slack}_{ij}(t) \geq \underline{slack}_{ij}(t_1) + (S - 2)(t - t_1)$$

if no older VOQ passes V_{ij} in $[t_1, t]$ and $B \geq 2L_M$.

Lemma 13: Let t_1 be the time of an input scheduling event in an active period of V_{ij} and let $t > t_1$ be no later than the next event at input i. For any prompt, ij-FIFO scheduler,

$$\underline{margin}_{ij}(t) \geq \underline{margin}_{ij}(t_1) + (S - 2)(t - t_1)$$

if no older VOQ passes V_{ij} in $[t_1, t]$ and $B \geq 2L_M$.

Our next lemma applies to any scheduling algorithm.

Lemma 14: If there is some VOQ that is older than V_{ij} and that precedes V_{ij} at time t, then there is some such VOQ V_{ih} for which $\bar{p}_{ij}(t) \leq \bar{p}_{ih}(t)$.

Proof: Let V_{ih} be a VOQ that is older than V_{ij} and that precedes V_{ij} at time t. More specifically, let V_{ih} be that VOQ that comes latest in the VOQ ordering, among all VOQs that satisfy the condition. Let X be the set of bits that precede V_{ij} at time t but not V_{ih}. Note that $|X| = p_{ij}(t) - p_{ih}(t)$ and that all bits in X must have arrived since s_{ij} (otherwise, there would be some VOQ older than V_{ij} that precedes V_{ij} and comes later in the VOQ ordering than V_{ih}). Since V_{ih} is older than V_{ij}, these bits also arrived after s_{ih}. Let Y be the set of bits that arrived after s_{ij} and are still present at time t and do not precede V_{ij}. Note that $|Y| = \bar{p}_{ij}(t) - p_{ij}(t)$ and that X and Y have no bits in common. Now, let Z be the set of bits that arrived since s_{ih} and do not precede V_{ih}. Both X and Y are subsets of Z and so $|X| + |Y| \leq |Z| = \bar{p}_{ih}(t) - p_{ih}(t)$. Consequently,

$$(p_{ij}(t) - p_{ih}(t)) + (\bar{p}_{ij}(t) - p_{ij}(t)) \leq \bar{p}_{ih}(t) - p_{ih}(t)$$

which implies that $\bar{p}_{ij}(t) \leq \bar{p}_{ih}(t)$. ■

C. T-Work-Conservation

Theorem 4: A buffered crossbar using any PLF scheduler is T-work-conserving if $S \geq 2, B \geq 2L_M S/(S-1)$, and $T \geq 2L_M/(S-1)$.

To prove the theorem, we need the following lemma.

Lemma 15: If V_{ij} is active at $t \geq \tau_{ij}$, then for any PLF scheduler, either

$$\underline{slack}_{ij}(t) \geq 0$$

or

$$\bar{p}_{ij}(t) \leq (1+1/S)L_M - (S-1)(t-\tau_{ij})$$

if $S \geq 2$ and $B \geq 2L_M S/(S-1)$.

Before we proceed with the proof of the lemma, we note that for $t \geq f_{ij} + 2L_M/(S-1)$,

$$\begin{aligned}(1+1/S)L_M &- (S-1)(t-\tau_{ij}) \\ &\leq (1+1/S)L_M + (S-1)(\tau_{ij}-f_{ij}) - 2L_M \\ &\leq (1+1/S)L_M + (1-1/S)L_M - 2L_M \leq 0.\end{aligned}$$

Consequently, the lemma implies that $\underline{slack}_{ij}(t) \geq 0$ for $t \geq f_{ij}+2L_M/(S-1)$. Since $slack_{ij}(t) \geq \underline{slack}_{ij}(t)$, $slack_{ij}(t) \geq 0$ also. We state this as a corollary.

Corollary 1: If V_{ij} is active at $t \geq f_{ij} + 2L_M/(S-1)$, then for any PLF scheduler, $slack_{ij}(t) \geq 0$, if $S \geq 2$ and $B \geq 2L_M S/(S-1)$.

Proof of Lemma 15: Assume that there is some time t when the lemma does not hold. More specifically, let t be the earliest time when it is not true for some VOQ and let V_{ij} be the oldest VOQ that violates the lemma at time t.

Suppose first there is no event in $[\tau_{ij}, t]$ at which there is an older VOQ that precedes V_{ij}. This implies that

$$\bar{p}_{ij}(\tau_{ij}) \leq \tau_{ij} - s_{ij} \leq (1+1/S)L_M.$$

It also implies that there are no two consecutive events in $[\tau_{ij}, t]$ between which an older VOQ passes V_{ij}. Consequently, by Lemma 12, $\underline{slack}_{ij}$ does not decrease between any two consecutive events in $[\tau_{ij}, t]$.

If $\beta_{ij} > t$ then V_{ij} is eligible for selection at every event in $[\tau_{ij}, t]$. This implies that at every such event, the selected packet precedes V_{ij}. Consequently,

$$\begin{aligned}\bar{p}_{ij}(t) &\leq \bar{p}_{ij}(\tau_{ij}) - (S-1)(t-\tau_{ij}) \\ &\leq (1+1/S)L_M - (S-1)(t-\tau_{ij})\end{aligned}$$

which contradicts our assumption that the lemma does not hold at t. Assume then that $\beta_{ij} \leq t$. By Lemma 1,

$$q_j(\beta_{ij}) \geq (1-1/S)(B-L_M)$$

and since

$$\bar{p}_{ij}(\beta_{ij}) \leq \bar{p}_{ij}(\tau_{ij}) - (S-1)(\beta_{ij}-\tau_{ij})$$

it follows that

$$\begin{aligned}\underline{slack}_{ij}(\beta_{ij}) &\geq (1-1/S)(B-L_M) \\ &\quad - ((1+1/S)L_M - (S-1)(\beta_{ij}-\tau_{ij})) \\ &\geq (1-1/S)B - 2L_M\end{aligned}$$

which is ≥ 0 for $B \geq 2(S/(S-1))L_M$. Since $\underline{slack}_{ij}$ does not decrease in $[\tau_{ij}, t]$, it follows that $\underline{slack}_{ij}(t) \geq 0$. This again, contradicts our assumption that the lemma does not hold at t.

From the above, it follows that there must be some event in $[\tau_{ij}, t]$ at which there is an older VOQ that precedes V_{ij}. Let t_1 be the time of the latest such event. By Lemma 14, there is a VOQ V_{ih} for which $\bar{p}_{ij}(t_1) \leq \bar{p}_{ih}(t_1)$ and since V_{ih} precedes V_{ij} at $t_1, q_j(t_1) \geq q_h(t_1)$ and consequently, $\underline{slack}_{ij}(t_1) \geq \underline{slack}_{ih}(t_1)$. If $t_1 = t$, then since V_{ij} is the oldest VOQ that does not satisfy the lemma at t, V_{ih} does satisfy the lemma. That is,

$$\underline{slack}_{ih}(t) \geq 0$$

or

$$\bar{p}_{ih}(t) \leq (1+1/S)L_M - (S-1)(t-\tau_{ih}).$$

If $\underline{slack}_{ih}(t) \geq 0$, then $\underline{slack}_{ij}(t) \geq 0$ also. On the other hand, if $\bar{p}_{ih}(t) \leq (1+1/S)L_M - (S-1)(t-\tau_{ih})$, then

$$\begin{aligned}\bar{p}_{ij}(t) &\leq (1+1/S)L_M - (S-1)(t-\tau_{ih}) \\ &\leq (1+1/S)L_M - (S-1)(t-\tau_{ij}).\end{aligned}$$

Once again, this contradicts our assumption that the lemma does not hold at t. Consequently, we must have $t_1 < t$ and since t is the earliest time at which the lemma is violated, either

$$\underline{slack}_{ij}(t_1) \geq 0$$

or

$$\bar{p}_{ij}(t_1) \leq (1+1/S)L_M - (S-1)(t_1-\tau_{ij}).$$

If $\underline{slack}_{ij}(t_1) \geq 0$ then by Lemma 12, $\underline{slack}_{ij}(t) \geq 0$ also. Assume then, that $\bar{p}_{ij}(t_1) \leq (1+1/S)L_M - (S-1)(t_1-\tau_{ij})$. Now, if $\beta_{ij} \geq t$, then V_{ij} is eligible for selection at every event in $[t_1, t]$ and so $\bar{p}_{ij}(t) \leq (1+1/S)L_M - (S-1)(t-\tau_{ij})$. On the other hand, if $t_1 \leq \beta_{ij} < t$, it follows that

$$\bar{p}_{ij}(\beta_{ij}) \leq (1+1/S)L_M - (S-1)(\beta_{ij}-\tau_{ij})$$

and since $q_j(\beta_{ij}) \geq (1-1/S)(B-L_M)$,

$$\begin{aligned}\underline{slack}_{ij}(\beta_{ij}) &\geq (1-1/S)(B-L_M) \\ &\quad - ((1+1/S)L_M - (S-1)(\beta_{ij}-\tau_{ij})) \\ &\geq (1-1/S)B - 2L_M \geq 0.\end{aligned}$$

Since $\underline{slack}_{ij}$ cannot decrease in $[t_1, t]$, it follows that $\underline{slack}_{ij}(t) \geq 0$. Once again, this contradicts the assumption that the lemma does not hold at t.

That leaves one more case: $\beta_{ij} < t_1$. Since $\underline{slack}_{ij}(t_1) \geq 0$ implies that $\underline{slack}_{ij}(t) \geq 0$, we must have

$$\bar{p}_{ij}(t_1) \leq (1+1/S)L_M - (S-1)(t_1-\tau_{ij}).$$

Also, since $q_j(\beta_{ij}) \geq (1-1/S)(B-L_M)$,

$$q_j(t_1) \geq (1-1/S)(B-L_M) - (t_1 - \beta_{ij})$$

and

$$\begin{aligned}\underline{slack}_{ij}(t_1) &\geq (1-1/S)(B-L_M) - (t_1-\beta_{ij}) \\ &\quad - ((1+1/S)L_M - (S-1)(t_1-\tau_{ij})) \\ &\geq (1-1/S)B + (S-2)(t_1-\tau_{ij}) - 2L_M \\ &\geq (1-1/S)B - 2L_M \geq 0.\end{aligned}$$

This completes the contradiction to our original assumption that the lemma does not hold at time t. ∎

Proof of Theorem 4: Suppose some output j is idle at time t, but some input i has a packet x for output j with $f(x)+T < t$. By Corollary 1, $slack_{ij}(t) \geq 0$. Since $q_j(t) = 0$, this implies that $p_{ij}(t) \leq 0$, which contradicts the fact that V_{ij} contains x at t. ∎

D. T-Emulation

In this section we show that a variant of the PLF algorithm is capable of emulating an output queued switch using any restricted PIFO queueing discipline. This variant differs from the standard PLF algorithm in that it orders VOQs based on the values of q_{ij}, rather than q_j. That is, when V_{ij} becomes nonempty, it is inserted into the VOQ ordering after the last VOQ V_{ih} for which $q_{ih} \leq q_{ij}$. If there is no such VOQ, V_{ij} is placed first in the ordering. Strictly speaking, this variant is different from PLF, so to avoid confusion we refer to it as *Refined PLF* or RPLF.

Theorem 5: Let X be an output-queued switch using a restricted PIFO scheduler. A crossbar using the corresponding RPLF scheduler T-emulates X if $S \geq 2$ and $B \geq 3L_MS/(S-1)$ and $T \geq (2/(S-1)+1/S)L_M$.

To prove the theorem, we need the following lemma.

Lemma 16: If V_{ij} is active at $t \geq \tau_{ij}$, then for any RPLF scheduler, either

$$\underline{margin}_{ij}(t) \geq L_M/S$$

or

$$\bar{p}_{ij}(t) \leq (1+1/S)L_M - (S-1)(t-\tau_{ij})$$

if $S \geq 2$ and $B \geq 3L_MS/(S-1)$.

Proof: Assume that there is some time t when the lemma does not hold. More specifically, let t be the earliest time when it is not true for some VOQ and let V_{ij} be the oldest VOQ that violates the lemma at time t.

Suppose first there is no event in $[\tau_{ij}, t]$ at which there is an older VOQ that precedes V_{ij}. This implies that

$$\bar{p}_{ij}(\tau_{ij}) \leq \tau_{ij} - s_{ij} \leq (1+1/S)L_M.$$

It also implies that there are no two consecutive events in $[\tau_{ij}, t]$ between which an older VOQ passes V_{ij}. Consequently, by Lemma 13, $\underline{margin}_{ij}$ does not decrease between any two consecutive events in $[\tau_{ij}, t]$.

If $\beta_{ij} > t$ then V_{ij} is eligible for selection at every event in $[\tau_{ij}, t]$. This implies that at every such event, the selected packet precedes V_{ij}. Consequently,

$$\begin{aligned}\bar{p}_{ij}(t) &\leq \bar{p}_{ij}(\tau_{ij}) - (S-1)(t-\tau_{ij}) \\ &\leq (1+1/S)L_M - (S-1)(t-\tau_{ij})\end{aligned}$$

which contradicts our assumption that the lemma does not hold at t. Assume then that $\beta_{ij} \leq t$. By Lemma 5,

$$q_{ij}(\beta_{ij}) \geq (1-1/S)(B-2L_M)$$

and since

$$\bar{p}_{ij}(\beta_{ij}) \leq \bar{p}_{ij}(\tau_{ij}) - (S-1)(\beta_{ij}-\tau_{ij})$$

it follows that

$$\begin{aligned}\underline{margin}_{ij}(\beta_{ij}) &\geq (1-1/S)(B-2L_M) \\ &\quad - ((1+1/S)L_M - (S-1)(\beta_{ij}-\tau_{ij})) \\ &\geq (1-1/S)B - (3-1/S)L_M\end{aligned}$$

which is $\geq L_M/S$ for $B \geq 3L_MS/(S-1)$. Since $\underline{margin}_{ij}$ does not decrease in $[\tau_{ij}, t]$, it follows that $\underline{margin}_{ij}(t) \geq L_M/S$. This again, contradicts our assumption that the lemma does not hold at t.

From the above, it follows that there must be some event in $[\tau_{ij}, t]$ at which there is an older VOQ that precedes V_{ij}. Let t_1 be the time of the latest such event. By Lemma 14, there is a VOQ V_{ih} for which $\bar{p}_{ij}(t_1) \leq \bar{p}_ih(t_1)$ and since V_{ih} precedes V_{ij} at t_1, $q_{ij}(t_1) \geq q_{ih}(t_1)$ and consequently, $\underline{margin}_{ij}(t_1) \geq \underline{margin}_{ih}(t_1)$. If $t_1 = t$, then since V_{ij} is the oldest VOQ that does not satisfy the lemma at t, V_{ih} does satisfy the lemma. That is,

$$\underline{margin}_{ih}(t) \geq L_M/S$$

or

$$\bar{p}_{ih}(t) \leq (1+1/S)L_M - (S-1)(t-\tau_{ih}).$$

If $\underline{margin}_{ih}(t) \geq L_M/S$, then $\underline{margin}_{ij}(t) \geq L_M/S$ also. On the other hand, if $\bar{p}_{ih}(t) \leq (1+1/S)L_M - (S-1)(t-\tau_{ih})$, then

$$\begin{aligned}\bar{p}_{ij}(t) &\leq (1+1/S)L_M - (S-1)(t-\tau_{ih}) \\ &\leq (1+1/S)L_M - (S-1)(t-\tau_{ij}).\end{aligned}$$

Once again, this contradicts our assumption that the lemma does not hold at t. Consequently, we must have $t_1 < t$ and since t is the earliest time at which the lemma is violated, either

$$\underline{margin}_{ij}(t_1) \geq L_M/S$$

or

$$\bar{p}_{ij}(t_1) \leq (1+1/S)L_M - (S-1)(t_1-\tau_{ij}).$$

If $\underline{margin}_{ij}(t_1) \geq L_M/S$ then by Lemma 13, $\underline{margin}_{ij}(t) \geq L_M/S$ also. Assume then, that

$$\bar{p}_{ij}(t_1) \leq (1+1/S)L_M - (S-1)(t_1-\tau_{ij}).$$

Now, if $\beta_{ij} \geq t$, then V_{ij} is eligible for selection at every event in $[t_1, t]$ and so

$$\bar{p}_{ij}(t) \leq (1+1/S)L_M - (S-1)(t-\tau_{ij}).$$

On the other hand, if $t_1 \leq \beta_{ij} < t$, it follows that

$$\bar{p}_{ij}(\beta_{ij}) \leq (1+1/S)L_M - (S-1)(\beta_{ij}-\tau_{ij})$$

and since $q_{ij}(\beta_{ij}) \geq (1-1/S)(B-2L_M)$,

$$\begin{aligned}\underline{margin}_{ij}(\beta_{ij}) &\geq (1-1/S)(B-2L_M) \\ &\quad - ((1+1/S)L_M - (S-1)(\beta_{ij}-\tau_{ij})) \\ &\geq (1-1/S)B - (3-1/S)L_M \geq L_M/S.\end{aligned}$$

Since $\underline{margin}_{ij}$ cannot decrease in $[t_1, t]$, it follows that $\underline{margin}_{ij}(t) \geq L_M/S$. Once again, this contradicts the assumption that the lemma does not hold at t.

That leaves one more case: $\beta_{ij} < t_1$. Since $\underline{margin}_{ij}(t_1) \geq L_M/S$ implies that $\underline{margin}_{ij}(t) \geq L_M/S$, we must have

$$\bar{p}_{ij}(t_1) \leq (1+1/S)L_M - (S-1)(t_1-\tau_{ij}).$$

Also, since $q_{ij}(\beta_{ij}) \geq (1-1/S)(B-2L_M)$,

$$q_{ij}(t_1) \geq (1-1/S)(B-2L_M) - (t_1-\beta_{ij})$$

and

$$\begin{aligned}\underline{margin}_{ij}(t_1) &\geq (1-1/S)(B-2L_M) - (t_1-\beta_{ij}) \\ &\quad - ((1+1/S)L_M - (S-1)(t_1-\tau_{ij})) \\ &\geq (1-1/S)B + (S-2)(t_1-\tau_{ij}) \\ &\quad - (3-1/S)L_M \\ &\geq (1-1/S)B - (3-1/S)L_M \geq L_M/S.\end{aligned}$$

This completes the contradiction to our original assumption that the lemma does not hold at time t. ∎

Corollary 2: If V_{ij} is active at $t \geq f_{ij} + 2L_M/(S-1)$, then for any RPLF scheduler, $margin_{ij}(t) \geq L_M/S$, if $S \geq 2$ and $B \geq 3L_M S/(S-1)$.

Proof: For $t \geq f_{ij} + 2L_M/(S-1)$,

$$\begin{aligned}&(1+1/S)L_M - (S-1)(t-\tau_{ij}) \\ &\quad \leq (1+1/S)L_M + (S-1)(\tau_{ij}-f_{ij}) - 2L_M \\ &\quad \leq (1+1/S)L_M + (1-1/S)L_M - 2L_M \leq 0.\end{aligned}$$

Consequently, the lemma implies that $\underline{margin}_{ij}(t) \geq 0$ for $t \geq f_{ij} + 2L_M/(S-1)$ and since $margin_{ij}(t) \geq \underline{margin}_{ij}(t)$, $margin_{ij}(t) \geq 0$ also. ∎

Proof of Theorem 5: Suppose that up until time t, the PLF crossbar faithfully emulates the output-queued switch with added delay T, but that at time t, the output-queued switch begins to forward an ij-packet x, while the crossbar does not.

Now suppose that in the crossbar, one or more bits of x have reached B_{ij} by time $t - L_M/S$. Note that the interval $[t - L_M/S, t)$ must contain at least one scheduling event at output j and all such events must select packets that precede x. However, this implies that during some non-zero time interval $[t_1, t]$, output j is continuously receiving bits that precede x at a faster rate than it can forward them to the output. This contradicts the fact that by time t the crossbar forwards all bits that precede x (since it faithfully emulates the output-queued switch up until time t).

Assume then that at time $t - L_M/S$, no bits of x have reached B_{ij}. Since the output-queued switch has a delay of T, $f(x) \leq t - T$ and so $t - L_M/S \geq f(x) + 2L_M/(S-1)$. Since the crossbar has sent everything sent by the output-queued switch up until time t, it follows that $q_{ij}(t - L_M/S) \leq L_M/S$. By Corollary 2, $margin_{ij}(t_1) \geq L_M/S$ and hence $p_{ij}(t_1) < 0$, which is not possible. ∎

VII. Segment-Based Switching

Chuang *et al.* [3] showed that cell-based crossbars can emulate an output-queued switch using any push-in, first-out (PIFO) queueing discipline. It is straightforward to define PIFO scheduling policies that keep the cells of a packet together (simply insert later arriving cells of a given packet right after their immediate predecessors). This makes it possible to provide strong performance guarantees for packets not just cells, using variants of standard crossbar schedulers that are *packet-aware*. (Thanks to the anonymous referee who made this observation in his insightful review of an earlier version of this paper.) Note that this method may require that the output line card forward cells that form the initial part of a packet, before all cells in the packet are received, but this is feasible in this context, since the crossbar scheduler can guarantee that the remaining cells are received by the time they are needed. While packet-aware schedulers can provide packet-level performance guarantees in systems that use cell-based crossbars, such systems still suffer from bandwidth fragmentation, since packet lengths are generally not even multiples of the cell length.

One possible objection to the use of crosspoint buffers that are large enough to hold packets is that they might be too expensive, even for modern integrated circuit components. A 32 port crossbar equipped with buffers large enough to hold two 1500 byte packets would require a total of more than 3 MB of SRAM. In [10], the authors propose switching variable length *segments* rather than cells, as a way of addressing the fragmentation problem with fixed-size cells. If this is coupled with a packet-aware crossbar scheduler that provides performance guarantees for variable length packets, we can reduce the crossbar buffer size to a multiple of the maximum segment length. For IP routers, a maximum segment length of 80 bytes is sufficient to eliminate bandwidth loss due to fragmentation effects. Even after adding 20 bytes for header information this reduces the required buffer size by a factor of 15, making it small enough to be easily accommodated within the constraints of current circuit technologies.

Also observe that in a segment-based system, an input line card can forward segments to an output before all segments of the packet have been received. The performance guarantee for the crossbar ensures that the remaining segments are transferred through the crossbar in time to be forwarded on the outgoing link, if the system is operated with a speedup of 2. Thus, we reduce both the amount of buffering required and the delay.

VIII. Concluding Remarks

The results of Sections V and VI can be extended to systems that place different constraints on where and when packets are buffered. In particular, most routers buffer packets at both input and output line cards, not just at the inputs. Modifying the analysis to handle this case is straightforward and requires only that the value of T be increased by L_M/S, to accommodate the added delay for a maximum length packet to be fully buffered at the outputs.

With an asynchronous crossbar, it is possible to build a system in which packets pass from inputs to outputs without ever being fully buffered. This is known as cut-through switching [7] and can provide superior delay performance when load is light. While our results cannot be directly applied to such systems, it seems likely that similar results could be developed for this model. Indeed, the segment-based switches already approach the behavior of a cut-through switch.

There are several ways the work described here can be extended. First, there are opportunities for tightening the results, particularly with respect to the crossbar buffer size. There seems to be no intrinsic reason that PLF should require a larger crossbar buffer size than PGV. An analysis that directly compares the behavior of a PLF scheduler to a PGV scheduler may be able to reduce the buffer size requirement for PLF.

It would also be interesting to see if the analysis techniques can be extended to provide stronger performance guarantees. In particular, it would be useful to show that an asynchronous buffered crossbar can emulate an output-queued switch using any PIFO queueing discipline, not just any restricted PIFO discipline. The difficulty in making the transition from restricted PIFO queueing disciplines to unrestricted PIFO disciplines is that once a packet is in a crossbar buffer, there is no way for a later arriving packet from the same input to reach the output line card before it does, even if the queueing discipline gives it higher priority. Reference [4] describes several techniques that can be used to allow cell switches using buffered crossbars to overcome this crosspoint blocking phenomenon. It seems likely that these methods can be generalized to accommodate asynchronous crossbars.

Still another direction to explore is how scheduling algorithms that deliver strong performance guarantees when operated with a speedup of 2 perform when operated with a smaller speedup. Since the crossbar cost increases in direct proportion to the speedup, there are practical reasons to be interested in the performance of systems with smaller speedup, even if they are not able to deliver strong performance guarantees. A comprehensive simulation study exploring how such systems perform under a wide range of conditions would have considerable practical value.

References

[1] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. Comput. Syst.*, vol. 11, pp. 319–352, 1993.

[2] H. Attiya, D. Hay, and I. Keslassy, "Packet-mode emulation of output-queued switches," in *Proc. ACM SPAA*, 2006, pp. 138–147.

[3] S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input output queued switch," *IEEE J. Sel. Areas Commun.*, vol. 17, pp. 1030–1039, Dec. 1999.

[4] S. T. Chuang, S. Iyer, and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," in *Proc. IEEE INFOCOM*, Mar. 2005, pp. 981–991.

[5] Y. Ganjali, A. Keshavarzian, and D. Shah, "Input queued switches: Cell switching vs. packet switching," in *Proc. IEEE INFOCOM*, 2003, pp. 1651–1658.

[6] S. Iyer, R. Zhang, and N. McKeown, "Routers with a single stage of buffering," in *Proc. ACM SIGCOMM*, Sep. 2002, pp. 251–264.

[7] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, vol. 3, pp. 267–286, 1979.

[8] P. Krishna, N. Patel, A. Charny, and R. Simcoe, "On the speedup required for work-conserving crossbar switches," *IEEE J. Sel. Areas Commun.*, vol. 27, pp. 1052–1066, Jun. 1999.

[9] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, and N. Chrysos, "Variable packet size buffered crossbar (CICQ) switches," in *Proc. IEEE Int. Conf. Communications*, Jun. 2004, pp. 1090–1096.

[10] M. Katevenis and G. Passas, "Variable-size multipacket segments in buffered crossbar (CICQ) architectures," in *Proc. IEEE Int. Conf. Communications*, May 2005, pp. 16–20.

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

[12] B. Magill, C. Rohrs, and R. Stevenson, "Output-queued switch emulation by fabrics with limited memory," *IEEE J. Sel. Areas Commun.*, vol. 21, pp. 606–615, May 2003.

[13] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Packet-mode scheduling in input-queued cell-based switches," *IEEE/ACM Trans. Networking*, vol. 10, pp. 666–678, 2002.

[14] N. McKeown, "*i*SLIP: A scheduling algorithm for input-queued switches," *IEEE Trans. Networking*, vol. 7, pp. 188–201, Apr. 1999.

[15] 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.

[16] L. Mhamdi and M. Hamdi, "MCBF: A high-performance scheduling algorithm for buffered crossbar switches," *IEEE Commun. Lett.*, vol. 7, pp. 451–453, 2003.

[17] S. Nojima, E. Tsutsui, H. Fukuda, and M. Hashimoto, "Integrated services packet network using bus matrix switch," *IEEE J. Sel. Areas Commun.*, vol. 5, pp. 1284–1292, Oct. 1987.

[18] T. Rodeheffer and J. Saxe, "An efficient matching algorithm for a high-throughput, low-latency data switch," Compaq Systems Research Ctr., Research Rep. 162, Nov. 5, 1998.

[19] Rojas-Cessa, E. Oki, Z. Jing, and H. J. Chao, "CIXB-1: Combined input-one-cell-crosspoint buffered switch," in *IEEE Workshop on High Performance Switching and Routing*, Jul. 2001, pp. 324–329.

[20] D. Stevens and H. Zhang, "Implementing distributed packet fair queueing in a scalable switch architecture," in *Proc. IEEE INFOCOM*, 1998, pp. 282–290.

[21] J. Turner, "Strong performance guarantees for asynchronous crossbar schedulers," *Proc. IEEE INFOCOM*, 2006, 11 pp, Online.

Jonathan S. Turner (F'90) holds the Barbara and Jerome Cox Chair of Computer Science at Washington University, and is the Director of the Applied Research Laboratory. The Applied Research Laboratory creates experimental networking technology to validate and demonstrate new research innovations. He served as Chief Scientist for Growth Networks, a startup company that developed scalable switching components for Internet routers and ATM switches, before being acquired by Cisco Systems in early 2000.

Prof. Turner is a Fellow of both the ACM and the IEEE and is a member of the National Academy of Engineering. He received the Koji Kobayashi Computers and Communications Award from the IEEE in 1994 and the IEEE Millenium Medal in 2000. He has been awarded 30 patents for his work on switching systems and has many widely cited publications.