2007-52

Strong Performance Guarantees for Asynchronous Buffered Crossbar Schedulers

Authors: Jonathan Turner

November 11, 2007

Corresponding Author: jon.turner@wustl.edu

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. Up to now, 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 the performance guarantees previously available only 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 segment-based crossbars, can deliver strong performance guarantees with small buffer requirements and no bandwidth fragmentation.

Strong Performance Guarantees for Asynchronous Buffered Crossbar Schedulers

Jonathan Turner
Washington University
jon.turner@wustl.edu

January 30, 2008

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. Up to now, 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 the performance guarantees previously available only 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.

1 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 [11]. Recently, there has been a growing body of work providing rigorous performance guarantees for such systems [9, 12] 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 [2, 6, 15], 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 [2] for a crossbar with small speedup, where the speedup of a crossbar switch is the ratio of the ideal throughput of the crossbar 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.

The strong performance guarantees that have been established to date, apply only to crossbars that forward fixed length data units, or 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 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 [2] and the Least Occupied Output First Algorithm (LOOFA) of [6, 15] 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 [14]. 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 [3, 4, 7, 8, 10, 13, 16]. A recent paper by Chuang et. al. [3] 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 [7, 8] 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 2 discusses the differences between switching cells and switching packets, and explains how buffered crossbars are particularly advantageous for systems that directly switch packets. Section 3 defines the terminology and notation used in the analysis to follow. Section 4 collects several key lemmas that are used repeatedly in the analysis. Section 5 presents strong performance guarantees for a packet variant of the Group by Virtual Output Queue crossbar scheduler. Section 6 presents a similar set of guarantees for a packet variant of the Least Occupied Output First scheduler. Section 7 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 8 provides some closing remarks, including a discussion of several ways this work can be extended.

2 Switching Packets vs. Switching 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

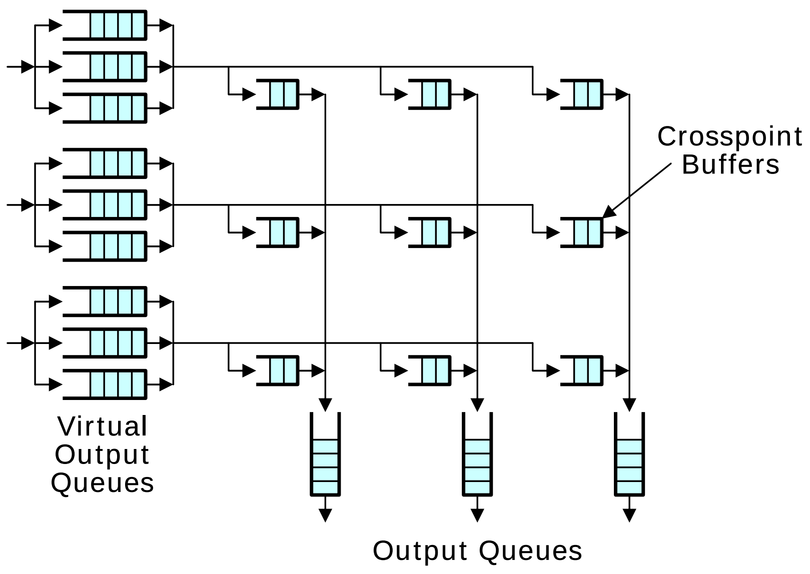

Figure 1: Buffered Crossbar

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, $> n\left(\lceil L/C \rceil - 1\right)$ cells will 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 more fully in Section 7.

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 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 Figure 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 [3], 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 virtual output queues, 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.

3 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*. 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 virtual output queue 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

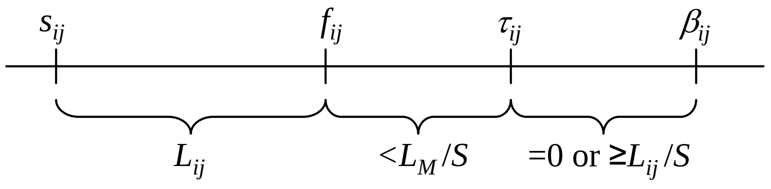

Figure 2: Basic Definitions for Active Periods

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 the quantities that include a time parameter, we sometimes omit the time parameter when its value can be understood from the context.

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 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 Figure 2. Note that $\tau_{ij} < f_{ij} + L_M/S$ and that if $\beta_{ij} \neq \tau_{ij}$, then $\beta_{ij} \geq \tau_{ij} + L_{ij}/S$. Figure 2 illustrates the definitions for key points in time in an active period and their relationships.

While we require that packets be fully buffered at inputs, we assume that packets can be streamed directly though crossbar buffers to outputs, and through output buffers to outgoing links. The former assumption is the natural design choice. The latter assumption 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 also say that bits in V_{ij} 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} We define $\mathit{slack}_{ij}(t) = q_j(t) - p_{ij}(t)$ and $\mathit{margin}_{ij}(t) = q_{ij}(t) - p_{ij}(t)$.

Our worst-case performance guarantees are defined relative to a reference system consisting of an ideal output-queued switch followed by a fixed output delay of length T. An ideal 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 [2], 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 queuing disciplines of practical interest belong to this class. In [3], 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.

We say that a crossbar *T-emulates* an output-queued switch with a specific restricted PIFO 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. 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. Alterntatively, we 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 will be defined by a specific crossbar scheduling algorithm and by the output queueing discipline of the switch being emulated. 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.

4 Common Properties

We consider only schedulers that 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*. Prompt schedulers have some common properties that will be useful in the analysis to follow.

Lemma 1 *For any prompt scheduler,* $q_j(t) \geq (1 - 1/S)B_{ij}(t)$.

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, if* $B_{ij}(\tau_{ij}) > 0$ *then*

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

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,

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

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

Lemma 3 *For any prompt, restricted PIFO 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 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}. 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 4 *Consider an active period for V_{ij}. For any prompt restricted PIFO scheduler, if $B_{ij}(\tau_{ij}) \geq L_M$ then*

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

proof. Since B_{ij} cannot grow between f_{ij} and τ_{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,

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

■

We say that a scheduling algorithm is *stable* if it does not change the relative order of any two VOQs during a period when they are both continuously active.

Lemma 5 *Let $t_1 \leq t \leq t_2$, where t_1 is the time of an input scheduling event in an active period of V_{ij} and t_2 is the time of the next event in the same active period if there is one, or the end of the active period, if there is not. For any prompt and stable scheduler, if $B \geq 2L_M$ then $slack_{ij}(t) \geq slack_{ij}(t_1) + (S-2)(t - t_1)$.*

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 $t_1 + L_M/S \geq t_2$. Consequently, $q_j(t) \geq q_j(t_1) + (S-1)(t - t_1)$. Since the VOQ ordering is stable 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 $slack_{ij}(t) \geq slack_{ij}(t_1) + (S-2)(t - t_1)$. ■

Lemma 5 implies that for any prompt and stable scheduler, if $S \geq 2$, $slack_{ij}$ does not decrease after the first event of an active period. However, $slack_{ij}$ may decrease before the first event, and indeed it may be negative.

5 Packet Group by Virtual Output Queue

Group by Virtual Output Queue (GVOQ) is a cell switch scheduling algorithm first described in [2] and extended to buffered crossbars in [3]. 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 stable. 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.

5.1 Work-Conservation of PGV

Our first result shows that regardless of the specific output scheduling policy used, PGV is work-conserving. Since the VOQ ordering used by PGV is stable, Lemma 5 implies that slack_{ij} does not decrease after the first input scheduling event of an active period, if $S \geq 2$. Our first lemma for PGV shows that there must be a backlog event close to the start of any active period.

Lemma 6 *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 a VOQ that precedes V_{ij}. Since the scheduling algorithm is stable, 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 $p_{ij}(\tau_{ij}) \leq L_{ij} + (\tau_{ij} - f_{ij})$,

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

which 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, $\mathit{slack}_{ij} \geq 0$.

Lemma 7 *Consider some 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 $B \geq 3L_M$, $\mathit{slack}_{ij}(t) > 0$.*

proof. We show that $\mathit{slack}_{ij}(\beta_{ij}) > 0$. The result then follows from Lemmas 5 and 6. If V_{ij} is backlogged at τ_{ij}, then $\beta_{ij} = \tau_{ij}$ and by Lemma 2, $q_j(\beta_{ij}) > (1-1/S)(B-L_M) + (S-1)(\tau_{ij} - f_{ij})$. For any PGV scheduler, $p_{ij}(\beta_{ij}) = p_{ij}(\tau_{ij}) \leq L_{ij} + (\tau_{ij} - f_{ij})$. Consequently,

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

since $S \geq 2$ and $B \geq 3L_M$.

Now, suppose V_{ij} is not backlogged at τ_{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,

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

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

Theorem 1 *Any PGV scheduler with $S \geq 2$ and $B \geq 3L_M$ is T-work-conserving for $T \geq 2L_M/(S-1)$.*

proof. 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 7, $slack_{ij}(t) > 0$. 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 from $3L_M$ to $2L_M$.

Lemma 8 *Let $t_1 \leq t \leq t_2$, where t_1 is the time of an input scheduling event in an active period of V_{ij} and t_2 is the time of the next event in the active period if there is one, or the end of the active period, if there is not. For any PGV 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 5. Assume then that $V_{ij}(t) < V_{ij}(t_1)$. This implies that V_{ij} was selected at t_1. Consequently, during the interval $[t_1, t]$, q_j is increasing at rate $S-1$, while p_{ij} is decreasing at rate $\geq S-1$. Thus, $slack(t) \geq slack(t_1) + 2(S-1)(t-t_1)$. 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,

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

since $S \geq 2$. ■

Lemma 9 *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 8, $slack_{ij}(t) > -V_{ij}(t)$.

Now, suppose that $B_{ij}(\tau_{ij}) = 0$. By Lemma 6, $\beta_{ij} \leq f_{ij} + 2L_M/(S-1) \leq t$. 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,

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

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

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

By Lemma 8, $slack_{ij}(t) > -V_{ij}(t)$. ■

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

proof. 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, $slack_{ij}(t) > -V_{ij}(t)$. Since, $q_j(t) = 0$, this implies that $p_{ij}(t) < V_{ij}(t)$, which contradicts the definition of p_{ij}. ■

5.2 Emulation Results for PGV

The analysis of the previous section can be extended to show that PGV can T-emulate an ideal output-queued switch using any restricted PIFO queueing discipline. We define a particular PGV scheduler based on the packet ordering defined by an output-queued switch using the queueing discipline of interest. That packet order is used by the crossbar's output scheduler when selecting crosspoint buffers. The output line card of the crossbar uses the same PIFO queueing discipline, while also holding each packet until at least T time units have passed since its arrival.

We refer to a such 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].

The analysis leading to the T-emulation result is similar to the analysis used to establish work-conservation. We start with two lemmas which establish that $margin_{ij}$ does not decrease and that it becomes positive a short time after the start of an active period.

Lemma 10 *Let $t_1 \leq t \leq t_2$, where t_1 is the time of an input scheduling event in an active period of V_{ij} and t_2 is the time of the next event in the active period if there is one, or the end of the active period, if there is not. For any stable, restricted PIFO 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 p_{ij} can increase at a rate no faster than 1 during this period, $\mathit{margin}_{ij}(t) \geq \mathit{margin}_{ij}(t_1) + (S-2)(t-t_1)$. If V_{ij} is not backlogged at t_1, p_{ij} decreases at rate $\geq S-1$ in the interval $[t_1, t]$ and since q_{ij} can decrease at a rate no faster than 1, $\mathit{margin}_{ij}(t) \geq \mathit{margin}_{ij}(t_1) + (S-2)(t-t_1)$. ■

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

proof. If V_{ij} is backlogged at τ_{ij}, then by Lemma 4,

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

and since $p_{ij}(\tau_{ij}) \leq L_{ij} + (\tau_{ij} - f_{ij})$,

$$\begin{aligned} \mathit{margin}_{ij}(\tau_{ij}) &> (1-1/S)(B-(L_{ij}+L_M)) + (S-1)(\tau_{ij}-f_{ij}) - (L_{ij} + (\tau_{ij}-f_{ij})) \\ &= (1-1/S)B - (2-1/S)L_{ij} - (1-1/S)L_M + (S-2)(\tau_{ij}-f_{ij}) \\ &\geq (1-1/S)B - (3-2/S)L_M \end{aligned}$$

This is $\geq L_M/S$ so long as $B \geq (S/(S-1))(3-1/S)L_M = (3+2/(S-1))L_M$, which is the condition on B in the statement of the lemma. By Lemma 10, $\mathit{margin}_{ij}(t) > L_M/S$.

Now suppose V_{ij} is not backlogged at τ_{ij}. By Lemma 6, $\beta_{ij} \leq t$ and by Lemma 3, $q_{ij}(\beta_{ij}) > (1-1/S)(B-2L_M)$. 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

$$\mathit{margin}_{ij}(\beta_{ij}) > (1-1/S)(B-2L_M) - 2L_M/S = (1-1/S)B - 2L_M$$

This is $\geq L_M/S$ so long as $B \geq (S/(S-1))(2+1/S)L_M = ((2+3/(S-1))L_M$ which is implied by the condition on B in the statement of the lemma. By Lemma 10 $\mathit{margin}_{ij}(t) > L_M/S$. ■

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 $T \geq (2/(S-1) + 1/S)L_M$.*

proof. 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 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 therefore 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, it follows that $f(x) \leq t - T$, 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, $\mathit{margin}_{ij}(t - L_M/S) > L_M/S$ 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 to $3L_M$ using a more sophisticated analysis. Unfortunately, the technique used in Lemma 9 cannot be applied here in a straightforward way, because q_{ij} does not increase whenever $B_{ij} > 0$.

6 Packet LOOFA

The Least Occupied Output First Algorithm (LOOFA) is a cell scheduling algorithm described in [6]. 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, 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 they can change, PLF is also more responsive to changes in output queue lengths than PGV. While this has no effect on work-conservation when $S \geq 2$, it can be expected to yield better performance for smaller speedups.

6.1 Work-Conservation for PLF

To analyze PLF, we need some additional terminology. A *non-empty interval* for input i, is any continuous time period during which there is some non-empty VOQ at input i. 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 V_{ij} is *mature* at time t if $t - f_{ij} \geq T_M = (2/(S-1))L_M$. 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.

We also need to generalize our notation for p_{ij}. In particular, we let $p_{ij}(t, I)$ be the number of bits at input i that precede V_{ij} at time t and that arrived in the interval I. So for example, $p_{ij}(t, [0, s_{ij}))$ counts those bits that precede V_{ij} that arrived before s_{ij}, and $p_{ij}(t, [s_{ij}, t])$ counts those bits that precede V_{ij} that arrived since s_{ij}. We also define $\overline{p}_i(t, I)$ to be the number of bits at input i at time t that arrived in the interval I, and we define $\overline{p}_{ij}(t) = p_{ij}(t, [0, s_{ij})) + \overline{p}_i(t, [s_{ij}, t])$. Note that $p_{ij}(t) \leq \overline{p}_{ij}(t)$. Finally, we let $\underline{\mathit{slack}}_{ij}(t) = q_j(t) - \overline{p}_{ij}(t)$ and $\underline{\mathit{margin}}_{ij}(t) = q_{ij}(t) - \overline{p}_{ij}(t)$. Note that $\mathit{slack}_{ij}(t) \geq \underline{\mathit{slack}}_{ij}(t)$

and $margin_{ij}(t) \geq \underline{margin}_{ij}(t)$. Our first lemma plays essentially the same role as Lemma 5, in the work-conservation result for PGV.

Lemma 12 *Let $C \geq 0$ be a constant and let $t_1 \leq t \leq t_2$, where t_1 is the time of an input scheduling event in an active period of V_{ij} and t_2 is the time of the next event in the active period if there is one, or the end of the active period, if there is not. For any PLF scheduler, if $B \geq 2L_M$, $\underline{slack}_{ij}(t_1) \geq C$ and all VOQs V_{ih}, that are older than V_{ij} at t_1 satisfy $\underline{slack}_{ih}(t) \geq C$, then $\underline{slack}_{ij}(t) \geq C + (S-2)(t-t_1)$.*

proof. To establish the lemma, we need to account for the bits in VOQs that pass V_{ij} in the interval $[t_1, t]$. Note that we need not account for the bits in VOQs that are younger than V_{ij}, since the relative ordering of these VOQs with V_{ij} does not affect $\overline{p}_{ij}$ and hence does not affect $\underline{slack}_{ij}$. In fact, we also need not account for bits in older VOQs that arrived after s_{ij}. Therefore, let P be the set of VOQs that are older than V_{ij} and that pass V_{ij} during $[t_1, t]$, and let r be the number of bits in VOQs in P that arrived before s_{ij} and are still present at input i at time t. Let $\Delta = t - t_1$.

Assume first that $r = 0$. If V_{ij} is backlogged at t_1, then output j receives $S\Delta$ bits during the interval $[t_1, t]$. If V_{ij} is not backlogged at t_1, then $S\Delta$ bits that precede V_{ij} at t_1, leave input i during $[t_1, t]$. This implies that $\underline{slack}_{ij}(t) \geq \underline{slack}_{ij}(t_1) + (S-2)\Delta \geq C + (S-2)\Delta$.

Next, assume that $r > 0$. Let V_{ih} be the VOQ in P that comes latest in the VOQ ordering at t_1, let $k = p_{ih}(t_1, [0, s_{ij})) - p_{ij}(t_1, [0, s_{ij}))$, and note that $r \leq k$. Also, note that since

$$\overline{p}_{ij}(t_1) = p_{ij}(t_1, [0, s_{ij})) + \overline{p}_i(t_1, [s_{ij}, t_1])$$

and

$$\overline{p}_{ih}(t_1) = p_{ih}(t_1, [0, s_{ih})) + \overline{p}_i(t_1, [s_{ih}, t_1]) \geq p_{ih}(t_1, [0, s_{ij})) + \overline{p}_i(t_1, [s_{ij}, t_1])$$

it follows that, $k \leq \overline{p}_{ih}(t_1) - \overline{p}_{ij}(t_1)$.

Since V_{ih} passes V_{ij}, output h must receive fewer bits than output j does during $[t_1, t]$, and since output j can receive no more than $S\Delta$ bits during $[t_1, t]$, output h receives fewer than $S\Delta$. This implies that $B_{ih}(t_1) < L_M \leq B - L_M$. Consequently, V_{ih} is eligible for selection at t_1, which implies that some packet x with $\geq S\Delta$ bits, that precedes V_{ih} at t_1 leaves input i during $[t_1, t]$.

We consider three cases. First, if x arrived after s_{ij} then, the departure of x reduces $\overline{p}_i(t_1, [s_{ij}, t_1])$ by $S\Delta$. Consequently,

$$\overline{p}_{ij}(t) \leq \overline{p}_{ij}(t_1) + r + \Delta - S\Delta \leq \overline{p}_{ij}(t_1) + k - (S-1)\Delta$$

Similarly, if x arrived before s_{ij} and x precedes V_{ij} at t_1 then the departure of x reduces $p_{ij}(t_1, [0, s_{ij}))$ by $S\Delta$. Consequently,

$$\overline{p}_{ij}(t) \leq \overline{p}_{ij}(t_1) + r + \Delta - S\Delta \leq \overline{p}_{ij}(t_1) + k - (S-1)\Delta$$

Finally, if x arrived before s_{ij} and x does not precede V_{ij} at t_1 then $r + S\Delta \leq k$ and

$$\overline{p}_{ij}(t) \leq \overline{p}_{ij}(t_1) + r + \Delta \leq \overline{p}_{ij}(t_1) + k - (S-1)\Delta$$

So, in all three cases $\overline{p}_{ij}(t) \leq \overline{p}_{ij}(t_1) + k - (S-1)\Delta$. Since V_{ih} passes V_{ij}, $q_j(t) > q_h(t) \geq q_h(t_1) - \Delta$ and so,

$$\begin{aligned}
\underline{slack}_{ij}(t) &> (q_h(t_1) - \Delta) - (\overline{p}_{ij}(t_1) + k - (S-1)\Delta) \\
&\geq (q_h(t_1) - \overline{p}_{ih}(t_1)) + (S-2)\Delta \\
&= \underline{slack}_{ih}(t_1) + (S-2)\Delta \\
&\geq C + (S-2)\Delta
\end{aligned}$$

■

Our next lemma establishes a lower bound on $\underline{slack}$ for all mature VOQs.

Lemma 13 *Given a crossbar with $S \geq 2$, $B \geq \max\{2 + 1/(S-1), (1/(2S-1))(3S + 5 + 4/(S-1))\}L_M$ and a PLF scheduler. If V_{ij} is mature at time t, $\underline{slack}_{ij}(t) \geq 0$.*

proof. Note that at the start of a non-empty period at input i, no VOQs are mature. So initially, all mature VOQs have $\underline{slack} \geq 0$. Assume then that t is the earliest time in the non-empty period when there is some mature VOQ with $\underline{slack} < 0$ and let V_{ij} be the oldest VOQ for which $\underline{slack}_{ij}(t) < 0$. Let $t_1 < t$ be the time of the most recent event at input i before t. Suppose first that V_{ij} was mature at t_1. Since $t_1 < t$, this implies that $\underline{slack}_{ij}(t_1) \geq 0$. Since all VOQs older than V_{ij} are also mature at t_1, then these all satisfy $\underline{slack} \geq 0$ at t_1. Consequently, by Lemma 12, $\underline{slack}_{ij}(t) \geq 0$.

Assume then that V_{ij} was not mature at t_1. This implies that

$$t_1 < f_{ij} + T_M \leq t \leq t_1 + L_M/S < f_{ij} + T_M + L_M/S$$

We divide the remainder of the analysis into three cases.

Case 1. $p_{ij}(t, [0, s_{ij})) = 0$ and V_{ij} was backlogged at τ_{ij}. Since V_{ij} was backlogged at τ_{ij}, Lemma 2 implies that $q_j(\tau_{ij}) \geq (1 - 1/S)(B - L_M) + (S-1)(\tau_{ij} - f_{ij})$ and q_j grows at rate $S - 1$ until at least $\tau_{ij} + (B - L_M)/S$.

If $t \leq \tau_{ij} + (B - L_M)/S$, $q_j(t) \geq (1 - 1/S)(B - L_M) + (S-1)(t - f_{ij})$ and since $p_{ij}(t, [0, s_{ij})) = 0$, $\overline{p}_{ij}(t) \leq t - s_{ij}$. Hence

$$\begin{aligned}
\underline{slack}_{ij}(t) &\geq (1 - 1/S)(B - L_M) + (S-1)(t - f_{ij}) - (t - (f_{ij} - L_{ij})) \\
&= (1 - 1/S)(B - L_M) + (S-2)(t - f_{ij}) - L_{ij} \\
&\geq (1 - 1/S)(B - L_M) - L_M \\
&= (1 - 1/S)B - (2 - 1/S)L_M
\end{aligned}$$

This is ≥ 0, so long as $B \geq (2 + 1/(S-1))L_M$, which is implied by the condition on B in the statement of the lemma.

Assume then that $t > \tau_{ij} + (B - L_M)/S$. In this case,

$$\begin{aligned}
q_j(t) &\geq 2(1 - 1/S)(B - L_M) + (S-1)(\tau_{ij} - f_{ij}) - [t - (\tau_{ij} + (B - L_M)/S)] \\
&\geq (2 - 1/S)(B - L_M) - (t - \tau_{ij})
\end{aligned}$$

and

$$\begin{aligned}
\underline{slack}_{ij}(t) &\geq [(2-1/S)(B-L_M)-(t-\tau_{ij})]-[t-(f_{ij}-L_{ij})] \\
&\geq (2-1/S)(B-L_M)-2(f_{ij}+T_M+L_M/S)+\tau_{ij}+f_{ij}-L_M \\
&\geq (2-1/S)(B-L_M)+(\tau_{ij}-f_{ij})-2T_M-(2/S)L_M-L_M \\
&\geq (2-1/S)B-[(2-1/S)+(4/(S-1))+(2/S)+1]L_M \\
&= (2-1/S)B-[3+4/(S-1)+1/S]L_M
\end{aligned}$$

which is ≥ 0 if $B \geq (1/(2S-1))(3S+5+4/(S-1))L_M$, which is implied by the condition on B in the statement of the lemma. This completes Case 1.

Case 2. $p_{ij}(t,[0,s_{ij}))=0$ and V_{ij} is not backlogged at τ_{ij}. Suppose there is no backlog event for V_{ij} in $[\tau_{ij},t]$. Then,

$$\begin{aligned}
\overline{p}_{ij}(t) &\leq L_{ij}+(\tau_{ij}-f_{ij})-(S-1)(t-\tau_{ij}) \\
&< L_{ij}+L_M/S-(S-1)(T_M-L_M/S) \\
&\leq 2L_M-(S-1)T_M \leq 0
\end{aligned}$$

This contradicts the fact that V_{ij} is active at t, so there must be some backlog event in $[\tau_{ij},t]$. Note that $\beta_{ij} \geq \tau_{ij}+L_{ij}/S$, so

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

By Lemma 1, $q_j(\beta_{ij}) \geq (1-1/S)(B-L_M)$, so $\underline{slack}_{ij}(\beta_{ij}) \geq (1-1/S)(B-L_M)-2L_M/S$ and this is ≥ 0 so long as $B \geq (1+2/(S-1))L_M$, which is implied by the condition on B in the statement of the lemma. If $\beta_{ij}+(B-L_M)/S \geq t$, then q_j continues to grow at rate $(S-1)$ until t. This is enough to compensate for any growth in $\overline{p}_{ij}$. Hence, $\underline{slack}_{ij}(t) \geq L_M/S$.

Assume then that $\beta_{ij}+(B-L_M)/S < t$. In this case, q_j continues to grow at rate $(S-1)$ until $\beta_{ij}+(B-L_M)/S$ giving $q_j(\beta_{ij}+(B-L_M)/S) > 2(1-1/S)(B-L_M)$. Thus,

$$q_j(t) > 2(1-1/S)(B-L_M)-(t-(\beta_{ij}+(B-L_M)/S)) \geq (2-1/S)(B-L_M)-(t-\beta_{ij})$$

and since $\overline{p}_{ij}(t) \leq t-s_{ij}$.

$$\begin{aligned}
\underline{slack}_{ij}(t) &\geq (2-1/S)(B-L_M)-(t-\beta_{ij})-(t-s_{ij}) \\
&\geq (2-1/S)(B-L_M)-2t+(\tau_{ij}+L_{ij}/S)+(f_{ij}-L_{ij}) \\
&\geq (2-1/S)(B-L_M)-2(f_{ij}+T_M+L_M/S)+(\tau_{ij}+L_{ij}/S)+(f_{ij}-L_{ij}) \\
&\geq (2-1/S)(B-L_M)-2(T_M+L_M/S)+(\tau_{ij}-f_{ij})-(1-1/S)L_{ij} \\
&\geq (2-1/S)B-[(2-1/S)+4/(S-1)+2/S+(1-1/S)]L_M \\
&\geq (2-1/S)B-[3+4/(S-1)]L_M
\end{aligned}$$

which is ≥ 0 if $B \geq (1/(2S-1))(3S+4+4/(S-1))L_M$, which is implied by the condition on B in the statement of the lemma. This completes Case 2.

Case 3. $p_{ij}(t, [0, s_{ij})) > 0$. This implies that there is some VOQ that precedes V_{ij} at t and is older than V_{ij}. Let V_{ih} be one such VOQ and assume further, that among all such VOQs, it comes latest in the VOQ ordering at t. Note that since V_{ij} is the oldest VOQ with $\underline{slack} < L_M/S$ at t, $\underline{slack}_{ih}(t) \geq 0$.

Let K be the set of bits that precede V_{ij} at time t but not V_{ih} and let $k = |K| = p_{ij}(t) - p_{ih}(t)$. Note that all bits in K 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 R be the set of bits that arrived after s_{ij} and are still present at time t and do not precede V_{ij} and let $r = |R| = \overline{p}_{ij}(t) - p_{ij}(t)$. Note that K and R have no bits in common. Now, let X be the set of bits that arrived since s_{ih} and do not precede V_{ih}. Note that both K and R are subsets of X and so $k + r \leq |X| = \overline{p}_{ih}(t) - p_{ih}(t)$. Consequently,

$$slack_{ih}(t) = q_h(t) - p_{ih}(t) \geq q_h(t) - \overline{p}_{ih}(t) + (k + r) \geq k + r$$

and

$$\underline{slack}_{ij}(t) = q_j(t) - (p_{ij}(t) + r) \geq q_h(t) - ((p_{ih}(t) + k) + r)) \geq 0$$

■

Note that Lemma 13 implies that that $slack_{ij}(t) \geq L_M/S$ for all mature V_{ij}. From this we obtain the work-conservation theorem for PLF.

Theorem 4 *A buffered crossbar with $S \geq 2$ and $B \geq \max\{2 + 1/(S-1), (1/(2S-1))(3S + 5 + 4/(S-1))\}L_M$ using any PLF scheduler is T-work-conserving for any $T \geq T_M$.*

proof. 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 Lemma 13, $slack_{ij}(t) \geq 0$. Since $q_j(t) = 0$, this implies that $p_{ij}(t) \leq 0$, which contradicts the fact that V_{ij} contains x at t. ■

6.2 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 non-empty, 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.

Lemma 14 *Let $C \geq 0$ be a constant and let $t_1 \leq t \leq t_2$, where t_1 is the time of an input scheduling event in an active period of V_{ij} and t_2 is the time of the next event in the active period if there is one, or the end of the active period, if there is not. For any RPLF scheduler, if $B \geq 2L_M$, $\underline{margin}_{ij}(t_1) \geq C$ and all VOQs V_{ih}, that are older than V_{ij} at t_1 satisfy $\underline{margin}_{ih}(t) \geq C$, then $\underline{margin}_{ij}(t) \geq C + (S-2)(t - t_1)$.*

proof. As in the proof of Lemma 12, we need to account for the bits in VOQs that pass V_{ij} in the interval $[t_1, t]$ and that arrived before s_{ij}. Therefore, let P be the set of VOQs that are older than V_{ij} and that pass V_{ij} during $[t_1, t]$, and let r be the number of bits in VOQs in P that arrived before s_{ij} and are still present at input i at time t. Let $\Delta = t - t_1$.

Assume first that $r = 0$. If V_{ij} is backlogged at t_1, then B_{ij} became non-empty before $t_1 - L_M/S$, which means that at the time of the most recent scheduling event at output j before t_1, B_{ij} contained a packet that precedes V_{ij}. Consequently, output j receives $S\Delta$ bits that precede V_{ij} during the interval $[t_1, t]$. If V_{ij} is not backlogged at t_1, then $S\Delta$ bits that precede V_{ij} at t_1, leave input i during $[t_1, t]$. This implies that $\underline{margin}_{ij}(t) \geq \underline{margin}_{ij}(t_1) + (S-2)\Delta \geq C + (S-2)\Delta$.

Next, assume that $r > 0$, let V_{ih} be the VOQ in P that comes latest in the VOQ ordering at t_1, let $k = p_{ih}(t_1, [0, s_{ij})) - p_{ij}(t_1, [0, s_{ij}))$, and note that $r \leq k$. Also, note that since

$$\overline{p}_{ij}(t_1) = p_{ij}(t_1, [0, s_{ij})) + \overline{p}_i(t_1, [s_{ij}, t_1])$$

and

$$\overline{p}_{ih}(t_1) = p_{ih}(t_1, [0, s_{ih})) + \overline{p}_i(t_1, [s_{ih}, t_1]) \geq p_{ih}(t_1, [0, s_{ij})) + \overline{p}_i(t_1, [s_{ij}, t_1])$$

it follows that, $k \leq \overline{p}_{ih}(t_1) - \overline{p}_{ij}(t_1)$.

Since V_{ih} passes V_{ij}, output h must receive fewer bits than output j does during $[t_1, t]$, and since output j can receive no more than $S\Delta$ bits during $[t_1, t]$, output h receives fewer than $S\Delta$. This implies that $B_{ih}(t_1) < L_M \leq B - L_M$. Consequently, V_{ih} is eligible for selection at t_1, which implies that the packet x that is selected at t_1 precedes V_{ih}.

We consider three cases. First, if x arrived after s_{ij} then, the departure of x reduces $\overline{p}_i(t_1, [s_{ij}, t_1])$ by $S\Delta$. Consequently,

$$\overline{p}_{ij}(t) \leq \overline{p}_{ij}(t_1) + r + \Delta - S\Delta \leq \overline{p}_{ij}(t_1) + k - (S-1)\Delta$$

Similarly, if x arrived before s_{ij} and x precedes V_{ij} at t_1 then the departure of x reduces $p_{ij}(t_1, [0, s_{ij}))$ by $S\Delta$. Consequently,

$$\overline{p}_{ij}(t) \leq \overline{p}_{ij}(t_1) + r + \Delta - S\Delta \leq \overline{p}_{ij}(t_1) + k - (S-1)\Delta$$

Finally, if x arrived before s_{ij} and x does not precede V_{ij} at t_1 then $r + S\Delta \leq k$ and

$$\overline{p}_{ij}(t) \leq \overline{p}_{ij}(t_1) + r + \Delta \leq \overline{p}_{ij}(t_1) + k - (S-1)\Delta$$

So, in all three cases $\overline{p}_{ij}(t) \leq \overline{p}_{ij}(t_1) + k - (S-1)\Delta$. Since V_{ih} passes V_{ij}, $q_{ij}(t) > q_{ih}(t) \geq q_{ih}(t_1) - \Delta$ and so,

$$\begin{aligned}
\underline{margin}_{ij}(t) &> (q_{ih}(t_1) - \Delta) - (\overline{p}_{ij}(t_1) + k - (S-1)\Delta) \\
&\geq (q_{ih}(t_1) - \overline{p}_{ih}(t_1)) + (S-2)\Delta \\
&= \underline{margin}_{ih}(t_1) + (S-2)\Delta \\
&\geq C + (S-2)\Delta
\end{aligned}$$

■

Our next lemma establishes a lower bound on $\underline{margin}$ for all mature VOQs.

Lemma 15 *Given a crossbar with $S \geq 2$, $B \geq \max\{3+2/(S-1), (1/(S-1))(2S+(3/2)+2/(S-1))\}L_M$ and a RPLF scheduler. If V_{ij} is mature at time t, $\underline{\mathit{margin}}_{ij}(t) \geq L_M/S$.*

proof. Note that at the start of a non-empty period at input i, no VOQs are mature. So initially, all mature VOQs have $\underline{\mathit{margin}} \geq L_M/S$. Assume then that t is the earliest time in the non-empty period when there is some mature VOQ with $\underline{\mathit{margin}} < L_M/S$ and let V_{ij} be the oldest VOQ for which $\underline{\mathit{margin}}_{ij} < L_M/S$ at time t. Let $t_1 < t$ be the time of the most recent event at input i before t. Suppose first that V_{ij} was mature at t_1. Since $t_1 < t$, this implies that $\underline{\mathit{margin}}_{ij}(t_1) \geq L_M/S$. Since all VOQs older than V_{ij} are also mature at t_1, then these all satisfy $\underline{\mathit{margin}} \geq L_M/S$ at t_1. Consequently, by Lemma 14, $\underline{\mathit{margin}}_{ij}(t) \geq L_M/S$.

Assume then that V_{ij} was not mature at t_1. This implies that

$$t_1 < f_{ij} + T_M \leq t \leq t_1 + L_M/S < f_{ij} + T_M + L_M/S$$

We divide the remainder of the analysis into three cases.

Case 1. $p_{ij}(t,[0,s_{ij})) = 0$ and V_{ij} was backlogged at τ_{ij}. Since V_{ij} was backlogged at τ_{ij}, by Lemma 4 $q_{ij}(\tau_{ij}) \geq (1-1/S)(B-2L_M) + (S-1)(\tau_{ij} - f_{ij})$ and q_{ij} continues to grows at rate $(S-1)$ until $\tau_{ij} + (B-L_M)/S$.

If $t \leq \tau_{ij} + (B-L_M)/S$, $q_{ij}(t) \geq (1-1/S)(B-2L_M) + (S-1)(t-f_{ij})$ and since $p_{ij}(t,[0,s_{ij})) = 0$, $\overline{p}_{ij}(t) \leq t - s_{ij}$. Hence

$$\begin{aligned}
\underline{\mathit{margin}}_{ij}(t) &\geq (1-1/S)(B-2L_M) + (S-1)(t-f_{ij}) - (t-(f_{ij}-L_{ij}))\\
&\geq (1-1/S)(B-2L_M) + (S-2)(t-f_{ij}) - L_M))\\
&\geq (1-1/S)B - (3-2/S)L_M
\end{aligned}$$

This is $\geq L_M/S$, so long as $B \geq (3+2/(S-1))L_M$, which is implied by the condition on B in the statement of the lemma.

Assume then that $t > \tau_{ij} + (B-L_M)/S$. In this case,

$$\begin{aligned}
q_{ij}(t) &\geq (1-1/S)(2B-3L_M) + (S-1)(\tau_{ij}-f_{ij}) - (t-(\tau_{ij}+L_M/S))\\
&\geq (1-1/S)(2B-3L_M) + -((f_{ij}+T_M+L_M/S) - (\tau_{ij}+L_M/S))\\
&= (1-1/S)(2B-3L_M) + (\tau_{ij}-f_{ij}) - 2L_M/(S-1)\\
&\geq 2(1-1/S)B - (3+2/(S-1)-3/S)L_M
\end{aligned}$$

and

$$\begin{aligned}
\underline{\mathit{margin}}_{ij}(t) &\geq 2(1-1/S)B - (3+2/(S-1)-3/S)L_M - (t-s_{ij})\\
&\geq 2(1-1/S)B - (3+2/(S-1)-3/S)L_M\\
&\qquad -((f_{ij}+T_M+L_M/S) - (f_{ij}-L_{ij}))\\
&\geq 2(1-1/S)B - (3+2/(S-1)-3/S)L_M - ((2/(S-1)) + (1+1/S))L_M\\
&= 2(1-1/S)B - (4+4/(S-1)-2S)L_M
\end{aligned}$$

This is $\geq L_M/S$ if $B \geq (1/(S-1))[2S+(3/2)+2/(S-1)]L_M$, which is implied by the condition in the statement of the lemma. This completes Case 1.

Case 2. $p_{ij}(t, [0, s_{ij})) = 0$ and V_{ij} is not backlogged at τ_{ij}. Suppose there is no backlog event for V_{ij} in $[\tau_{ij}, t]$. Then,

$$\begin{array}{rcl} \overline{p}_{ij}(t) & \leq & L_{ij} + (\tau_{ij} - f_{ij}) - (S-1)(t - \tau_{ij}) \\ & < & L_{ij} + L_M/S - (S-1)(T_M - L_M/S) \\ & \leq & 2L_M - (S-1)T_M \leq 0 \end{array}$$

This contradicts the fact that V_{ij} is active at t, so there must be some backlog event in $[\tau_{ij}, t]$. Note that $\beta_{ij} \geq \tau_{ij} + L_{ij}/S$, so

$$\begin{array}{rcl} \overline{p}_{ij}(\beta_{ij}) & \leq & L_{ij} + (\tau_{ij} - f_{ij}) - (S-1)(\beta_{ij} - \tau_{ij}) \\ & < & L_{ij} + L_M/S - (1 - 1/S)L_{ij} \\ & = & L_{ij}/S + L_M/S \leq 2L_M/S \end{array}$$

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

$$\underline{margin}_{ij}(\beta_{ij}) \geq (1 - 1/S)(B - 2L_M) - 2L_M/S = (1 - 1/S)B - 2L_M$$

which 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. If $\beta_{ij} + (B - L_M)/S \geq t$, then q_{ij} continues to grow at rate $(S-1)$ until t. This is enough to compensate for any growth in $\overline{p}_{ij}$. Hence, $\underline{margin}_{ij}(t) \geq L_M/S$.

Assume then that $\beta_{ij} + (B - L_M)/S < t$. In this case, q_{ij} continues to grow at rate $(S-1)$ until $\beta_{ij} + (B - L_M)/S$ giving $q_{ij}(\beta_{ij} + (B - L_M)/S) > (1 - 1/S)(2B - 3L_M)$. Thus,

$$\begin{array}{rcl} q_{ij}(t) & > & 2(1 - 1/S)B - 3(1 - 1/S)L_M - (t - (\beta_{ij} + (B - L_M)/S)) \\ & \geq & (2 - 1/S)B - (3 - 2/S)L_M - (t - \beta_{ij}) \end{array}$$

and since $\overline{p}_{ij}(t) \leq t - s_{ij}$.

$$\begin{array}{rcl} \underline{margin}_{ij}(t) & \geq & (2 - 1/S)B - (3 - 2/S)L_M - (t - \beta_{ij}) - (t - s_{ij}) \\ & \geq & (2 - 1/S)B - (3 - 2/S)L_M - 2(f_{ij} + T_M + L_M/S) + \beta_{ij} + s_{ij} \\ & \geq & (2 - 1/S)B - (3 - 2/S)L_M - 2(2/(S-1)L_M + L_M/S) \\ & & \qquad + (\tau_{ij} + L_{ij}/S) - f_{ij} - L_{ij} \\ & \geq & (2 - 1/S)B - (3 - 2/S)L_M - 2(2/(S-1)L_M + L_M/S) - (1 - 1/S)L_{ij} \\ & \geq & (2 - 1/S)B - (4 + 4/(S-1) - 1/S)L_M \end{array}$$

which is $\geq L_M/S$ if $B \geq (4S/(2S-1))(1 + 1/(S-1))L_M$, which is implied by the condition on B in the statement of the lemma. This completes Case 2.

Case 3. $p_{ij}(t, [0, s_{ij})) > 0$. This implies that there is some VOQ that precedes V_{ij} at t and is older than V_{ij}. Let V_{ih} be one such VOQ and assume further, that among all such VOQs, it comes latest in the VOQ ordering at t. Note that since V_{ij} is the oldest VOQ with $\underline{margin} < L_M/S$ at t, $\underline{margin}_{ih}(t) \geq L_M/S$.

Let K be the set of bits that precede V_{ij} at time t but not V_{ih} and let $k = |K| = p_{ij}(t) - p_{ih}(t)$. Note that all bits in K 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 R be the set of bits that arrived after s_{ij} and are still present at time t and do not precede V_{ij} and let $r = |R| = \overline{p}_{ij}(t) - p_{ij}(t)$. Note that K and R have no bits in common. Now, let X be the set of bits that arrived since s_{ih} and do not precede V_{ih}. Note that both K and R are subsets of X and so $k + r \leq |X| = \overline{p}_{ih}(t) - p_{ih}(t)$. Consequently,

$$margin_{ih}(t) = q_{ih}(t) - p_{ih}(t) \geq q_{ih}(t) - \overline{p}_{ih}(t) + (k + r) \geq L_M/S + (k + r)$$

and

$$\underline{margin}_{ij}(t) = q_{ij}(t) - (p_{ij}(t) + r) \geq q_{ih}(t) - ((p_{ih}(t) + k) + r)) \geq L_M/S$$

■

Note that Lemma 15 implies that that $margin_{ij}(t) \geq L_M/S$ for all mature V_{ij}. From this we obtain the emulation theorem for 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 $S \geq 2$, $B \geq \max\{3 + 2/(S - 1), (1/(S-1))(2S + (3/2) + 2/(S-1))\}L_M$ and $T \geq T_M + L_M/S$.*

proof. 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 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 therefore 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, it follows that $f(x) \leq t - T$ and so $t - L_M/S \geq f(x) + T_M$. 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 Lemma 15, $margin_{ij}(t_1) \geq L_M/S$ and hence $p_{ij}(t_1) < 0$, which is not possible. ■

7 Segment-Based Switching

As mentioned earlier, Chuang, et. al. [2] showed that cell-based crossbars can emulate an output-queued switch using any push-in, first-out (PIFO) queueing discipline. It is straight-forward 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 [17].) 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. To achieve the desired performance guarantees in the worst-case, one must still double the speedup implied by the idealized analysis, significantly adding to the system cost.

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. A buffer large enough to hold the 7.5 maximum size packets needed to emulate any restricted PIFO discipline using RPLF would require close to 11 MB. In [8], 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 the required buffer size is reduced 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 line card before all segments of the packet have been received. The performance guarantee for the crossbar will ensure that 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 not only reduce the amount of buffering required, but we reduce the delay as well.

8 Concluding Remarks

The results of sections 5 and 6 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. Buffering packets at the inputs allows error checks to be performed on the packets before forwarding them to the switch. Buffering them at the outputs allows similar checks to be performed, but is arguably less essential, since packet errors are less likely to occur within a router than on the external links. Having said that, other considerations may dictate that packets be buffered at outputs, as well as inputs and this raises the question of how the performance guarantees are affected. It turns out that the effect is fairly minor, requiring 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 [5] and can provide superior delay performance. While cut-through switching is not typically used in routers, it can be useful in system contexts where it is important to minimize latency. 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 discussed in the previous section already approach the behavior of a cut-through switch, and there seems little reason to suppose that the results would not generalize to the cut-through model. The key requirement needed to obtain work-conservation is that once a packet has been selected to advance from an input line card to the crossbar or from the crossbar to an output line card, the flow of bits in that packet must not be interrupted until the end of the packet is reached. Inputs (outputs) must also be able to forward multiple packets to (from) the crossbar concurrently in certain cases. Consider for example, an input that is forwarding bits of a packet x to the crossbar as they come in. Since the bits are arriving at the link rate, the transfer of the bits of x to the crossbar uses only half the crossbar bandwidth (assuming $S = 2$). If another packet y at the input becomes eligible for forwarding while x is still coming in (because its crossbar buffer has drained sufficiently to accommodate it), the input must be able to forward y to the crossbar concurrently with x in order to fully exploit the crossbar bandwidth. Without the ability to transfer packets concurrently to and from the crossbar, it will not be possible to achieve work-conservation.

There are several ways the work described here can be extended. First, there are opportunities for tightening the results shown here, particularly with respect to the crossbar buffer size. Our analysis showing that a PGV scheduler can emulate an output-queued switch with a restricted PIFO scheduler requires a buffer size of $5L_M$. As noted earlier, it seems likely that this can be reduced to $3L_M$. The buffer size results for PLF are also not as strong as one might expect. There seems no intrinsic reason to suppose that PLF requires a larger crossbar buffer size than PGV. An analysis that directly compares the behavior of a PLF scheduler to the PGV scheduler may be able to reduce the buffer size requirement for PLF. Another worthwhile direction for further work is developing performance guarantees for other scheduling algorithms.

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 [3] describes several techniques that can be used to allow cell switches using buffered crossbars to overcome this crosspoint blocking phenomenon. One involves increasing the speedup and allowing later arriving packets to displace packets already in crossbar buffers. Another method requires no increase in speedup, but uses a more complex form of buffering in the crossbar. 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] Anderson, T., S. Owicki., J. Saxe and C. Thacker. "High speed switch scheduling for local area networks," *ACM Transactions on Computer Systems*, 11/1993.

[2] Chuang, S.-T. A. Goel, N. McKeown, B. Prabhakar "Matching output queueing with a combined input output queued switch," *IEEE Journal on Selected Areas in Communications*, 12/1999.

[3] Chuang, Shang-Tse, Sundar Iyer, Nick McKeown. Practical Algorithms for "Performance Guarantees in Buffered Crossbars," *Proceedings of IEEE INFOCOM*, 3/2005.

[4] Iyer, S., R. Zhang, and N. McKeown, "Routers with a Single Stage of Buffering," *Proceedings of ACM SIGCOMM*, 9/2002.

[5] Kermani, Parviz and Leonard Kleinrock. "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks*, 267–286, 1979.

[6] Krishna, P., N. Patel, A. Charny and R. Simcoe. "On the speedup required for work-conserving crossbar switches," *IEEE J. Selected Areas of Communications*, 6/1999.

[7] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, N. Chrysos. "Variable Packet Size Buffered Crossbar (CICQ) Switches," *Proceedings IEEE International Conference on Communications*, pp. 1090-1096, 6/2004.

[8] M. Katevenis, G. Passas. "Variable-Size Multipacket Segments in Buffered Crossbar (CICQ) Architectures," *Proceedings IEEE International Conference on Communications*, 5/2005.

[9] Leonardi, E., M. Mellia, F. Neri, and M.A. Marsan, "On the stability of input-queued switches with speed-up," *IEEE/ACM Transactions on Networking*, Vol. 9, No. 1, pp. 104–118, 2/2001.

[10] B. Magill, C. Rohrs, R. Stevenson, "Output-Queued Switch Emulation by Fabrics With Limited Memory," *IEEE Journal on Selected Areas in Communications*, pp. 606–615, 5/2003.

[11] McKeown, Nick. iSLIP: a scheduling algorithm for input-queued switches," *IEEE Transactions on Networking*, 4/1999.

[12] McKeown, N., A. Mekkittikul, V. Anantharam, and J. Walrand. "Achieving 100*IEEE Transactions on Communications*, Vol. 47, No. 8, 8/1999.

[13] Mhamdi, L., Mounir Hamdi. "MCBF: A High-Performance Scheduling Algorithm for Buffered Crossbar Switches," *IEEE Communications Letters*, 2003.

[14] Nojima, S., E. Tsutsui, H. Fukuda, M.Hashimoto. "Integrated Services Packet Network Using Bus Matrix Switch," *IEEE Journal on Selected Areas of Communications*, 10/87.

[15] Rodeheffer, Thomas L. and James B. Saxe. "An Efficient Matching Algorithm for a High-Throughput, Low-Latency Data Switch," Compaq Systems Research Center, Research Report 162, 11/5/98.

[16] Rojas-Cessa, E. Oki, Z. Jing, and H. J. Chao, "CIXB-1: Combined Input-One-cell-Crosspoint Buffered Switch," *IEEE Workshop on High Performance Switching and Routing*, 7/2001.

[17] Turner, Jonathan. "When is a Work-Conserving Switch Not?" Washington University Computer Science and Engineering Technial Report: WUCSE-2005-14, 4/2005.

[18] Turner, Jonathan. "Strong Performance Guarantees for Asynchronous Crossbar Schedulers," *Proceedings of Infocom*, 2006.