Strong Performance Guarantees for Asynchronous Crossbar Schedulers

Jonathan Turner

Washington University
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 schedulers may fail to deliver the expected performance guarantees when used in routers that forward packets of variable length. 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 speedups ≥2. 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 [9]. Recently, there has been a growing body of work providing rigorous performance guarantees for such systems [7,12] 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 non-blocking 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. (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 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 practi-

cal 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 [12]. 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 [3,4,8,9,10,12,16]. 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 [8,9] 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 provides a more detailed discussion of the issue of switching cells vs. packets. Our main results are given in sections 3 and 4. Section 3, contains results for an asynchronous version of GVOQ and section 4 presents results for an asynchronous version of LOOFA. In section 5, we discuss the impact of buffering on crossbar implementations and show that there are schedulers for *segment-based crossbars* [9] that can achieve strong performance guarantees with fairly modest buffer requirements and no bandwidth fragmentation. Section 6 concludes the paper.

2. 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 in 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 cannot typically begin transmission of a packet on its outgoing link until 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 un-

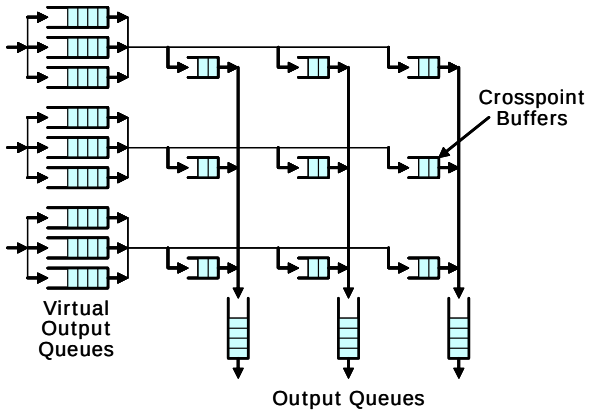

Figure 1. Packet switch with unbuffered crossbar

avoidable, 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*. Chuang, et. al. [2] showed that cell-based crossbars can emulate an output-queued switch using any *push-in, first-out* (PIFO) queueing discipline. A PIFO queueing discipline is one in which the relative transmission order of the cells awaiting transmission at any time is predictable. That is, the relative transmission order of cells currently in the queue is not affected by future arrivals. Such a queueing discipline can be implemented by inserting arriving cells into a list. Arriving cells may be inserted in the list at any point, but departing cells are always taken from the front of the list. 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 predecessor). This makes it possible to provide strong performance guarantees for packets, not just cells. (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 double the speedup implied by the idealized analysis, significantly adding to the system cost.

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. PACKET GVOQ

In this section, we show that a well-known cell-switching scheduler can be converted into an asynchro-

nous crossbar scheduler with comparable worst-case performance.

3.1 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 is present at input i and is to be forwarded to 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 a *scheduling event* or more simply, an *event*. 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 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 say that a given asynchronous crossbar scheduler is *T-work-conserving* for a given speedup S and crosspoint buffer size B, if whenever there is an idle output link, no input contains a packet x for that output link for which $f(x)+T$ is less than the current time. That is, a T-work conserving scheduler allows an output link to be idle only so long as there are no packets present for that output that are "older" than T.

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. We say that a VOQ is *active*, if the last bit of the first packet in the VOQ has been received from the external link. Otherwise, it is *inactive*. Note that a VOQ can become inactive, even while it remains non-empty. For an active VOQ, we refer to the time period since it last became active as the *current active period* and for VOQ V_{ij}, we let $s_{ij}(t)=s(x)$, where x is the first packet received by V_{ij} in the current active period at time t.

Once a packet has been selected by an input line card for transmission to the crossbar, it is sent at the rate allowed by the system's speedup S. Similarly, once an output line card selects a packet from a crosspoint buffer, it transfers bits from the crosspoint buffer at the rate allowed by the speedup, until the packet is fully transferred. Packets may be streamed through the crosspoint buffer without fully buffering them and may be forwarded by an output line card to the external link as soon as the first bit is received by the output line card. Since the speedup is at least 1, the output line card is guaranteed that once it receives the first bit, the remaining bits will arrive in time to be sent on the outgoing link. (Note, this property is not shared by systems that use cell-based schedulers that are not packet-aware, forcing those systems to wait until the last cell of a packet has been received before starting transmission of a packet on the outgoing link.)

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 that is complete (all bits received from the input link) and the crosspoint buffer for x has room for it, 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*.

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) packet switch scheduling algorithm as follows. The algorithm imposes a total order on the active VOQs at each input. The relative order of two VOQs does not change so long as they both remain active. When a VOQ becomes active, it is placed first in the VOQ ordering. When a VOQ becomes inactive, it is removed from the VOQ ordering. At each event, the scheduler selects some active VOQ for which the crosspoint buffer has enough space to accommodate the first packet in the VOQ. If multiple VOQs are eligible under this criterion, it selects the VOQ that comes first in the ordering.

We say that one of two active VOQs *precedes* another if it comes before the other in the VOQ ordering. We extend the VOQ ordering to apply to packets and to individual bits in the VOQs. Packets (and bits) in different VOQs are ordered according to the VOQ ordering, while those in the same VOQ are ordered according to their position within the VOQ. We also say that a packet (or bit) x precedes a VOQ V if x is in V, or in some VOQ that precedes V.

3.2 Work-Conservation

We first prove a work-conservation result for PGV. This result does not depend on the specific policy used by the output line card to select a crosspoint buffer, so we leave the output policy undefined here. Hence, the result applies to a class of PGV schedulers with a variety of specific instantiations.

For an active VOQ V_{ij}, we let $p_{ij}(t)$ be the number of bits in VOQs at input i that precede V_{ij} at time t (this includes bits in V_{ij}). To simplify the analysis, we also define $\tilde{p}_{ij}(t)$ to be the number of bits present at input i at time t that have arrived since s_{ij}. Note that $\tilde{p}_{ij}(t)$ may include bits from a packet that is currently arriving on the input link but has not yet been fully received. Also, note that any VOQ that became non-empty after x last became active, is either not yet active or became active after V_{ij}. Furthermore, any VOQ that last become non-empty before V_{ij} last became active, also became active before V_{ij}. Hence, $p_{ij}(t) \le \tilde{p}_{ij}(t)$. So if $\tilde{p}_{ij}(t)=V_{ij}(t)$, then V_{ij} is first in the VOQ ordering at input i. We define $q_j(t)$ to be the number of bits at output j at time t, we define $slack_{ij}(t)=q_j(t)-p_{ij}(t)$ and we define $\sigma_{ij}(t)=q_j(t)-\tilde{p}_{ij}(t)$. Note that $slack_{ij}(t) \ge \sigma_{ij}(t)$.

In the remainder of this section, we show that PGV schedulers are T-work-conserving when $S \ge 2$, $B \ge 2L_M$ and $T \ge 2L_M$. However, before we get into the details of the analysis, we give an overview the overall argument, to provide some intuition for the more technical points to come. The first major step in the analysis is to show that σ_{ij} does not decrease with time. We then use this to show that any VOQ that has been active for longer than T time units has $\sigma_{ij}>0$ and hence it has positive *slack*.

We can establish both these properties by making a few observations about the behavior of the crosspoint buffers when $S>1$. First, during any time period when a crosspoint buffer B_{ij} is non-empty, q_j must increase. Second, during any time period when B_{ij} is not too full to accept a new packet, p_{ij} must decrease and hence $\tilde{p}_{ij}$ must decrease. Note also, that there is a minimum time before B_{ij} becomes too full for a new packet, that it must be non-empty. Similarly, it remains non-empty for a minimum time period after it ceases to be too full to accept a new packet. Consequently, every time the status of B_{ij} changes between being too full for a new packet and not being too full, there is an increase in σ_{ij}. The duration of these periods is directly related to the size of the crosspoint buffer. These points are illustrated in Figure 2.

The asynchronous nature of the crossbar means that at the start of an active period, σ_{ij} can be negative. We can overcome this initial deficit using the fact that σ_{ij} increases when the crosspoint buffer becomes too full to accept a new packet and the fact that the crosspoint buffer must become too full to accept a new packet within T time units of the start of an active period.

The final step in the analysis establishes the T-work-conservation property using the fact that shortly after a VOQ becomes active, it has positive slack. Turning these high level ideas into an actual proof of work-conservation requires that we first formulate and prove two lemmas.

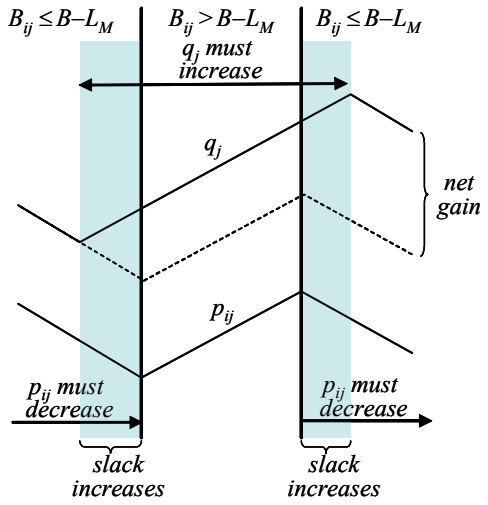

Figure 2. Gain in slack due to changes in crosspoint buffer status

Lemma 1. Consider an active period of V_{ij} that spans the time interval $[t_1,t_2]$ where $t_1 \ge$ the time of the first scheduling event of the period. For any PGV scheduler with speedup $S \ge 2$ and $B \ge 2L_M$, $\sigma_{ij}(t_2) \ge \sigma_{ij}(t_1)$.

proof. First, note that during any time period, $\tilde{p}_{ij}$ can increase by at most the duration of the time period, due to new bits arriving at input i at the link rate. Similarly, q_j can decrease by at most the duration of the time period, as bits are sent on the outgoing link at the link rate. The resulting decrease in σ_{ij} can be offset by increases caused by the transfer of bits from input i to the crossbar or from the crossbar to output j.

For now, let t_1 and t_2 be the times of consecutive events at input i. Let x be the first packet in V_{ij} at time t_1. If, at time t_1, B_{ij} does not have room for x, then it contains at least L_M bits. This means that output j will be transferring bits from some crosspoint buffer for at least the next L_M/S time units at rate S, by which time the next event at input i must have occurred. Since $S \ge 2$, this is sufficient to offset any decrease in σ_{ij} caused by new arrivals at input i or departures from output j. On the other hand, if B_{ij} does have room for x at time t_1, then either x must be selected for transmission to the crossbar or some packet that precedes x must be selected. In either case, the transfer of bits to the crossbar offsets the effects of new arrivals and departures. Hence σ_{ij} does not decrease.

By induction, this generalizes to t_1 and t_2 equal to the times of any scheduling events at input i. It also holds between any two times that follow the first scheduling event of the active period, since the change

in σ_{ij} between two events is determined by the situation at the first event. ■

Lemma 2. Let x be the first packet to be received at the start of an active period for V_{ij} and let $t \geq f(x)+2L_M$ belong to the active period in which x arrives. For any PGV scheduler with speedup S=2 and $B \geq 3L_M$, $slack_{ij}(t) \geq \sigma_{ij}(t) > 0$.

proof. Let $\tau < f(x)+L_M/2$ be the time of the first scheduling event at input i in the active period. Note that if $B_{ij}(\tau) > B-L_M$ then $B_{ij}(f(x)) > B-L_M$ and B_{ij} has been non-empty since before $f(x)-(B-L_M)/2 \leq s(x)$. This implies that q_j has been increasing at rate 1 since before $s(x)$, so $q_j(\tau) > L(x)+(\tau-f(x))$. Since $\tilde{p}_{ij}(f(x))=L(x)$ and $\tilde{p}_{ij}$ can grow at a rate no faster than 1, $\sigma_{ij}(\tau)>0$. By Lemma 1, this remains true for the remainder of the active period.

Now, suppose that $B_{ij}(\tau) \leq B-L_M$. If there is no event in the interval $[\tau, f(x)+2L_M]$ when B_{ij} contains more than $B-L_M$ bits, then at every event in this interval, the scheduling algorithm will select either V_{ij} or a VOQ that precedes V_{ij}. Consequently,

$$\begin{aligned}\tilde{p}_{ij}(f(x)+2L_M) &\leq L(x)+(\tau-f(x))-((f(x)+2L_M)-\tau)\\ &\leq L(x)+2\tau-2f(x)-2L_M\\ &< L(x)+L_M-2L_M\\ &\leq 0\end{aligned}$$

This implies that V_{ij} is empty by $f(x)+2L_M$ contradicting the hypothesis of the lemma. Hence, there must be some event in $[\tau, f(x)+2L_M]$ when B_{ij} contains more than $B-L_M$ bits. Let t_0 be the time of the first such event. So, $B_{ij}(t_0) > B-L_M$. and $t_0 \leq f(x)+2L_M$. Note also that $t_0 > \tau+L(x)/2$ since at least one packet must be transferred to B_{ij} between τ and t_0. This implies that

$$\begin{aligned}\tilde{p}_{ij}(t_0) &\leq L(x)+(\tau-f(x))-(t_0-\tau)\\ &< L(x)+L_M/2-L(x)/2\\ &\leq L_M\end{aligned}$$

Since $B_{ij}(t_0) > B-L_M$, B_{ij} has been non-empty since before $t_0-(B-L_M)/2$, meaning that $q_j(t_0) > (B-L_M)/2 \geq L_M$. Consequently, $\sigma_{ij}(t_0)>0$ and by Lemma 1, this remains true for the remainder of the active period. ■

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

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 $f(x)+2L_M < t$. By Lemma 2, $slack_{ij}(t)>0$. 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.

Theorem 2. Any PGV scheduler with S=2 and $B \geq 2L_M$ is T-work-conserving for $T \geq 2L_M$.

The proof of Theorem 2 requires two additional lemmas. The first is a stronger version of Lemma 1 and the second is a stronger version of Lemma 2.

Lemma 3. Consider an active period of V_{ij} that spans the time interval $[t_1,t_2]$ where $t_1 \geq$ the time of the first scheduling event of the period. For any PGV scheduler with $S \geq 2$ and $B \geq 2L_M$, if $\sigma_{ij}(t_1) > -V_{ij}(t_1)$ then $\sigma_{ij}(t_2) > -V_{ij}(t_2)$.

Lemma 4. Let x be the first packet to be received at the start of an active period for V_{ij} and let $t \geq f(x)+2L_M$ belong to the active period in which x arrives. For any PGV scheduler with speedup S=2 and $B \geq 2L_M$, $slack_{ij}(t) \geq \sigma_{ij}(t) > -V_{ij}(t)$.

The proofs of these lemmas, together with the proof of Theorem 2 can be found in the appendix.

3.3 More general performance guarantees

The analysis of the previous section can be modified to show that some variants of PGV can provide more general performance guarantees.

We say that an asynchronous crossbar *T-emulates* an output-queued switch with a given scheduling policy if it transmits packets at the same time that they would be sent in an output-queued switch that buffers arriving packets at its input, then places them directly into an output queueing system that is followed by a fixed output delay of length T. We show that variants of PGV can T-emulate an output-queued switch using any *restricted* PIFO queueing discipline, where a restricted PIFO queueing discipline is one that forwards the packets from each input in the order they were received at the input (packets from different inputs need not be forwarded in the order of reception). Our result for PGV generalizes the corresponding result for cell-based crossbars given in [3].

The output line cards in the asynchronous crossbar system are assumed to use the same PIFO policy as the ideal output-queued switch that is being emulated. They also select packets from crosspoint buffers in accordance with the PIFO policy. In addition, they do not start to forward packets on outgoing links, until their *age* (the difference between the current time and the time of the arrival of the last bit) is at least equal to a threshold value, T. Note that a system that holds packets until their age is equal to T can still be T-work-conserving. The input line cards order their VOQs in the same way discussed in the previous section. We call the variant of PGV defined in this way PGV-RP, where RP stands for restricted PIFO.

We extend our definition of the *precedes* relation to packets at different inputs, sending to, a common output. In particular, if x and y are packets with a common destination j, we say that x precedes y if the PIFO-RP queueing discipline under consideration transmits x before y. For packets at the same input, this is consistent with our original definition since the PIFO-RP queueing discipline transmits packets going to the same output in their arrival order, as does the FIFO queueing discipline that applies to the individual VOQs. We define $q_{ij}(t)$ to be the number of bits at output j that precede the bits in V_{ij}. We re-define $slack_{ij}(t)=q_{ij}(t)-p_{ij}(t)$ and $\sigma_{ij}(t)=q_{ij}(t)-\tilde{p}_{ij}(t)$.

The key elements of the proof are captured in two lemmas which establish that σ_{ij} does not decrease and that it exceeds a specified minimum value shortly after the start of an active period. The proofs of the lemmas can be found in the appendix. The proofs are similar to the ones used to establish work-conservation. The main difference is that q_{ij} does not necessarily increase whenever the crosspoint buffer B_{ij} is non-empty, since the output may be receiving a "lower priority" packet at the time B_{ij} becomes non-empty. However, q_{ij} is guaranteed to start increasing within $L_M/2$ of the time that B_{ij} becomes non-empty, and this is enough to establish the desired performance guarantee, albeit with some adjustment in the crossbar buffer size.

Lemma 5. Consider an active period of V_{ij} that spans the time interval $[t_1,t_2]$ where $t_1\geq$the time of the first scheduling event of the period. For a PGV-RP scheduler with speedup $S\geq 2$ and $B\geq 2L_M$, $\sigma_{ij}(t_2)\geq\sigma_{ij}(t_1)$.

Lemma 6. Let x be the first packet to be received at the start of an active period for V_{ij} and let $t\geq f(x)+2L_M$ belong to the active period in which x arrives. For any PGV-RP scheduler with speedup $S=2$ and $B\geq 5L_M$, $slack_{ij}(t)\geq\sigma_{ij}(t)> L_M/2$.

Theorem 3. An output-queued switch using any restricted PIFO scheduler can be T-emulated by an asynchronous crossbar with a PGV-RP scheduler with $S=2$, $B\geq 5L_M$ and $T\geq(5/2)L_M$.

proof. Suppose that up until time t, the asynchronous crossbar faithfully emulates the output-queued switch, but that at time t, the output queued switch begins to forward a packet x from input i while the asynchronous crossbar does not. Since the output queued switch has an output delay of T, it follows that $f(x)\leq t-T$, so $t-L_M/2\geq f(x)+2L_M$. Since the crossbar has sent everything sent by the output-queued switch up until t, it follows that $q_{ij}(t-L_M/2)\leq L_M/2$. By Lemma 6, $slack_{ij}(t-L_M/2)>L_M/2$ and hence $p_{ij}(t-L_M/2)<0$, which contradicts the fact that V_{ij} is active at $t-L_M/2$. ■

The analysis of Lemma 6 requires a crossbar buffer of size at least $5L_M$. We conjecture that this can be reduced to $3L_M$ and possibly $2L_M$, using a more sophisticated analysis.

4. 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 imposes a total order on the VOQs at each input, which is extended to an order on all the bits at the input. At each scheduling event, the PLF scheduler selects some VOQ for which the crosspoint buffer has enough space to accommodate the first packet in the VOQ. If multiple VOQs are eligible under this criterion, it selects the VOQ that comes first in the ordering. The work-conservation result we prove below does not depend on the specific policy used by the output to select a crosspoint buffer.

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 each scheduling event, 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.

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 W last became active after V. We say that a VOQ is *mature* at time t if $f(x)\leq t-T$, where x is the first packet received by V_{ij} in its current active period and $T=2L_M$. We define $\pi_{ij}(t)$ to be the number of bits present at input i at time t that precede V_{ij} and that arrived before $s_{ij}(t)$. We define $\tilde{p}_{ij}(t)$ to be $\pi_{ij}(t)$ plus the number of bits present at input i that have arrived since s_{ij}. Finally, we let $slack_{ij}(t)=q_j(t)-p_{ij}(t)$ and $\sigma_{ij}(t)=q_j(t)-\tilde{p}_{ij}(t)$. Note that $p_{ij}(t)\leq\tilde{p}_{ij}(t)$ and $slack_{ij}(t)\geq\sigma_{ij}(t)$.

Our first lemma plays essentially the same role as Lemma 1, in the work-conservation result for PGV.

Lemma 7. Let times t_1 and t_2 be the times of consecutive scheduling events in the same non-empty interval at input i and let M be the set of VOQs that are mature at t_1. For any PLF scheduler with speedup $S \geq 2$ and $B \geq 2L_M$, if all V_{ij} in M satisfy $\sigma_{ij}(t_1) \geq L_M/2$, then all V_{ij} in M that are still mature at t_2 satisfy $\sigma_{ij}(t_2) \geq L_M/2$.

Our next lemma extends Lemma 7 to establish a lower bound on σ for all mature VOQs.

Lemma 8. Let times t_1 and t_2 be the times of consecutive scheduling events in the same non-empty interval at input i and let M be the set of VOQs that are mature at t_1. For any PLF scheduler with speedup $S \geq 2$ and $B \geq 16L_M/3$, if all V_{ij} in M satisfy $\sigma_{ij}(t_1) \geq L_M/2$, then all V_{ij} that are mature at t_2 satisfy $\sigma_{ij}(t_2) \geq L_M/2$.

Note that we can use Lemma 8 to show by induction that $\sigma_{ij}(t) \geq L_M/2$, if V_{ij} is mature at time t, where t is the time of some event at input i. The basis of the induction is trivially satisfied, since, at the time τ of the first scheduling event following a period when all VOQs at input i are empty, there are no mature VOQs, hence all mature VOQs satisfy $slack_{ij}(\tau) \geq \sigma_{ij}(\tau) \geq L_M/2$.

Theorem 4. Any PLF scheduler with $S=2$ and $B \geq 16L_M/3$ is T-work-conserving for $T \geq 5L_M/2$.

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 $f(x)+T<t$. This implies that at the time τ of the most recent event at input i, V_{ij} was mature and this means (by the discussion following Lemma 8) that $slack_{ij}(\tau) \geq L_M/2$. Since $q_j(t)=0$, $q_j(\tau) \leq L_M/2$, but this implies that $p_{ij}(\tau) \leq 0$, which contradicts the assumption that V_{ij} is not empty. ■

By combining the analyses used to prove Theorems 3 and 4, we can obtain the following result for which the proof is omitted.

Theorem 5. An output-queued switch using any restricted PIFO scheduler can be T-emulated by an asynchronous crossbar with a PLF-RP scheduler with $S=2$, $B \geq 22L_M/3$ and $T \geq 3L_M$.

5. COST OF CROSSBAR BUFFERS

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. While this is a substantial amount for on-chip memory, it falls within the range of what is currently feasible, as has been demonstrated recently by Katevenis et. al. [8]. Moreover, high performance crossbars are often implemented using multiple crossbar components operating in parallel. The buffer space required by each such component is thus reduced in proportion to the number of parallel components. For a system designed to support 40 Gb/s external links, a typical design might use 16 to 32 chips operating in parallel. This reduces the memory requirement per chip to about 100 to 200 KB. This is a fairly modest requirement and opens up the possibility of handling larger packets.

It is also possible to substantially reduce the size of the crossbar buffers by switching variable length *segments* rather than complete packets [9]. In section 2, we observed that a packet-aware variant of a GVOQ scheduler for cell-based crossbars can be used to provide strong performance guarantees for packets. We can apply the same approach to a system in which packets are divided into segments of variable length before sending them through the crossbar. If segment lengths vary from the minimum packet size to at least twice the minimum packet size, we can divide a packet into segments with no "wasted space". We can then forward the segments through an asynchronous crossbar using a packet-aware scheduling algorithm, which keeps the segments of a packet together. This requires that the PIFO scheduling policy at the outputs, insert segments of a packet into consecutive positions in the output list. It also requires that the output scheduling policy for the crossbar select the next segment based on where the segment goes in the output list.

The size of the crossbar buffers in such a system is a function of the maximum segment size, not the maximum packet size. A 32 port crossbar with a minimum segment size of 50 bytes, a maximum segment size of 100 bytes and crossbar buffers large enough for two maximum length segments, would require a total of 200 KB of memory. If the crossbar were implemented using 16 components in parallel, the amount of memory per component drops to about 12.5 KB. Moreover, such a crossbar can be used in systems switching packets of arbitrary length. If we dimension the crosspoint buffer size to hold eight maximum length segments (allowing us to use a PLF-RP scheduler), the amount of memory per component grows only to 50 KB.

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.

6. CONCLUDING REMARKS

The results of sections 3 and 4 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 $L_M/2$, 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 over-come 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 Trans. on Computer Systems*, 11/93.

[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*, 12/99.

[3] Chuang, Shang-Tse, Sundar Iyer, Nick McKeown. "Practical Algorithms for Performance Guarantees in Buffered Crossbars," *Proceedings of IEEE INFOCOM*, 3/05.

[4] Iyer, S., R. Zhang, and N. McKeown, "Routers with a Single Stage of Buffering", *ACM SIGCOMM '02*, Pittsburgh, USA, Sep. 2002.

[5] Kermani, Parviz and Leonard Kleinrock. "Virtual Cut-Through: A New Computer Communication Switching Technique." *Computer Networks* 3: 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*, 6/99.

[7] 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, February 2001.

[8] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, N. Chrysos: "Variable Packet Size Buffered Crossbar (CICQ) Switches", *Proc. IEEE International Conference on Communications (ICC 2004)*, Paris, France, 20-24 June 2004, vol. 2, pp. 1090-1096.

[9] M. Katevenis, G. Passas: "Variable-Size Multipacket Segments in Buffered Crossbar (CICQ) Architectures", *Proc. IEEE International Conference on Communications (ICC 2005)*, Seoul, Korea, 16-20 May 2005.

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

[11] McKeown, Nick. "iSLIP: a scheduling algorithm for input-queued switches," *IEEE Transactions on Networking*, 4/99.

[12] McKeown, N., A. Mekkittikul, V. Anantharam, and J. Walrand. "Achieving 100% Throughput in an Input-Queued Switch," *IEEE Transactions on Communications*, Vol. 47, No. 8, Aug. 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*, Dallas, TX, July 2001.

[17] Turner, Jonathan. "When is a Work-Conserving Switch Not?" Washington University Computer Science and Engineering Technial Report, WUCSE-2005-14, 4/05.

APPENDIX

Here, we collect various proofs that were omitted from the main body of the paper. Lemmas 3 and 4 are used to prove Theorem 2, the stronger version of the work-conservation result for PGV.

Lemma 3. Consider an active period of V_{ij} that spans the time interval $[t_1,t_2]$ where $t_1 \geq$the time of the first scheduling event of the period. For any PGV scheduler with $S \geq 2$ and $B \geq 2L_M$, if $\sigma_{ij}(t_1) > -V_{ij}(t_1)$ then $\sigma_{ij}(t_2) > -V_{ij}(t_2)$.

proof. If $V_{ij}(t_2) \geq V_{ij}(t_1)$ then the result follows from Lemma 1. Assume then that $V_{ij}(t_2) < V_{ij}(t_1)$. Let t_1 and t_2 be the times of consecutive events at input i. $V_{ij}(t_2) < V_{ij}(t_1)$ implies that V_{ij} was selected at t_1. This means that during the interval $[t_1,t_2]$, q_j is increasing at rate ≥ 1, while p_{ij} and $\tilde{p}_{ij}$ are decreasing at rate ≥ 1. This results in a net increase in σ_{ij} at least equal to the decrease in V_{ij}. Consequently, $\sigma_{ij}(t_2) > -V_{ij}(t_2)$. The result generalizes to any interval following the first scheduling event of the active period, by the same argument used in the proof of Lemma 1. ■

Lemma 4. Let x be the first packet to be received at the start of an active period for V_{ij} and let $t \geq f(x)+2L_M$ belong to the active period in which x arrives. For any PGV scheduler with speedup $S=2$ and $B \geq 2L_M$, $slack_{ij}(t) \geq \sigma_{ij}(t) > -V_{ij}(t)$.

proof. Let $\tau < f(x)+L_M/2$ be the time of the first scheduling event at input i in the active period. Note that if $B_{ij}(\tau) > 0$ then B_{ij} has been positive since before $f(x)$. This implies that q_j has been increasing at rate ≥ 1 since before $f(x)$, so $q_j(\tau) > (\tau - f(x))$. Since $\tilde{p}_{ij}(\tau) - V_{ij}(\tau) \leq (\tau - f(x))$, it follows that $\sigma_{ij}(\tau) > -V_{ij}(\tau)$. By Lemma 3, this remains true for the remainder of the active period.

Now, suppose that $B_{ij}(\tau)=0$. Define $z(t)$ to be the first packet in V_{ij} at time t. If there is no event in the interval $[\tau, f(x)+2L_M]$ when B_{ij} contains more than $B-L(z(t))$ bits, then at every event in this interval, the scheduling algorithm will select either V_{ij} or some other VOQ that precedes V_{ij}. Consequently,

$$\begin{aligned}\tilde{p}_{ij}(f(x)+2L_M) &\leq L(x) + (\tau - f(x)) \\ &\quad - ((f(x)+2L_M) - \tau) \\ &\leq L(x) + 2(\tau - f(x)) - 2L_M \\ &< 0\end{aligned}$$

This implies that V_{ij} is empty by $f(x)+2L_M$ contradicting the hypothesis of the lemma. Hence, there must be some event in $[\tau, f(x)+2L_M]$ when B_{ij} contains more than $B-L(z(t))$ bits. Let t_0 be the time of the first such event. So, $B_{ij}(t_0) > B-L(z(t_0))$. and $t_0 \leq f(x)+2L_M$. Note also that $t_0 > \tau + (B-L(z(t)))/2$ since $B_{ij}(\tau)=0$. This implies that

$$\begin{aligned}\tilde{p}_{ij}(t_0) &\le L(x) + (\tau - f(x)) - (t_0 - \tau) \\ &< L(x) + L_M/2 - (B - L(z(t_0)))/2\end{aligned}$$

Since $B_{ij}(t_0)>B-L(z(t_0))$, B_{ij} has been non-empty since before $t_0-(B-L(z(t_0)))/2$, meaning that $q_j(t_0)>(B-L(z(t_0)))/2$. Consequently,

$$\begin{aligned}\sigma_{ij}(t_0) &> (B - L(z(t_0))) - (L(x) + L_M/2) \\ &\ge L_M/2 - L(z(t_0)) \ge -V_{ij}(t_0)\end{aligned}$$

and by Lemma 3, this remains true for the remainder of the active period. ■

Theorem 2. Any PGV scheduler with S=2 and $B\ge 2L_M$ is T-work-conserving for $T\ge 2L_M$.

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 $f(x)+2L_M<t$. By Lemma 4, $slack_{ij}(t)>-V_{ij}(t)$. Since, $q_j(t)=0$, this implies that $p_{ij}(t)< V_{ij}(t)$, which is not possible. ■

Lemmas 5 and 6 are used to prove Theorem 3, the emulation result for PGV-RP.

Lemma 5. Consider an active period of V_{ij} that spans the time interval $[t_1,t_2]$ where $t_1\ge$the time of the first scheduling event of the period. For a PGV-RP scheduler with speedup $S\ge 2$ and $B\ge 2L_M$, $\sigma_{ij}(t_2)\ge\sigma_{ij}(t_1)$.

proof. Assume that t_1 and t_2 are the times of two consecutive events at input i. Let z be the first packet in V_{ij} at t_1. If B_{ij} does not have room for z at time t, then $B_{ij}(t_1)>L_M$. This means that B_{ij} became non-empty before $t_1-L_M/2$. This implies that between the time B_{ij} become non-empty and t_1, there has been a scheduling event at output j. So, by t_1, output j is receiving bits that precede those in V_{ij}, leading to an increase in q_{ij}. Since $B_{ij}(t_1)>L_M$, q_{ij} will continue to increase until at least the next scheduling event at input i.

On the other hand, if B_{ij} does have room for z at t_1, then either z must be selected for transmission to the crossbar at t_1 or some packet that precedes z must be selected. In either case, this leads to a decrease in $\tilde{p}_{ij}$. Consequently, whether B_{ij} has room for z or not, the transfer of bits to or from the crossbar is sufficient to offset the effects of new arrivals at input i and departures from output j. This ensures that there is no decrease in σ_{ij}. ■

Lemma 6. Let x be the first packet of an active period for V_{ij} and let $t\ge f(x)+2L_M$ belong to the active period in which x arrives. For any PGV-RP scheduler with speedup S=2 and $B\ge 5L_M$, $slack_{ij}(t)\ge\sigma_{ij}(t)>L_M/2$.

proof. Let $\tau< f(x)+L_M/2$ be the time of the first scheduling event at input i in the active period. Suppose first, that $B_{ij}(\tau)>4L_M$. Consequently, B_{ij} became non-empty before $f(x)-2L_M$. This implies that q_{ij} has been increasing at rate 1 since before $s(x)-L_M/2$ and since $\tilde{p}_{ij}(f(x))=L(x)$ and $\tilde{p}_{ij}$ can increase at a rate no greater than 1, it follows that $\sigma_{ij}(\tau)>L_M/2$. By Lemma 5, this remains true for the remainder of the active period.

Now, suppose that $B_{ij}(\tau)\le 4L_M$. If there is no event in the interval $[\tau,f(x)+2L_M]$ when B_{ij} contains more than $B-L_M$ bits, then at every event in this interval, the scheduling algorithm will select either V_{ij} or a VOQ that precedes V_{ij}. Consequently,

$$\begin{aligned}\tilde{p}_{ij}(f(x)+2L_M) &\le L(x) + (\tau - f(x)) \\ &\quad - ((f(x)+2L_M) - \tau) \\ &\le L(x) + 2(\tau - f(x)) - 2L_M \\ &\le 0\end{aligned}$$

This implies that V_{ij} is empty by $f(x)+2L_M$ contradicting the hypothesis of the lemma. Hence, there must be some event in $[\tau,f(x)+2L_M]$ when B_{ij} contains more than $B-L_M$ bits. Let t_0 be the time of the first such event. So, $B_{ij}(t_0)>B-L_M$. and $t_0\le f(x)+2L_M$. Note also that $t_0>\tau+L(x)/2$ since at least one packet must have been sent from V_{ij} in order to make $B_{ij}(t_0)>B-L_M$. This implies that

$$\begin{aligned}\tilde{p}_{ij}(t_0) &\le L(x) + (\tau - f(x)) - (t_0 - \tau) \\ &< L(x) + L_M/2 - L(x)/2 \\ &\le L_M\end{aligned}$$

Since $B_{ij}(t_0)>B-L_M$, B_{ij} has been non-empty since before $t_0-(B-L_M)/2$, meaning that q_{ij} has been growing since before $t_0-(B-2L_M)/2$, so $q_{ij}(t_0)>(B-2L_M)/2\ge 3L_M/2$. Consequently, $\sigma_{ij}(t_0)>L_M/2$ and by Lemma 5, this remains true for the remainder of the active period. ■

Lemmas 7 and 8 are used to prove Theorem 4, the work-conservation result for PLF.

Lemma 7. Let t_1 and t_2 be the times of consecutive scheduling events in the same non-empty interval at input i and let M be the set of VOQs that are mature at t_1. For any PLF scheduler with speedup $S\ge 2$ and $B\ge 2L_M$, if all V_{ij} in M satisfy $\sigma_{ij}(t_1)\ge L_M/2$, then all V_{ij} in M that are still mature at t_2 satisfy $\sigma_{ij}(t_2)\ge L_M/2$.

proof. Let V_{ij} be a VOQ in M that is still mature at t_2 and let $\Delta=t_2-t_1$. If $B_{ij}(t_1)>B-L_M$, then output j receives 2Δ bits during the interval $[t_1,t_2]$. However, if $B_{ij}(t_1)\le B-L_M$, then 2Δ bits that precede V_{ij} at t_1, leave input i during $[t_1,t_2]$. This implies that if no VOQ that is older than V_{ij} passes V_{ij} during $[t_1,t_2]$, then $\sigma_{ij}(t_2)\ge\sigma_{ij}(t_1)\ge L_M/2$. Note that we are not concerned with VOQs younger than V_{ij} passing it, since the relative ordering of these VOQs

with V_{ij} does not affect $\tilde{p}_{ij}$ and hence does not affect σ_{ij}. In fact, $\sigma_{ij}(t_2) \geq \sigma_{ij}(t_1)$ even if there are some older VOQs that pass V_{ij}, so long as at t_1, these VOQs no longer contain bits that arrived before $s_{ij}(t_1)$.

Assume then that $r>0$ bits belonging to older VOQs that arrived before $s_{ij}(t_1)$ do pass V_{ij} during $[t_1,t_2]$ and let V_{ih} be the VOQ in the set of older VOQs that pass V_{ij} that comes latest in the VOQ ordering at t_1. Let k be the number of bits present at input i at t_1 that precede V_{ih} but not V_{ij}, and that arrived before $s_{ij}(t_1)$. Note that $k \leq \tilde{p}_{ih}(t_1) - \tilde{p}_{ij}(t_1)$ and that $r \leq k$. Since V_{ih} passes V_{ij}, output h must receive fewer bits than output j does during $[t_1,t_2]$, and since output j can receive no more than 2Δ bits during $[t_1,t_2]$, output h receives fewer than 2Δ. This implies that $B_{ih}(t_1) < 2\Delta \leq B - L_M$. Consequently, V_{ih} is eligible for selection at t_1, which implies that some packet z with 2Δ bits, that precedes V_{ih} at t_1 left input i during $[t_1,t_2]$.

We consider three cases. First, if z arrived after $s_{ij}(t_1)$ then, the departure of z reduces by 2Δ, the number of bits that are present at input i that arrived after $s_{ij}(t_1)$. Consequently,

$$\tilde{p}_{ij}(t_2) \leq \tilde{p}_{ij}(t_1) + r + \Delta - 2\Delta \leq \tilde{p}_{ij}(t_1) + k - \Delta$$

Similarly, if z arrived before $s_{ij}(t_1)$ and z precedes V_{ij} at t_1 then the departure of z reduces π_{ij} by 2Δ. Consequently,

$$\tilde{p}_{ij}(t_2) \leq \tilde{p}_{ij}(t_1) + r + \Delta - 2\Delta \leq \tilde{p}_{ij}(t_1) + k - \Delta$$

Finally, if z arrived before $s_{ij}(t_1)$ and z does not precede V_{ij} at t_1 then $r + 2\Delta \leq k$ and

$$\tilde{p}_{ij}(t_2) \leq \tilde{p}_{ij}(t_1) + r + \Delta \leq \tilde{p}_{ij}(t_1) + k - \Delta$$

So, in all three cases $\tilde{p}_{ij}(t_2) \leq \tilde{p}_{ij}(t_1) + k - \Delta$.

Since V_{ih} passes V_{ij}, $q_j(t_2) > q_h(t_2) \geq q_h(t_1) - \Delta$ and so,

$$\begin{aligned}\sigma_{ij}(t_2) &\geq (q_h(t_1) - \Delta) - (\tilde{p}_{ij}(t_1) + k - \Delta) \\ &\geq q_h(t_1) - \tilde{p}_{ih}(t_1) \geq L_M / 2\end{aligned}$$

■

Our next lemma extends Lemma 7.

Lemma 8. Let t_1 and t_2 be the times of consecutive scheduling events in the same non-empty interval at input i and let M be the set of VOQs that are mature at t_1. For any PLF scheduler with speedup $S \geq 2$ and $B \geq 16L_M/3$, if all V_{ij} in M satisfy $\sigma_{ij}(t_1) \geq L_M/2$, then all V_{ij} that are mature at t_2 satisfy $\sigma_{ij}(t_2) \geq L_M/2$.

proof. Since Lemma 7 covers the case of VOQs that were mature at t_1, we only concern ourselves with those VOQs that become mature during $[t_1,t_2]$. Let V_{ij} be such a VOQ, let x be the first packet that entered V_{ij} in the current active period for V_{ij} and let τ be the time of the first event in the current active period for V_{ij}. Note that because V_{ij} became mature during $[t_1,t_2]$, $t_2 - L_M/2 \leq t_1 < f(x) + T \leq t_2$. We divide the analysis into three cases.

Case 1. $\pi_{ij}(t_2)=0$ and $B_{ij}(\tau) > B - L_M$. Since $B_{ij}(\tau) > B - L_M$, output j has been receiving bits from the crossbar at rate 2, since before $f(x) - (B - L_M)/2 \leq s(x) - 3L_M/2$ and will continue to do so until at least $\tau + (B - L_M)/2$. If $t_2 \leq \tau + (B - L_M)/2$, $q_j(t_2) \geq t_2 - (s(x) - 3L_M/2)$ and since $\pi_{ij}(t_2)=0$, $\tilde{p}_{ij}(t_2) \leq t_2 - s(x)$. Hence $\sigma_{ij}(t_2) \geq L_M/2$. Assume then that $t_2 > \tau + (B - L_M)/2$. In this case,

$$\begin{aligned}q_j(t_2) &\geq \left[(\tau + (B - L_M)/2) - (f(x) - (B - L_M)/2)\right] \\ &\quad - \left[t_2 - (\tau + (B - L_M)/2)\right] \\ &= 3(B - L_M)/2 + 2\tau - t_2 - f(x)\end{aligned}$$

and

$$\begin{aligned}\sigma_j(t_2) &\geq \left(3(B - L_M)/2 + 2\tau - t_2 - f(x)\right) - (t_2 - s(x)) \\ &\geq 3(B - L_M)/2 + 2\tau - 2t_2 - L(x) \\ &\geq 3(B - L_M)/2 + 2f(x) \\ &\quad - 2(f(x) + T + L_M/2) - L(x) \\ &\geq 3(B - L_M)/2 - 6L_M \geq L_M/2\end{aligned}$$

This completes Case 1.

Case 2. $\pi_{ij}(t_2)=0$ and $B_{ij}(\tau) \leq B - L_M$. In this case, V_{ij} is eligible to be selected at τ, so V_{ij} or some other VOQ preceding V_{ij} must be selected at τ. Suppose there is no event in $[\tau, t_2]$ when B_{ij} has more than $B - L_M$ bits. Then,

$$\begin{aligned}\tilde{p}_{ij}(t_2) &\leq L(x) + (\tau - f(x)) - (t_2 - \tau) \\ &\leq L(x) + L_M/2 - (T - L_M/2) \\ &\leq 2L_M - T = 0\end{aligned}$$

This contradicts the fact that V_{ij} is active at t_2, so there must be some event in $[\tau,t_2]$ when B_{ij} has more than $B - L_M$ bits. Let t be the time of the first such event and note that $t \geq \tau + L(x)/2$, since at least one packet must enter B_{ij} to make $B_{ij}(t) > B - L_M$. Since $B_{ij}(t) > B - L_M$, $q_j(t) > (B - L_M)/2$. Also,

$$\begin{aligned}\tilde{p}_{ij}(t) &\leq L(x) + (\tau - f(x)) - (t - \tau) \\ &\leq L(x) + (\tau - f(x)) - L(x)/2 \\ &\leq L_M\end{aligned}$$

So, $\sigma_{ij}(t) \geq (B - L_M)/2 - L_M \geq L_M/2$. If $t + (B - L_M)/2 \geq t_2$, then q_j continues to grow at rate 1 until t_2. This is enough to compensate for any growth in $\tilde{p}_{ij}$. Hence, $\sigma_{ij}(t_2) \geq L_M/2$. Assume then that $t + (B - L_M)/2 < t_2$. In this case, q_j contin-

ues to grow at rate 1 until $t+(B-L_M)/2$ giving $q_j(t+(B-L_M)/2)>B-L_M$. Thus,

$$\begin{aligned} q_j(t_2) &\geq (B-L_M) - \left(t_2 - (t+(B-L_M)/2)\right) \\ &\geq 3(B-L_M)/2 - (t_2-\tau) \end{aligned}$$

and since $\tilde{p}_{ij}(t_2)\leq t_2-s(x)$,

$$\begin{aligned} \sigma_{ij}(t_2) &\geq 3(B-L_M/2) - (t_2-\tau) - (t_2-s(x)) \\ &\geq 3(B-L_M/2) - 2t_2 + 2f(x) - L(x) \\ &\geq 3(B-L_M/2) - 2(f(x)+T+L_M/2) \\ &\qquad + 2f(x) - L_M \\ &\geq 3(B-L_M/2) - 6L_M \geq L_M/2 \end{aligned}$$

This completes Case 2.

Case 3. $\pi_{ij}(t_2)>0$. This implies that there is some VOQ that precedes V_{ij} at t_2 and is older than V_{ij}. Suppose that V_{ij} is the first such VOQ to become mature in $[t_1,t_2]$. Let V_{ih} be an older VOQ that precedes V_{ij} at t_2 and assume further, that among all such VOQs, it comes latest in the VOQ ordering at t_2. Note that with these assumptions, either V_{ih} was mature at t_1, or there is no VOQ that is both older than V_{ih} and that precedes V_{ih}. In either case $\sigma_{ih}(t_2)\geq L_M/2$, Let $k=p_{ij}(t_2)-p_{ih}(t_2)$. Note that all bits that precede V_{ij} at t_2, but not V_{ih} must have arrived since $s(x)$, where x is first packet of the current active period for V_{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 V_{ih} became active. Consequently, $p_{ih}(t_2)\leq\tilde{p}_{ih}(t_2)-k$ and so,

$$\begin{aligned} slack_{ih}(t_2) &= q_h(t_2) - p_{ih}(t_2) \\ &\geq q_h(t_2) - \tilde{p}_{ih}(t_2) + k \\ &\geq L_M/2 + k \end{aligned}$$

Consequently, $q_j(t_2)-p_{ij}(t_2) \geq q_h(t_2)-p_{ih}(t_2)-k \geq L_M/2$. Note that this inequality holds even when $p_{ij}(t_2)=\tilde{p}_{ij}(t_2)$ implying that, $\sigma_{ij}(t_2)\geq L_M/2$ also. We can establish the result for all VOQs that become active during $[t_1,t_2]$ by induction. The inductive step follows by the same argument, using the fact that all older VOQs to become active in $[t_1,t_2]$ have $\sigma(t_2)\geq L_M/2$. ■