On Achieving Throughput in an Input-Queued Switch

Saad Mneimneh and Kai-Yeung Siu

***Abstract*—We establish some lower bounds on the speedup required to achieve throughput for some classes of switching algorithms in a input-queued switch with virtual output queues (VOQs). We use a weak notion of throughput, which will only strengthen the results, since an algorithm that cannot achieve weak throughput cannot achieve stronger notions of throughput. We focus on priority switching algorithms, i.e., algorithms that assign priorities to VOQs and forward packets of high priority first. We show a lower bound on the speedup for two fairly general classes of priority switching algorithms: input priority switching algorithms and output priority switching algorithms. An input priority scheme prioritizes the VOQs based on the state of the input queues, while an output priority scheme prioritizes the VOQs based on their output ports. We first show that, for output priority switching algorithms, a speedup $S \geq 2$ is required to achieve weak throughput. From this, we deduce that both maximal and maximum size matching switching algorithms do not imply weak throughput unless $S \geq 2$. The bound of $S \geq 2$ is tight in all cases above, based on a result in Dai *et al.* Finally, we show that a speedup $S \geq 3/2$ is required for the class of input priority switching algorithms to achieve weak throughput.**

***Index Terms*—Lower bounds, priority switching algorithms, speedup, throughput.**

I. Introduction

TRADITIONAL output queued or shared memory architectures are becoming increasingly inadequate to meet high-bandwidth requirements, because having to account for multiple arrivals to the same output requires their switch memories to operate at N times the line speed, where N is the number of input ports. Although input-queued switches provide an attractive alternative because their memory and switch fabrics may operate at only the line speed, they present a challenge for providing quality-of-service (QoS) guarantees comparable to those provided by output-queued switches, and they require a sophisticated scheduler or arbiter, making it a critical component of the switch.

For instance, traditional switching algorithms that achieve 100% throughput in an input-queued switch do not provide strict delay guarantees and are based on computing a maximum weighted matching that requires a running time of $O(N^3)$ [11], [13], or $O(N^{2.5})$ [14], making them impractical to implement on high-speed switches. Some recent work [4] has, therefore, focused on asking whether an input-queued switch can be made to emulate an output-queued switch, and has demonstrated that this can be achieved by a combination of a speedup in the fabric (of $2 - 1/N$) and a special switching algorithm based on computing a stable-marriage matching. Such emulation involves substantial bookkeeping and communication overhead at the scheduler, however, and despite its theoretical significance, is not practical at high speeds.

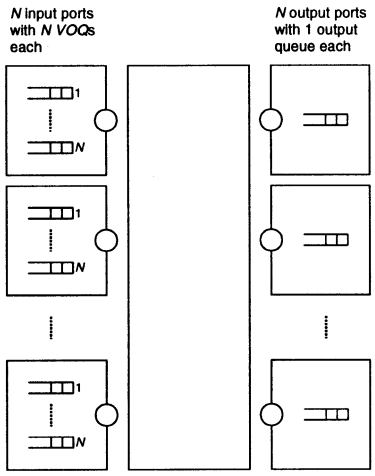

Fig. 1. Input–output-queued switch.

Most practical switching algorithms for input-queued switches (see, for instance, [3] and [8]) require a speedup of between 2 and 4 to achieve adequate QoS guarantees. These algorithms are priority-based switching algorithms and operate by assigning priorities to the input queues (the virtual output queues, VOQs) and forwarding packets of high priority first. We will prove that the speedup requirement for these algorithms is inherent, in the sense that they cannot achieve throughput without a speedup in the switch. As a result, the switch fabric and the memory need to operate faster than the line speed by the speedup factor. This also implies that the input-queued switch will require queues at the output as well, since now more than one packet can be forwarded to an output in a single time unit. Fig. 1 depicts the traditional architecture of an input–output-queued switch with VOQs. Our model of a switch will be essentially the same general model of Fig. 1.

We will denote by VOQ_{ij} the j^{th} VOQ at input i, which will hold packets originating at input i and destined to output j. With no speedup in the switch, at most one packet can be forwarded to a particular output in a single time unit. This packet can be consumed by the output port by the end of the time unit and, hence, there will be no need to store packets at the output. Therefore, with no speedup, no output queues are needed in the architecture of Fig. 1. Similarly, at most one packet will be forwarded from an input port in a single time unit when the switch has no speedup. Therefore, the set of packets that are forwarded in a particular time unit satisfies the condition that no two packets

Manuscript received December 5, 2000; revised April 8, 2002 and November 22, 2002; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor N. McKeown. This work was supported by the Networking Research Program of the National Science Foundation under Award 9973015.

S. Mneimneh is with the Southern Methodist University, Dallas, TX 75275 USA (e-mail: saad@engr.smu.edu).

K.-Y. Siu is with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: ssiu@razafoundries.com).

Digital Object Identifier 10.1109/TNET.2003.818180

will share an input or an output. This notion is formally abstracted in the literature as a matching.

The operation of a switch with no speedup is, therefore, modeled as successive computations of matchings in a bipartite graph, one matching in each time unit. The bipartite graph is obtained as follows: the input ports form a set of nodes, the output ports form the other set of nodes, and the nonempty VOQs form the edges between the two sets of nodes. Thus, a nonempty VOQ_{ij} is modeled as an edge between node i (input) and node j (output). The matching is a set of disjoint edges, i.e., edges that do not share nodes. Therefore, the matching matches input ports to output ports, and if input i is matched to output j in a particular time unit, VOQ_{ij} is served in that time unit.

The general model of a switch with speedup can be described hereafter as follows: the switch operates in matching phases, computing a matching in every phase. We will assume that the switch computes a maximal matching in every phase, i.e., a matching where no more inputs and outputs can be matched. Assuming continuous time, a switch with speedup S takes $1/S$ time units to complete a matching phase before starting the next phase. Therefore, if $S > 1$, output queues are used at the output ports because packets will be forwarded to the output at a speed higher than the line speed.

II. Traffic Models

We will mention three traffic models used in the literature. A traffic model describes the arrival of packets to the switch as a function of time. A traffic model can be probabilistic or deterministic, as will be seen shortly. Before we proceed to the different traffic models, we need to define a quantity that tracks the number of packets arriving to the switch. Let $A_{ij}(t)$ be the number of packets that arrive to the switch by time t at input i and are destined to output j.

A. Strong Law of Large Numbers (SLLN) Traffic

This traffic conforms to a probabilistic model that obeys the Strong Law of Large Numbers, hence, its name SLLN. In simpler terms, this means that it is possible to define a rate λ_{ij} for the flow of packets from input i to output j.

SLLN:

- $\lim_{t\to\infty}(A_{ij}(t))/(t) = \lambda_{ij}$ with probability 1;
- $\sum_k \lambda_{ik} \leq \alpha$;
- $\sum_k \lambda_{kj} \leq \alpha$;
- $\alpha \leq 1$.

The parameter α is called the *loading* of the switch and will be present in the two traffic models below, which are based on the (σ, ρ) model, where σ represents the burstiness and ρ represents an upper bound on the long-term average rate (if it exists).

B. Weak Constant Burst Traffic

In some sense, the weak constant burst traffic is a stronger model than SLLN because it is deterministic. However, it does not necessarily define a rate for the flow of packets from input i to output j. Alternatively, it provides a bound B on the burst of packets at input i and output j. This bound is a constant independent of time. It also defines an upper bound (α in this case) on the rate of the aggregate flow at each input and output if it exists. Nevertheless, the model does not directly constrain the flow of packets from input i to output j, hence, the use of the term *weak* in the burst characterization of this traffic model.

Weak Constant Burst:

- $\forall t_1 \leq t_2, \sum_k A_{ik}(t_2) - A_{ik}(t_1) \leq \alpha(t_2 - t_1) + B$;
- $\forall t_1 \leq t_2, \sum_k A_{kj}(t_2) - A_{kj}(t_1) \leq \alpha(t_2 - t_1) + B$;
- $\alpha \leq 1$.

It can be seen that for any rate assignment, some flow from input i to output j can still have a time-dependent burst without violating the above conditions. For instance, assume $\alpha = 1$ and all the flows are sending packets at their assigned rates. If all flows stop sending packets except for one flow that starts sending packets at a rate of 1, none of the conditions above is violated, but that flow can exhibit a time-dependent burst until it exceeds any burst bound, then comes back to normal, where all flows start sending at their assigned rates again.

C. Strong Constant Burst Traffic

The following model implies the previous model and is, therefore, stronger (more constrained). It imposes a time-independent bound on the burst of the individual flows, hence, a time-independent bound on the burst at each input and output port. The strong constant burst model does not necessarily define a rate for an individual flow of packets from input i to output j. Unlike the weak constant burst model, however, it defines an upper bound λ_{ij} on that rate if it exists.

Strong Constant Burst:

- $\forall t_1 \leq t_2, A_{ij}(t_2) - A_{ij}(t_1) \leq \lambda_{ij}(t_2 - t_1) + B$;
- $\sum_k \lambda_{ik} \leq \alpha$;
- $\sum_k \lambda_{kj} \leq \alpha$;
- $\alpha \leq 1$.

Note that both the weak constant burst and the strong constant burst models do not necessarily imply the SLLN model because $\lim_{t\to\infty}(A_{ij}(t))/(t)$, which is the rate, might not exist. However, if that limit exists, then the strong constant burst model satisfies the SLLN model.

III. Throughput

Throughput basically means that, as time evolves, the switching algorithm will be able to forward all the packets that arrive to the switch. There are many definitions of throughput, and some definitions depend on the adopted traffic model. One possible definition of throughput under a probabilistic traffic model is for the expected length of each VOQ to be bounded. Therefore, if $X_{ij}(t)$ denotes the length of VOQ_{ij} at time t, we require that $E[X_{ij}(t)] \leq M < \infty$ [11], [13], [14]. One can show that this implies that for any $\epsilon > 0$, there exists a time t_0 such that $Pr[(X_{ij}(t))/(t) \geq \epsilon] \leq \epsilon$ for every $t \geq t_0$. Therefore, $(X_{ij}(t))/(t)$ converges to 0 in probability. If $\lim_{t\to\infty}(A_{ij}(t))/(t) = \lambda_{ij}$ in probability, this definition of throughput implies that $\lim_{t\to\infty}(D_{ij}(t))/(t) = \lambda_{ij}$ in probability, where $D_{ij}(t) = A_{ij}(t) - X_{ij}(t)$. Other definitions of throughput require that under an SLLN traffic, $\lim_{t\to\infty}(D_{ij}(t))/(t) = \lambda_{ij}$ with probability 1 [5]. It is possible to show that if $E[X_{ij}^2(t)]$ is bounded, then

$\lim_{t\to\infty}(X_{ij}(t))/(t) = 0$ with probability 1, which in turn implies that $\lim_{t\to\infty}(D_{ij}(t))/(t) = \lambda_{ij}$ with probability 1 if $\lim_{t\to\infty}(A_{ij}(t))/(t) = \lambda_{ij}$ with probability 1.

In this paper, we will use a weak notion of throughput, defined below.

Definition 1 (Weak Throughput): Let $X_{ij}(t)$ be the length of VOQ_{ij} at time t. Then $\lim_{t\to\infty}(X_{ij}(t))/(t) = 0$.

The above definition can be also expressed as follows. For every $\epsilon > 0$, there exists a time t_0 such that $(X_{ij}(t))/(t) \leq \epsilon$ for any time $t \geq t_0$.

Note that in the above definition, the throughput does not rely on the fact that $\lim_{t\to\infty}(A_{ij}(t))/(t)$ exists. Note also that the definition does not impose any strict bound on the size of the VOQs (in fact, the size can grow to infinity).

IV. Some Previous Work

Here, we review some of the practical switching algorithms found in the literature and their speedup requirements.

Charny *et al.* proved in [3] that any maximal matching policy (i.e., any switching algorithm that computes a maximal matching in every matching phase) achieves a bounded delay on every packet in an input-queued switch with a speedup $S > 4$ under a weak constant burst traffic. We will prove that the simple policy of computing just any maximal matching does not imply weak throughput for a speedup $S < 2$. In fact, we prove that a maximum size matching policy does not imply weak throughput for $S < 2$.

Since switches with speedup are not desired due to their manufacturing cost and impracticality, it is legitimate to look at what loading α a switch with no speedup (i.e., $S = 1$) can tolerate. A work in [7] addresses this issue and provides a switching algorithm (called the *Central Queue* algorithm) that computes a $1/2$-approximation of the maximum weighted matching, where the length of VOQ_{ij} is used as the weight for edge (i, j). This work is a slight generalization of the result described in [16] applied to the very special setting of a switch. The $1/2$-approximation algorithm used in [7] is a priority switching algorithm where VOQs with larger length are considered first as candidates for the matching (the description of this algorithm is actually different in [7], but for our purposes it is to convenient to describe it in this way). The *Central Queue* algorithm achieves bounded VOQ length when $\alpha < 1/2$ under a strong constant burst traffic. The results obtained in this paper will prove that it cannot achieve weak throughput unless $S \geq 3/2\alpha$ and, hence, if $S = 1$ (no speedup), it cannot achieve weak throughput for $\alpha > 2/3$.

In [3], the authors provide an algorithm called *Oldest Cell First* that guarantees a bounded delay on every packet with a speedup $S > 2$ under a weak constant burst traffic. The same algorithm can be proved to achieve bounded VOQ length with a speedup of 2 under a strong constant burst traffic. This switching algorithm is a priority switching algorithm that assigns higher priority to VOQs with older head-of-line (HOL) packets. Similarly, the results obtained here will prove that this algorithm cannot achieve weak throughput unless $S \geq 3/2$. Note that another switching algorithm called *Oldest Cell First* appears in [13] and is based on computing a maximum weighted matching. This is not to be confused with the algorithm described above.

In [8], Krishna *et al.* provide an algorithm called *Lowest Occupancy Output Queue First* (LOOFA) that guarantees a bounded delay on every packet with a speedup of 2 and a strong constant burst traffic, and uses a more sophisticated priority scheme. This algorithm has also a work conservation property that we will not address here. The same lower bound of $S \geq 3/2$ applies for this algorithm as well in the sense that LOOFA does not imply weak throughput unless $S \geq 3/2$.

Before we prove our lower bound results for the different classes of priority switching algorithms, we need to define the traffic model that we will assume for the rest of the paper. We also need to provide a formal definition of priority to obtain the framework for a priority switching algorithm.

V. Traffic Assumptions

We define a restricted model of traffic under which we will prove our lower bound results on S. Note that more restricted traffic yields stronger results.

Definition 2: An *α-shaped* traffic is a traffic that satisfies the following:

- $\forall t_1 \leq t_2$, $A_{ij}(t_2) - A_{ij}(t_1) = \lambda_{ij}(t_2 - t_1) \pm O(1)$;
- $\forall t_1 \leq t_2$, $\sum_k A_{ik}(t_2) - A_{ik}(t_1) = \sum_k \lambda_{ik}(t_2 - t_1) \pm O(1)$;
- $\forall t_1 \leq t_2$, $\sum_k A_{kj}(t_2) - A_{kj}(t_1) = \sum_k \lambda_{kj}(t_2 - t_1) \pm O(1)$;
- $\sum_k \lambda_{ik} \leq \alpha$;
- $\sum_k \lambda_{kj} \leq \alpha$;
- $\alpha \leq 1$.

The above conditions state that the rate of the flow from input i to output j exists and is equal to λ_{ij}. Moreover, the burst of the flow from input i to output j (first condition), and the aggregate flow at any input and any output (second and third conditions), is $B = O(1)$ independent of time as well as the size of the switch N. Note that the first condition does not necessarily imply the second and third conditions, namely, because the individual flow bursts of $O(1)$ may result in an $O(N)$ burst at some input or output port. The α-shaped traffic satisfies the SLLN model as well as the strong constant burst model. This is the traffic model under which we will prove the various lower bound results. As a consequence, the results will hold for all traffic models described in Section II, namely, the SLLN traffic, the weak constant burst traffic, and the strong constant burst traffic.

VI. Priority Scheme

In this section, we formally define a priority scheme which will provide the framework for a priority switching algorithm. A priority scheme imposes on the VOQs an order by which they are considered for the matching.

Definition 3: A priority scheme π defines for every matching phase m a partial order relation π_m on the VOQs.

We will use the notation $\mathrm{VOQ}_{ij} \prec \mathrm{VOQ}_{kl}$ to denote that VOQ_{ij} has higher priority than VOQ_{kl} during matching

phase m. We will also use the notation $\mathrm{VOQ}_{ij} \not\prec_{\pi_m} \mathrm{VOQ}_{kl}$ to denote that VOQ_{ij} does not have higher priority than VOQ_{kl} during matching phase m.

Note that since π_m is a partial order relation, two VOQs might be unordered by π_m (e.g., for VOQ_{ij} and VOQ_{kl}, $\mathrm{VOQ}_{ij} \not\prec_{\pi_m} \mathrm{VOQ}_{kl}$ and $\mathrm{VOQ}_{kl} \not\prec \mathrm{VOQ}_{ij}$). In order for this to cleanly reflect the notion of equal priority, we define a well-behaved priority scheme, as follows.

Definition 4: A well-behaved priority scheme π is a priority scheme such that for every matching phase m, if VOQ_{ij} and VOQ_{kl} are unordered by π_m, and VOQ_{kl} and VOQ_{mn} are unordered by π_m, then VOQ_{ij} and VOQ_{mn} are unordered by π_m.

The above condition on the priority scheme reflects the notion of equal priority. Hence, if during a particular matching phase, VOQ_{ij} and VOQ_{kl} have equal priority, and VOQ_{kl} and VOQ_{mn} have equal priority, then VOQ_{ij} and VOQ_{mn} will have equal priority. This condition defines an equivalence relation on the VOQs which will help us later to explicitly extend the partial order relation to a total order relation by which all VOQs are ordered.

In practice, a priority switching algorithm breaks ties among the VOQs with equal priorities. We will assume that ties are broken using the indices of the ports and, hence, we assume the existence of a fixed total order relation on the (i, j) pairs used for breaking ties. Adopting the assumption that breaking a tie among two VOQs involves only the two VOQs in question and no other information, this is a fairly general deterministic way of breaking ties; other policies that are more sophisticated can usually be incorporated into the priority scheme itself. The following definition captures the idea.

Definition 5: Let π be a well-behaved priority scheme and ϕ be a total order relation on the (i, j) pairs. We define the ϕ extension of π to be the priority scheme π^ϕ as follows. For any matching phase m, if $\mathrm{VOQ}_{ij} \prec_{\pi_m} \mathrm{VOQ}_{kl}$, then $\mathrm{VOQ}_{ij} \prec_{\pi_m^\phi} \mathrm{VOQ}_{kl}$. For any matching phase m, if VOQ_{ij} and VOQ_{kl} are unordered by π_m, and $(i,j) \prec_\phi (k,l)$, then $\mathrm{VOQ}_{ij} \prec_{\pi_m^\phi} \mathrm{VOQ}_{kl}$.

Since π is well-behaved, π_m induces the equal priority equivalence relation on the VOQs in every matching phase m. This in turn implies that we can extend π as described above without violating the property of an order relation. We omit the proof of this fact. Therefore, if π is a well-behaved priority scheme, then π^ϕ is a priority scheme such that π_m^ϕ orders all VOQs for every matching phase m.

Note that our definition of a priority scheme is general enough to tolerate changing the definition of the partial order relation in every matching phase. Therefore, it is possible to prioritize the VOQs based on their lengths in one matching phase and based on the age of their HOL packets in another.

A priority switching algorithm computes its matchings based on a given priority scheme π by forwarding packets with higher priority first. Fig. 2 describes one possible implementation (greedy) of one matching phase.

For a given priority scheme π, we can generally define a matching that describes the outcome of a priority switching algorithm, as follows.

Definition 6: For a given priority scheme π, a matching computed in matching phase m is *π-stable* iff it satisfies the following condition: if a nonempty VOQ_{ij} is not served, then a nonempty VOQ_{ik} is served and $\mathrm{VOQ}_{ij} \not\prec_{\pi_m} \mathrm{VOQ}_{ik}$, or a nonempty VOQ_{kj} is served and $\mathrm{VOQ}_{ij} \not\prec_{\pi_m} \mathrm{VOQ}_{kj}$.

Priority Switching Algorithm

start with an empty matching $M = \emptyset$
prioritize all VOQs according to π_m
repeat the following until M is maximal
  choose a non-empty VOQ_{ij} with a highest priority
  if $M \cup (i,j)$ is a matching, then $M = M \cup (i,j)$
  discard VOQ_{ij}

Fig. 2. Matching phase m of a priority switching algorithm.

The notion of a π-stable matching is more general than the process depicted in Fig. 2. In other terms, a priority switching algorithm for the priority scheme π will always compute a π-stable matching. Although the most intuitive and straightforward way of computing a π-stable matching is the greedy approach described in Fig. 2, Definition 6 does not impose any restriction on how the π-stable matching is computed.

In the next section, we prove lower bound results on the speedup under an α-shaped traffic.

VII. Lower Bounds

We will start by stating, without proof, the following simple lemma.

Lemma 1: If an event E occurs every $\tau \neq 0$ time units, then the number of times $E_{[t_1,t_2]}$ the event occurs in the interval $[t_1, t_2]$, satisfies

$$\frac{t_2 - t_1}{\tau} - 1 < E_{[t_1,t_2]} \leq \frac{t_2 - t_1}{\tau} + 1.$$

We will later use this lemma to argue a lower bound on the number of packets arriving from a particular input i during an interval of time, and an upper bound on the number of matching phases during the same interval of time. Using these bounds, we will prove our different results by showing that the number of packets arriving to the switch at a particular input is more than the number of times that input is served (i.e., some VOQ at that input is served) by the matching phases. In order to obtain such scenario for a given algorithm, we make use of an adversary. The adversary will supply the switch (the algorithm) with an α-shaped traffic that will force the algorithm to fail in achieving weak throughput unless the speedup is high enough.

We will denote by a matching policy a switching algorithm which computes a matching that satisfies the policy in every matching phase. For instance, a π-stable matching policy denotes a priority switching algorithm for the priority scheme π. We will also use loosely the notion of *reducibility*. For instance, when we say that weak throughput is not reducible to some matching policy, we mean that using this matching policy does not necessarily imply weak throughput for any traffic and switch size. As an example, a matching policy could be merely any maximal matching with no other conditions on the matching. Therefore, if we say that weak throughput is not reducible to a maximal matching policy, we mean that an algorithm which computes a maximal matching in every matching phase does not

necessarily imply weak throughput for any traffic and switch size.

A. Output Priority Switching Algorithms

In this section, we establish a lower bound on the speedup for a class of priority switching algorithms that employ an output priority scheme defined next.

Definition 7: An output priority scheme π is a priority scheme that satisfies the following: for every matching phase m, there exists a partial order relation π'_m on the output ports such that $\mathrm{VOQ}_{ij} \prec_{\pi_m} \mathrm{VOQ}_{kl}$ iff $j \prec_{\pi'_m} l$.

Note that according to this definition, VOQ_{ij} and VOQ_{kj} are unordered by an output priority scheme (since $j \not\prec_{\pi'_m} j$ for any matching phase m), reflecting the fact that neither has priority over the other because they share the same output. An example of an output priority scheme is *lowest output occupancy* where a VOQ_{ij} has higher priority if there are less packets in output queue j. This scheme was used in LOOFA [8] which forward packets for the smallest length output queue first.

Next, we describe the first adversary that we will use.

The ϕ-adversary: Let ϕ be any total order relation on the (i, j) pairs. We will assume, without loss of generality, that (1,1) is the highest ranked according to ϕ. Similarly, after discarding (1, k) and (k, 1) for all $k = 1 \ldots N$, we assume that (2,2) has the highest rank according to ϕ among the remaining pairs. We continue until we obtain pairs $(3,3) \ldots (N-1, N-1)$ in the same way. The adversary produces an α-shaped traffic as shown in Fig. 3.

At input N, the flow of rate α is divided equally among the $N-1$ outputs in a round-robin fashion. The adversary produces a packet at input N every $1/\alpha$ time units. Similarly, the adversary produces a packet at input i, where $i = 1 \ldots N-1$, every $(N-1)/(N-2)\alpha$ time units. It can be shown that this traffic is α-shaped. For instance, using Lemma 1 and the fact that the adversary uses a round-robin order at input N to generate packets for the first $N-1$ outputs, we can show that during any time interval $[t_1, t_2]$, the number of packets from input N to any of the first $N-1$ outputs satisfies

$$\left\lfloor \frac{\alpha T - 1}{N-1} \right\rfloor \leq A_{Nj}(t_2) - A_{Nj}(t_1) \leq \left\lceil \frac{\alpha T + 1}{N-1} \right\rceil$$

where $T = t_2 - t_1$. This conforms with the first condition of an α-shaped traffic. The condition is also true for all other flows. We can show that the rest of the conditions are also satisfied. Note also that no overloading occurs since, at any port, the sum of the rates of all flows is at most

$$\frac{\alpha}{N-1} + \frac{N-2}{N-1}\alpha = \alpha.$$

Lemma 2: For any well-behaved output priority scheme π and any total order relation ϕ on the (i, j) pairs, a π^ϕ-stable matching policy, under the ϕ-adversary, cannot serve inputs 1 and N during the same matching phase.

Proof: By the property of an output priority scheme π, for any matching phase m, VOQ_{jj} and VOQ_{Nj} are unordered by π_m. Therefore, we have that $\mathrm{VOQ}_{jj} \prec_{\pi^\phi_m} \mathrm{VOQ}_{Nj}$ for every matching phase m because $(j, j) \prec_\phi (N, j)$ (see the definition of the ϕ-adversary). Hence, the π^ϕ-stable matching policy will choose the matching $\{(1,1), (2,2), \ldots, (N-1, N-1)\}$ whenever possible. Since the ϕ-adversary provides the same traffic for flows $(1,1), (2,2), \ldots$, and $(N-1, N-1)$, the matching policy will always be able to pick the corresponding edges together. In other words, it is not possible that VOQ_{ii} is nonempty and VOQ_{jj} is not for $i, j = 1 \ldots N-1$. As a result, inputs 1 and N cannot be served during the same matching phase. ■

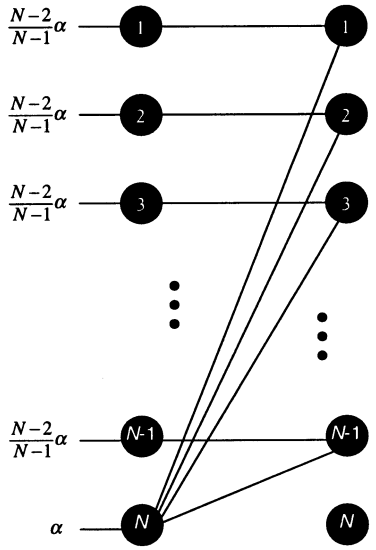

Fig. 3. The ϕ-adversary.

Theorem 1: For any well-behaved output priority scheme π and any total order relation ϕ on the (i, j) pairs, a π^ϕ-stable matching policy cannot achieve weak throughput under an α-shaped traffic unless $S \geq 2\alpha$.

Proof: Consider the ϕ-adversary. Pick a time t. By Lemma 1, we have at most $tS + 1$ matching phases by time t, each of which is forwarding at most one packet from inputs 1 and N by Lemma 2. By Lemma 1, the number of packets arriving to input 1 and N by time t is at least

$$\alpha t - 1 + \frac{N-2}{N-1}\alpha t - 1.$$

Therefore, at time t, the number of packets remaining at inputs 1 and N is at least

$$\left(\frac{2N-3}{N-1}\alpha - S\right) t - 3.$$

For $S < 2\alpha$, there exists a large enough N, say, N_0, such that $(2N_0 - 3)/(N_0 - 1)\alpha - S = \delta > 0$. If weak throughput is to be achieved, then for every $\epsilon > 0$, there must exist a large enough t, say, t_0, such that for every VOQ_{ij}, $(X_{ij}(t))/(t) \leq \epsilon$ for any $t \geq t_0$. Assume that weak throughput is achieved and let $\epsilon < \delta/N_0$ and t_0 be as defined above. Let $t \geq t_0$ be such that $\delta/N_0 - (3)/(N_0 t) > \epsilon$. Since at inputs 1 and N_0 we have at most $1 + (N_0 - 1) = N_0$ nonempty VOQs, there exists a VOQ_{ij} such that the number of packets remaining in VOQ_{ij} at time t is at least $(\delta t - 3)/(N_0)$. Therefore

$$\frac{X_{ij}(t)}{t} \geq \frac{\delta}{N_0} - \frac{3}{N_0 t} > \epsilon.$$

Since $t \geq t_0$, we have a contradiction. ■

We have proved that any switching algorithm based on an output priority scheme that breaks ties using the indices of the ports cannot achieve weak throughput under an α-shaped traffic unless $S \geq 2$. The implication of this result is that a speedup of at least 2 is required for an output priority switching algorithm to provide throughput with a full loading of the switch. Below, we prove a corollary.

Corollary 1: For any output priority scheme π, weak throughput is not reducible to a π-stable matching policy unless $S \geq 2$.

Proof: There exists a well-behaved output priority scheme π' such that $\text{VOQ}_{ij} \prec_{\pi_m} \text{VOQ}_{kl} \Rightarrow \text{VOQ}_{ij} \prec_{\pi'_m} \text{VOQ}_{kl}$ for every matching phase m.[1] Therefore, for any total order relation ϕ on the (i, j) pairs, $\text{VOQ}_{ij} \prec_{pi_m} \text{VOQ}_{kl} \Rightarrow \text{VOQ}_{ij} \prec_{\pi'^{\phi}_m} \text{VOQ}_{kl}$ for every matching phase m. Hence, a π'^{ϕ}-stable matching policy is a π-stable matching policy and the result is immediate from Theorem 1 using $\alpha = 1$. ■

The basic version of LOOFA [8] considers first the VOQs with lower output queue occupancy as candidates for the matching. As a consequence, it only guarantees that some π-stable matching policy will be used, where π is the *lowest output occupancy* priority scheme. Therefore, we proved that this switching algorithm does not imply weak throughput for $S < 2$. LOOFA assumes that at most one packet arrives to any input per time unit. The ϕ-adversary satisfies this condition (see Fig. 3).

Theorem 1 also implies that a greedy switching algorithm, where the VOQs are always considered for the matching in a fixed particular order ϕ, cannot achieve weak throughput unless $S \geq 2\alpha$. To see this, simply set the output priority scheme π such that π_m is the empty relation for every matching phase m. As a result, $\pi^{\phi}_m = \phi$ will be the order by which the greedy algorithm serves the VOQs in every matching phase m, resulting in a π^{ϕ}-stable matching. By Theorem 1, $S \geq 2\alpha$ is required to achieve weak throughput. An example of such a greedy switching algorithm is the Wave Front Arbiter (WFA) [15] when the priorities do not rotate.

B. Maximum Size Matching

Consider the switching algorithm that computes a maximum size matching in every matching phase. McKeown *et al.* proved in [11] that such an algorithm, with probability 1, will not achieve weak throughput unless $S \geq 1.037\alpha$, when arrivals to the switch are i.i.d. Bernoulli arrivals and a random maximum size matching is computed. We will consider the lower bound on S when this switching algorithm is deterministic. Consider the ϕ-adversary described earlier. Note that for any well-behaved output priority scheme π and any total order relation ϕ on the (i, j) pairs, a π^{ϕ}-stable matching policy is a maximum size matching policy under the ϕ-adversary. To see this, note that the maximum possible size for a matching is $N-1$ when the first $N-1$ outputs are matched. Note also that, whenever possible, the π^{ϕ}-stable matching policy will choose the matching $\{(1,1),(2,2),\ldots,(N-1,N-1)\}$ where VOQ_{ii} for $i = 1 \ldots N-1$ are either nonempty together or none of them is. As a consequence, we have the following result.

Corollary 2: Weak throughput is not reducible to a maximum size matching policy unless $S \geq 2$.

Proof: Immediate from Theorem 1 using $\alpha = 1$ since, as argued above, for any well-behaved output priority scheme π and any total order relation ϕ on the (i, j) pairs, under the ϕ-adversary, a π^{ϕ}-stable matching policy is a maximum size matching policy. ■

C. Maximal Matching

Since a maximum size matching is also a maximal matching, we have the following result.

Corollary 3: Weak throughput is not reducible to a maximal matching policy unless $S \geq 2$.

Proof: Immediate from Corollary 2 since a maximum size matching is a maximal matching. ■

In a recent paper, Dai *et al.* [5] proved that with $S \geq 2$, any maximal matching policy guarantees weak throughput with probability 1 under an SLLN traffic. We have just proved (Corollary 3) that this is not true when $S < 2$. Therefore, since both a π-stable matching and a maximum size matching are maximal matchings, the lower bound results obtained so far are tight.

Charny *et al.* proved in [3] that a delay guarantee is reducible to a maximal matching policy if $S > 4$ under any weak constant burst traffic. It can be shown that bounded VOQ length is reducible to a maximal matching policy if $S = 4$ under a strong constant burst traffic. The question of achieving bounded VOQ length with any maximal matching policy under constant burst traffic models for $S \in [2, 4]$ remains to be answered.

D. Input Priority Switching Algorithms

In this section, we will prove a lower bound on the speedup for another class of priority switching algorithms that use input priority.

We can define an input priority scheme in a similar way to the output priority scheme by reversing the role of input and output ports and, hence, obtaining the same results above, by using the ϕ-adversary where input and output ports are interchanged. However, we choose to define an input priority scheme differently to take into account the state of the input queues. Therefore, an input priority scheme provides a more interesting framework (note, however, that it will not be a generalization of an output priority scheme).

Before we define an input priority scheme, we introduce the state of a VOQ.

Definition 8: For a matching phase m, let A_{ijm} be a function of time such that $A_{ijm}(t) = A_{ij}(t)$ if $t \in [0, m/S]$, and $A_{ijm}(t) = A_{ij}(m/S)$ otherwise. Similarly, let D_{ijm} be a function of time such that $D_{ijm}(t) = D_{ij}(t)$ if $t \in [0, m/S]$, and $D_{ijm}(t) = D_{ij}(m/S)$ otherwise. The state of a VOQ_{ij} during matching phase m, S_{ijm}, is the tuple (A_{ijm}, D_{ijm}).

[1] π'_m can be obtained by forcing some order on VOQs that are unordered by π_m and do not share an output.

In other terms, the state of VOQ_{ij} during matching phase m is the history of packet arrivals and departures to and from VOQ_{ij} up to the beginning of matching phase m.[2]

Definition 9: An input priority scheme π is a priority scheme that satisfies the following: for every matching phase m, there exists a partial order relation π'_m on the states of the VOQs such that $\mathrm{VOQ}_{ij} \prec_{\pi_m} \mathrm{VOQ}_{kl}$ iff $S_{ijm} \prec_{\pi'_m} S_{klm}$.

According to the definition of an input priority scheme, VOQs with equal VOQ states are unordered and, therefore, have equal priority. An example of an input priority scheme is *largest queue length* where VOQs with more packets have more priority. This scheme was used in the *Central Queue* algorithm [7]. Another example is *oldest HOL packet* where the VOQs with the older HOL packets have more priority. This scheme was used in the *Oldest Cell First* algorithm [3].

Next, we will prove a lower bound result on the speedup required by priority switching algorithms with an input priority scheme. Before we do so, we start with few definitions and lemmas. The reader can skip Definition 10, Definition 11, and Lemma 3 if desired. They are only needed to prove the existence of the next adversary we will consider.

Definition 10: A *ϕ-ordered* $K_{N,N}$ is an $N \times N$ complete bipartite graph with a total order relation ϕ on its edges.

Definition 11: In a ϕ-ordered $K_{N,N}$, an *ℓ-symmetric* cycle is a cycle $n_1, n_2, \ldots, n_{2\ell}, n_1$ (i.e., the nodes alternate on each side of the $K_{N,N}$) that satisfies the following: $(n_{i-1}, n_i) \prec_\phi (n_i, n_{i+1})$ iff $(n_{i-1+\ell}, n_{i+\ell}) \prec_\phi (n_{i+\ell}, n_{i+1+\ell})$ for $i = 1 \ldots \ell$, where n_0 is the same as $n_{2\ell}$ and $n_{2\ell+1}$ is the same as n_1.

Lemma 3: For any $\ell > 2$ and a large enough N, any ϕ-ordered $K_{N,N}$ contains an ℓ-symmetric cycle.

Proof: Let L and R be the two disjoint sets of nodes of $K_{N,N}$. Consider the bipartite graph induced by any $k_L = (\ell - 1)\ell! + 1$ nodes in L and any $k_R = (\ell - 1)k_L! + 1$ nodes in R. Let U and V be the two disjoint sets of nodes of the new bipartite graph. Every node v in V orders the k_L nodes of U according to the order of their respective edges to node v. Since there are at most $k_L!$ possible orders, we can find at least ℓ nodes in V that define the same order ϕ_u on U. Let these nodes be $v_1, v_2, \ldots, v_\ell$ and let V_ℓ be the set $\{v_1, v_2, \ldots, v_\ell\}$. Now every node u in U orders the ℓ nodes of V_ℓ according to the order of their respective edges to node u. Since there are at most $\ell!$ possible orders, we can find at least ℓ nodes in U that define the same order ϕ_v on V_ℓ. Let these nodes be $u_1, u_2, \ldots, u_\ell$ and let U_ℓ be the set $\{u_1, u_2, \ldots, u_\ell\}$. Therefore, we obtain two ordered sets U_ℓ and V_ℓ that satisfy the following properties:

$$(u_i, v) \prec_\phi (u_j, v) \text{ iff } u_i \prec_{\phi_u} u_j \; \forall u_i, u_j \in U_\ell, \; \forall v \in V_\ell$$
$$(u, v_i) \prec_\phi (u, v_j) \text{ iff } v_i \prec_{\phi_v} v_j \; \forall u \in U_\ell, \forall v_i, v_j \in V_\ell.$$

Without loss of generality, let $u_1, u_2, \ldots, u_\ell$ be the ordered elements of U_ℓ and let $v_1, v_2, \ldots, v_\ell$ be the ordered elements of V_ℓ. We can verify that the two cycles of Fig. 4 are ℓ-symmetric cycles. ■

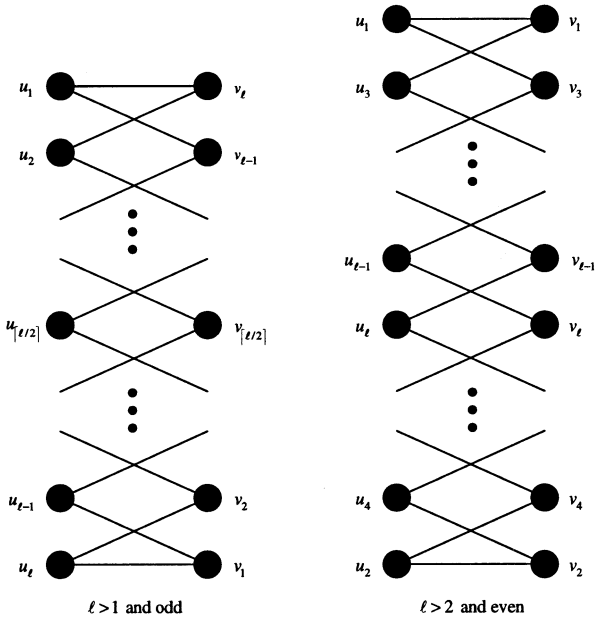

Fig. 4. ℓ-symmetric cycles.

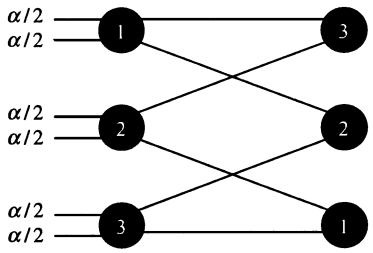

Fig. 5. The 3-symmetric ϕ-adversary.

We define the following adversary.

The 3-symmetric ϕ-adversary: Consider an adversary that generates packets for the (i, j) pairs that form a 3-symmetric cycle in the ϕ-ordered $K_{N,N}$, as shown in Fig. 5. What this essentially means is that our adversary satisfies the following main property: $(1,3) \prec_\phi (2,3) \Leftrightarrow (3,1) \prec_\phi (3,2)$, $(2,3) \prec_\phi (2,1) \Leftrightarrow (3,2) \prec_\phi (1,2)$, and $(2,1) \prec_\phi (3,1) \Leftrightarrow (1,2) \prec_\phi (1,3)$. Definition 10, Definition 11, and Lemma 3 guarantee that for a large enough switch size, such an adversary exists for any total order relation ϕ on the (i, j) pairs. Therefore, as in the case of the ϕ-adversary, the 3-symmetric ϕ-adversary is defined for a given ϕ (which is fixed). The adversary will generate packets for every flow shown in Fig. 5 at a rate of $\alpha/2$, i.e., a packet every $2/\alpha$ time units, producing an α-shaped traffic.

Note that the traffic of the 3-symmetric ϕ-adversary is a theoretical one where two packets from the same input can arrive to the switch simultaneously. This is possible if the VOQs at an input can be accessed independently. For instance, each VOQ is a physically separate queue. In any case, the use of such an adversary can be justified by the following reasoning. We can assume that time is discretized to fixed size intervals of length δ and, hence, as long as packets arrive during the same interval, they will have the same timestamp. Therefore, we can consider a discrete version of the 3-symmetric ϕ-adversary. The discrete

[2]The definition of the VOQ state can be made more general by letting S_{ijm} be the tuple (A_{ij}, D_{ijm}). Therefore, the state of a VOQ_{ij} will additionally incorporate the knowledge of any future packet arrivals to VOQ_{ij}. This definition of the state of a VOQ allows the switching algorithm to somehow foresee the future (as a matter of fact, to know everything about the adversary). Although this is practically impossible, it will only strengthen our result for a lower bound on the speedup. It is important to note, however, that ϕ is fixed by the algorithm prior to any knowledge of the adversary since the adversary is defined in terms of ϕ.

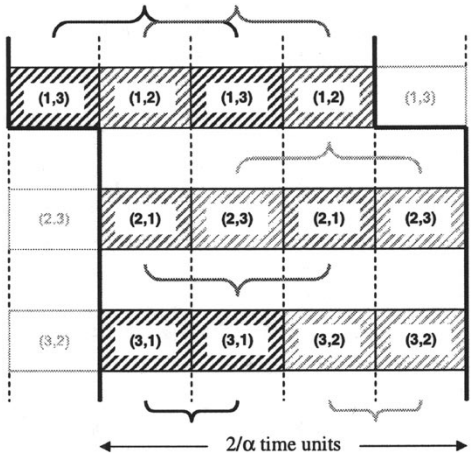

Fig. 6. 3-symmetric ϕ-adversary interleaving packet arrivals.

adversary will write the two packets in parallel to the memory of the input-queued switch by writing a bit of each in an alternating fashion. Since any two write operations to the memory will complete in a particular order, specifically the last two write operations, we still have that one packet will arrive before the other. However, we can prove that for any rational $\alpha\delta$, there exists a line speed (equivalently, a packet size) beyond which any two simultaneous packets in the 3-symmetric ϕ-adversary will arrive during the same interval of length δ in the discrete adversary. Similarly, we can prove that for any rational α/S, there exists a line speed beyond which any two simultaneous packets in the 3-symmetric ϕ-adversary cannot straddle the beginning of a matching phase in the discrete adversary. Therefore, if $\alpha\delta$ and α/S are both rational, there exists a line speed beyond which any two simultaneous packets in the theoretical 3-symmetric ϕ-adversary will appear to arrive simultaneously in the discrete adversary.

Nevertheless, the assumption on the 3-symmetric ϕ-adversary that entails the two packets at an input to arrive simultaneously is actually not needed, and we can work with a more relaxed assumption. We only need that packets of flows (i, j) and (j, i) arrive simultaneously. This can be realized by making the adversary send the packets in parallel; however, unlike the discretization argument described above, we can make the arrival time of packets exact, i.e., a packet of flow (i, j) and a packet of flow (j, i) will arrive at exactly the same time. The idea is that the 3-symmetric ϕ-adversary will generate equal-sized packets, and will divide each packet into two equal chunks and interleave the arrival of these chunks in a way to satisfy the relaxed assumption. Fig. 6 shows a pattern of how this can be done.

As can be seen from the pattern of Fig. 6, the second chunks of packets pertaining to flows (i, j) and (j, i) arrive simultaneously. The pattern can be repeated without wasting any time units as suggested by the dimmed chunks of packets. Therefore, this discretization of the 3-symmetric ϕ-adversary provides the desired property that packets of flows (i, j) and (j, i) always arrive at exactly the same time. In what follows, we will not make an explicit distinction between the 3-symmetric ϕ-adversary and its discrete version. We will assume that the adversary is the 3-symmetric ϕ-adversary of Fig. 5, keeping in mind that there is a way of constructing from it a practical adversary with the desired property that packets of flows (i, j) and (j, i) arrive simultaneously, without requiring that two packets arrive simultaneously at an input.

Next, we prove a lemma similar to Lemma 2 for the case of the 3-symmetric ϕ-adversary.

Lemma 4: For any well-behaved input priority scheme π and any total order relation ϕ on the (i, j) pairs, a π^ϕ-stable matching policy, under the 3-symmetric ϕ-adversary, serves at most two VOQs in each matching phase.

Proof: We will prove that if there are packets at the input side during a matching phase, the π^ϕ-stable matching policy will choose one of the following maximal matchings: {(1,3), (3,1)}, {(1,2), (2,1)}, or {(2,3), (3,2)}. This is enough to conclude that at most two VOQs are served in each matching phase. We will prove this by induction on the number of matching phases.

Base case: The claim is trivially true at a fictitious matching phase before the beginning of the first matching phase.

Inductive step: We assume that the claim is true up to matching phase $m-1$. We need to prove that it remains true for matching phase m. First, we denote by (i, j) and (j, i) two edges belonging to one of the above three matchings. Since the claim is true up to matching phase $m-1$ and the adversary assigns the same traffic to flows (i, j) and (j, i), VOQ_{ij} and VOQ_{ji} will have the same state by the beginning of matching phase m. Second, we can see from Fig. 5 that if (k, l) is adjacent to (i, j) (i.e., either $i = k$ or $j = l$), then (l, k) is adjacent to (j, i); moreover, by the property of the 3-symmetric ϕ-adversary we have $(i,j) \prec_\phi (k,l)$ iff $(j,i) \prec_\phi (l,k)$ (see the above description of the 3-symmetric ϕ-adversary).

If there are no packets at the input side during matching phase m, then we are done. Otherwise, let (i, j) be the edge in the graph such that there is no other edge (k, l) in the graph that satisfies $(k,l) \prec_{\pi^\phi} (i,j)$. Therefore, by the property of the π^ϕ-stable matching, (i, j) will be in the matching during matching phase m. We will prove that (j, i) is also in the matching.

Consider an edge (l, k) in the graph during matching phase m that is adjacent to (j, i). By equality of VOQ states, we know that (k, l) is in the graph during matching phase m (VOQ_{lk} nonempty implies VOQ_{kl} nonempty). We also know that (k, l) is adjacent to (i, j).

Case 1) If $\mathrm{VOQ}_{ij} \prec_{\pi_m} \mathrm{VOQ}_{kl}$, then by equality of VOQ states, $\mathrm{VOQ}_{ji} \prec_{\pi_m} \mathrm{VOQ}_{lk}$.

Case 2) Otherwise, it must be that $\mathrm{VOQ}_{ij} \not\prec_{\pi_m} \mathrm{VOQ}_{kl}$ and $\mathrm{VOQ}_{kl} \not\prec_{\pi_m} \mathrm{VOQ}_{ij}$ and $(i,j) \prec_\phi (k,l)$ by our choice of (i, j). By equality of VOQ states and the property of the 3-symmetric ϕ-adversary, $\mathrm{VOQ}_{ji} \not\prec_{\pi_m} \mathrm{VOQ}_{lk}$ and $\mathrm{VOQ}_{lk} \not\prec_{\pi_m} \mathrm{VOQ}_{ji}$ and $(j,i) \prec_\phi (l,k)$.

Therefore, in both cases, $\mathrm{VOQ}_{ji} \prec_{\pi_m^\phi} \mathrm{VOQ}_{lk}$ for any nonempty VOQ_{lk} such that (l, k) is adjacent to (j, i) and, hence, (j, i) is in the matching during matching phase m by the property of the π^ϕ-stable matching. ■

Theorem 2: For any well-behaved input priority scheme π and any total order relation ϕ on the (i, j) pairs, a π^ϕ-stable

matching policy cannot achieve weak throughput under an α-shaped traffic unless $S \geq 3/2\alpha$.

Proof: We will use the 3-symmetric ϕ-adversary. Consider a time t. By Lemma 1, we have at most $tS+1$ matching phases by time t, each of which forward at most two packets by Lemma 6. Therefore, the number of packets forwarded by time t is at most $2(tS+1)$. By Lemma 1, the number of packets arrived by time t to the switch is at least

$$6\left(\frac{\alpha}{2}t-1\right).$$

Therefore, at time t, the number of packets remaining at the inputs is at least

$$(3\alpha-2S)t-8.$$

For $S < 3/2\alpha$, $(3\alpha-2S) = \delta > 0$. If weak throughput is to be achieved, then for every $\epsilon > 0$, there must exist a large enough t, say, t_0, such that for every VOQ_{ij}, $(X_{ij}(t))/(t) \leq \epsilon$ for any $t \geq t_0$. Assume that weak throughput is achieved and let $\epsilon < \delta/6$ and t_0 be as defined above. Let $t \geq t_0$ be such that $\delta/6 - 8/6t > \epsilon$. Since at the inputs we have at most six nonempty VOQs, there exists a VOQ_{ij} such that the number of packets remaining in VOQ_{ij} at time t is at least $(\delta t-8)/(6)$. Therefore

$$\frac{X_{ij}(t)}{t} \geq \frac{\delta}{6}-\frac{8}{6t} > \epsilon.$$

Since $t \geq t_0$, we have a contradiction. ∎

We have proved that any switching algorithm based on an input priority scheme that breaks ties using the indices of the ports cannot achieve weak throughput under an α-shaped traffic unless $S > 3/2\alpha$. For instance, the *Central Queue* and the *Oldest Cell First* switching algorithms cannot achieve weak throughput unless $S \geq 3/2$, under the assumption that indices of the input and output ports are used to break ties when two VOQs have the same priority (i.e., same VOQ length and same age of HOL packet, respectively). We can prove that the *Oldest Cell First* switching algorithm cannot achieve weak throughput even without the above tie-breaking assumption, by considering the discrete version of the 3-symmetric ϕ-adversary shown in Fig. 6. This adversary creates a traffic in which packets of flows (1,2) and (2,1) are delayed by $(1)/(2\alpha)$ time units, and packets of flows (2,3) and (3,2) are delayed by $1/\alpha$ time units. In that case, the *Oldest Cell First* algorithm will have to choose matchings in a way similar to before, forwarding only two packets per matching phase, because the VOQs that will have the oldest HOL packets are the ones that belong to one of the three matchings listed previously.

Theorem 2 suggests that a speedup of at least $3/2$ is required for an input priority switching algorithm to provide throughput with a full loading of the switch. Next, we prove a corollary.

Corollary 4: For any input priority scheme π, weak throughput is not reducible to a π-stable matching policy unless $S \geq 3/2$.

Proof: There exists a well-behaved input priority scheme π' such that $\mathrm{VOQ}_{ij} \prec_{\pi_m} \mathrm{VOQ}_{kl} \Rightarrow \mathrm{VOQ}_{ij} \prec_{\pi'_m} \mathrm{VOQ}_{kl}$ for every matching phase m.[3] Therefore, for any total order relation ϕ on the (i, j) pairs, $\mathrm{VOQ}_{ij} \prec_{\pi_m} \mathrm{VOQ}_{kl} \Rightarrow \mathrm{VOQ}_{ij} \prec_{\pi'^{\phi}_m} \mathrm{VOQ}_{kl}$ for every matching phase m. Hence, a π'^{ϕ}-stable matching policy is a π-stable matching policy and the result is immediate from Theorem 2 using $\alpha = 1$. ∎

Now, we discuss the enhanced version of the LOOFA algorithm presented in [8] which uses a combined input–output priority scheme. Although LOOFA assumes that only one packet can arrive to an input port per time unit (which is not true with the 3-symmetric ϕ-adversary), we will show that the priority scheme of LOOFA does not imply weak throughput under an α-shaped traffic if $S < 3/2$.

LOOFA computes a matching in the following way. It finds the port with the smallest output queue and selects an input with which to match it, then repeats until the matching is maximal. In the deterministic version of LOOFA, the input selection criterion can be either the input with the oldest HOL packet, or it can be performed in a round-robin fashion. We can show that this combined input–output priority scheme also suffers the same limitations. We will not go into the details; we will just illustrate a sketch of the proof.

Consider the example of Fig. 5. Let $S < (3/2)\alpha$, which means that $2/\alpha < 3/S$. Since every VOQ accumulates packets at a rate of $\alpha/2$, every three matching phases, a VOQ will receive a new packet. This means that a policy can continuously select the following matchings in that order:

$$\{(1,3),(3,1)\},\{(1,2),(2,1)\},\{(2,3),(3,2)\}.$$

One can show, irrespective of the speedup of the switch, that this order in choosing the matching satisfies the smallest output queue criterion, assuming that forwarded packets arrive at the same time to their output queues and that output queues are served as soon as possible (there is no restriction on the order in which packets are delivered at the output). This assumption can be justified by an adversary that controls the timing of the algorithm. Moreover, this order in choosing the matching satisfies three input selection criteria: *round-robin*, *oldest HOL packet*, and *largest queue length*. Since a matching of size 2 is computed in every matching phase, weak throughput cannot be achieved, as proved in Theorem 2.

VIII. Conclusion

We proved lower bounds on the speedup required by several classes of switching algorithms to achieve weak throughput. By doing so, we showed that most of the practical switching algorithms suffer the same theoretical limitation, which is the fact that speedup cannot be avoided for throughput to be guaranteed. An algorithm based on a Birkhoff–von Neumann decomposition of the rate matrix that provides a delay guarantee with no speedup under a strong constant burst traffic has been suggested in [2]. This algorithm requires a preprocessing step on the rate matrix of $O(N^{4.5})$ time complexity, but after that it runs in $O(\log N)$ time. Theoretically speaking, we can disregard the preprocessing step because it occurs only once; how-

[3] π'_m can be obtained by forcing some order on VOQs that are unordered by π_m and do not have equal states.

ever, the algorithm still requires an explicit knowledge of the rates λ_{ij}s and, therefore, its correct operation is sensitive to the exact values of the λ_{ij}s. The results presented here suggest that using the practical switching algorithms known so far (running in $O(N^2)$ or $O(N^2 \log N)$ time with no prior knowledge of the rates) requires a speedup in the switch. As a future work, it would be interesting to prove lower bounds on the speedup for particular running times of the switching algorithm. In this paper, we did not consider the family of iterative switching algorithms (e.g., Parallel Iterative Matching (PIM) [1], iSLIP [12], iLQF and iOCF [10], Dual Round Robin (DRR) [9], and Prioritized Dual Round Robin (pDRR) [6]), but similar limitations for most of these algorithms can also be proved. For instance, under any 3-symmetric ϕ-adversary and our assumption of breaking ties, both iLQF and iOCF compute a π^{ϕ}-stable matching, where π is the *largest queue length* for iLQF and the oldest HOL packet for *iOCF* (although they do no do it using the greedy approach of Fig. 2). Moreover, we can also prove that iSLIP and DRR require a speedup $S > 3/2\alpha$ to achieve weak throughput. Using the 3-symmetric ϕ-adversary (although ϕ is not an important parameter in this case), the round-robin pointers in both of these algorithms can exhibit a behavior in which only two VOQs will be served in each matching phase, namely, VOQ_{13} and VOQ_{31}, VOQ_{12} and VOQ_{21}, and VOQ_{23} and VOQ_{32}.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their very useful comments and suggestions.

REFERENCES

[1] T. E. Anderson, S. Owicki, J. Saxes, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319–352, Nov. 1993.

[2] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "On service guarantees for input-buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann," in *Proc. 7th Int. Workshop Quality of Service (IWQOS)*, 1999, pp. 79–86.

[3] A. Charny, P. Krishna, N. Patel, and R. Simcoe, "Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup," in *Proc. 6th Int. Workshop Quality of Service (IWQOS)*, May 1998, pp. 235–244.

[4] S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queuing with combined input-output queued switches," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1030–1039, June 1999.

[5] J. G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proc. IEEE INFOCOM*, 2000, pp. 556–564.

[6] G. Damm, J. Blanton, P. Golla, D. Verchere, and M. Yang. Fast scheduler solutions to the problem of priorities for polarized data traffic. Alcatel USA, Richardson, TX. [Online]. Available: http://www.utdallas.edu/~meiyang/papers/ist01.pdf.

[7] A. Kam, K.-Y. Siu, and R. Barry, "A cell switching WDM broadcast LAN with bandwidth guarantee and fair access," *J. Lightwave Technol.*, vol. 16, pp. 2265–2280, Dec. 1998.

[8] P. Krishna, N. S. Patel, and A. Charny, "On the speedup requirement for work-conserving crossbar switches," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1057–1069, June 1999.

[9] Y. Li, S. Panwar, and H. J. Chao, "Saturn: a terabit packet switch using dual round-robin (DRR)," *IEEE Commun. Mag.*, vol. 38, pp. 78–84, Dec. 2000.

[10] N. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. Dissertation, Univ. California, Berkeley, May 1995.

[11] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proc. IEEE INFOCOM*, vol. 1, Mar. 1996, pp. 296–302.

[12] N. Mckeown, "The iSLIP scheduling algorithm for input queues switches," *IEEE/ACM Trans. Networking*, vol. 7, pp. 188–201, Apr. 1999.

[13] A. Mekkittikul and N. McKeown, "A starvation-free algorithm for achieving 100% throughput in an input-queued switch," in *Proc. Int. Conf. Computer Communication and Networking*, Oct. 1996, pp. 226–231.

[14] ——, "A practical scheduling algorithm to achieve 100% throughput in input-queued switches," in *Proc. IEEE INFOCOM*, vol. 2, Mar. 1998, pp. 792–799.

[15] Y. Tamir and H. Chi, "Symmetric crossbar arbiters for VLSI communication switches," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, pp. 13–27, Jan. 1993.

[16] L. Tassiulas and A. Ephremides, "Stability properties of constrained queuing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.*, vol. 37, pp. 1936–1949, Dec. 1992.

Saad Mneimneh was born in Beirut, Lebanon, in 1973. He received the B.E. degree in computer and communication engineering from the American University of Beirut (AUB) in 1995, and the S.M. degree in information technology and the Ph.D. degree in communication networks from the Massachusettes Institute of Technology, Cambridge, in 1997 and 2002, respectively.

He is currently an Assistant Professor in the Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX. His interests include various algorithmic aspects of high-speed network switches, optical routing, and in general networking and graph problems and their applications to other disciplines like biology.

Kai-Yeung Siu received the B.S. degree (*summa cum laude*) in mathematics and computer science from New York University, New York, NY, and the B.Eng. degree (*summa cum laude*) in electrical engineering from The Cooper Union, New York, NY, both in 1987. He received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1988 and 1991, respectively.

From 1989 to 1990, he was a Research Student Associate with the IBM Almaden Research Center, San Jose, CA. From 1991 to 1995, he was Assistant Professor of electrical and computer engineering with the University of California, Irvine. He joined the Massachusetts Institute of Technology, Cambridge, in 1996, as an Associate Professor. He was with the d'Arbeloff Laboratory for Information Systems and Technology of Mechanical Engineering and also affiliated with the Laboratory for Information and Decision Systems of Electrical Engineering and Computer Science. He was the founding research director of the MIT Auto-ID Center, an industry-funded center which develops next-generation automatic identification systems with e-commerce applications. He has also served as a Consultant for major Internet equipment vendors and service providers. He has published over 100 research papers in the areas of optical networking, wireless communications, Internet routing and congestion control protocols, parallel and distributed algorithms, and computational complexity theory.

Dr. Siu was a recipient of the d'Arbeloff Career Development Chair at MIT. He also received a National Science Foundation Young Investigator Award in 1993, the UC Irvine Distinguished Assistant Professor Award in 1995, the IEEE Browder J. Thompson Memorial Prize Paper Award in 1997, and the Best Paper Award of the SPIE Conference on All-Optical Networking in 1998. He has served on the Editorial Board of the IEEE/ACM TRANSACTIONS ON NETWORKING.